

Power Extraction Circuits for Piezoelectric Energy Harvesters and Time Series Data in Water Supply Systems

James Dicken

Civil Engineering Department
Imperial College London

PhD Thesis

May 2013

Abstract

This thesis investigates two fundamental technological challenges that prevent water utilities from deploying infrastructure monitoring apparatus with high spatial and temporal resolution: providing sufficient power for sensor nodes by increasing the power output from a vibration-driven energy harvester based on piezoelectric transduction, and the processing and storage of large volumes of data resulting from the increased level of pressure and flow rate monitoring.

Piezoelectric energy harvesting from flow-induced vibrations within a water main represents a potential source of power to supply a sensor node capable of taking high-frequency measurements. A main factor limiting the amount of power from a piezoelectric device is the damping force that can be achieved. Electronic interface circuits can modify this damping in order to increase the power output to a reasonable level. A unified analytical framework was developed to compare circuits able to do this in terms of their power output. A new circuit is presented that out-performs existing circuits by a factor of 2, which is verified experimentally.

The second problem concerns the management of large data sets arising from resolving challenges with the provision of power to sensor devices. The ability to process large data volumes is limited by the throughput of storage devices. For scientists to execute queries in a timely manner, query execution must be performant. The large volume of data that must be gathered to extract information from historic trends mandates a scalable approach. A scalable, durable storage and query execution framework is presented that is able to significantly improve the execution time of user-defined queries.

A prototype database was implemented and validated on a cluster of commodity

servers using live data gathered from a London pumping station and transmission mains. Benchmark results and reliability tests are included that demonstrate a significant improvement in performance over a traditional database architecture for a range of frequently-used operations, with many queries returning results near-instantaneously.

Contents

1	Introduction	27
1.1	Power for Sensor Nodes	31
1.2	Research Objectives	33
2	Background	35
2.1	Experimental Programme	36
2.2	Data Requirements	39
2.3	Time-series data	41
2.4	Powering Sensor Nodes	43
2.5	Energy from Fluid Flow	46
2.6	Piezoelectric Transduction	57
2.7	Conclusion	60
3	Development and Implementation of a Distributed Query Architecture for Water Network Research	61
3.1	Requirements	62

3.2	Existing Systems for Large Time-Series Datasets	66
3.3	Summary	76
3.4	<i>Frames</i> Implementation Overview	78
3.5	Preliminary Concepts and Assumptions	79
3.6	Distributed Storage Architecture	81
3.7	Node-Local Append Atomicity	91
3.8	Operator Definition	94
3.9	Query DSL Structure	101
3.10	Indexing	104
3.11	Query Evaluation	106
3.12	Validation and Performance Tests	114
4	Analysis of Power-Electronic Interface Circuits for Piezoelectric Energy Har-	
	vesting	125
4.1	Voltage Profile Modification	127
4.2	Modelling of the Electromechanical System	129
4.3	Methodology	131
4.4	Purely Resistive Load	133
4.5	Tuned-Out Shunt Capacitance	134
4.6	Rectified DC Load	135
4.7	Synchronous Switched Extraction	137
4.8	Synchronous Switched Extraction with DC Output	139

4.9	Piezoelectric Fixed Voltage Control	143
4.10	Parallel SSHI with Resistive Load	146
4.11	Parallel SSHI with DC Output	149
4.12	Series SSHI with Resistive Load	151
4.13	Series SSHI with DC Output	153
4.14	Parallel SSHI with Synchronous Extraction	155
5	Improved Power Output by Pre-Biasing	161
5.1	Pre-Biasing	162
5.2	Single-Supply Pre-Biasing	167
5.3	Comparisons and Conclusions	174
5.4	Further Work	177
6	Experimental Validation of Pre-Biasing Techniques	179
6.1	Simulation	180
6.2	Experimental Results	188
6.3	Summary	196
7	Conclusions	197
7.1	Summary and Contributions	197
7.2	Energy Harvesting Interface Circuits	199
7.3	Data Management	200

8 Further Work	203
8.1 Limitations of the Analytical Model	203
8.2 Pre-Biasing Experimental Programme	204
8.3 Data Management	204
Appendices	207
A Pre-biasing experimentation PCB Layout	209
B Protocol Definitions	213
B.1 Client to Node	213
B.2 Node to Node	214
C Random Dataset Generator	217

Declaration of Originality

Unless specified otherwise or attributed, all of the work in this thesis is my own.

Copyright Notice

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work

List of Figures

1.1	Stress-rupture curve of high-density polyethylene water pipe commonly used in the water industry (reproduced from [4]).	29
1.2	Manhole access to an underground pressure-reducing valve chamber.	32
2.1	<i>InfraSense</i> ^{HP} sensor node showed affixed to 3 MW mains distribution pump in a London pumping station.	37
2.2	Overview of deployed system showing main hardware components.	38
2.3	The daily demand fluctuation of a farm in the South of England, recorded in 2011.	38
2.4	Abnormal flow rate showing large rate of change recorded on June 26th 2011.	39
2.5	2nd generation <i>InfraSense</i> low-power sensor node (A. Hoskins, personal communication).	44
2.6	Mains water access chamber showing installation of both ABB insertion flow meter and <i>InfraSense</i> data logger.	45
2.7	Pressure-reducing valve installation, with CLAVAL bypass turbine, in simulated recirculating flow test environment (Source: I. Stoianov, <i>InfraSense</i> lab)	48
2.8	Flow-induced vibration phenomenon downstream of a fluff placed in a water flow. Flow direction is left-to-right. (Source: [37]).	49

2.9	Separation of flow, from van Dyke [38]. A is the leading edge, S are points of separation, B is the trailing edge of low pressure, and V is the vortex formed by the interaction of the fluid with the boundary layer.	50
2.10	Bluff body vortex-shedding energy harvester.	51
2.11	Bluff body vortex shedding energy harvester with flat plate.	52
2.12	Pneumatically mounted bluff assembly, with the direction of flow highlighted.	54
2.13	Printed target used to estimate in-plane displacement of bluff body holder assembly.	54
2.14	Interface of position detection software, showing raw feed from camera (top left), and position estimation of control dots shown by crosshairs (right).	55
2.15	Experimental data showing oscillation of 6cm bluff body placed in open-channel water flume moving at 0.25 m/s (top), and the frequency content of this signal (bottom) showing a predominant frequency of 0.3 Hz.	56
2.16	Diagram of a model of a piezoelectric cantilever.	57
3.1	Example MySQL schema for time-series data storage.	67
3.2	Screenshot of a live <i>Frames</i> installation.	79
3.3	Distribution of data amongst nodes in a cluster with no replication, showing master nodes for streams A and B storing pointers to the location of data elements.	86
3.4	Two example streams, ABC and XYZ , split up into frames.	87
3.5	Flow diagram showing distributed append operation between master node, cohort nodes, and client.	89
3.6	State-flow diagram of the transactional append logic, showing absence of an intermediate state where data can be lost. Each path through the diagram represents the operations performed to the slave (b) and master (a) files.	93

3.7	Computation of a moving-window filter on a dataset.	97
3.8	8th-order moving-window average filter. The filter sums the value at each position multiplied by the associated coefficient, then the entire filter moves forward one place and the process is repeated.	99
3.9	Format of index block record.	106
3.10	Example of automatic translation of constant sub-expression into numerical term. . .	108
3.11	Example of replacement of reducible fact with numerical value.	109
3.12	Simplified example of a data stream split up into five frames, with query bounds starting within a frame and extending past the end of data.	112
3.13	Time-range filter on a five-frame dataset, showing elimination of data and operator aggregating the data of the returned time ranges.	113
3.14	Data insert performance for a single server. Aggregate throughput across all inserting nodes is measured along with the proportion of insert operations that complete within 1 second (grey line) and 2 seconds (black line).	115
3.15	Data insert performance for a cluster of 10 servers. Aggregate throughput across all inserting nodes is measured along with the proportion of insert operations that complete within 1 second (grey line) and 2 seconds (black line).	116
3.16	Downsampled view of 32 GB sample stream rendered by <i>Frames</i>	118
3.17	Query execution of aggregation function, showing returned time ranges from a 32 GB data stream.	120
3.18	Chart showing the latencies to queries during a simulated node failure event. The simulated failure takes place at $t = 2000$	123

4.1	Inertial energy harvester with parasitic damping and displacement-constrained travel. D_p and D_e are the parasitic and electrical, Z_l is the internal range of motion, $y(t)$ is the instantaneous absolute position of the harvester, $x(t)$ is the instantaneous absolute position of the proof mass m , and $z(t)$ is the instantaneous position of the proof mass relative to the harvester.	126
4.2	Comparison sketch of piezoelectric capacitance voltage profiles; 1) reference mechanical position (nominal amplitude), 2) optimal voltage profile in order to maximise damping and hence energy generation, 3) voltage profile of purely resistive load, and 4) voltage profile of Shenck's synchronous extraction circuit.	129
4.3	Microgenerator with piezoelectric transduction.	130
4.4	Microgenerator equivalent circuit with low electromechanical coupling.	130
4.5	Simplified piezoelectric model. The open-circuit amplitude of $V_{C_p}(t)$ is V_{po}	131
4.6	Piezoelectric energy harvester connected to a purely resistive load.	134
4.7	Full-bridge rectifier with output smoothing capacitor.	135
4.8	Scale sketch of piezoelectric input current, piezoelectric capacitance voltage and output current (labeled I_{out} in Figure 4.7) in the presence of a bridge rectifier with a fixed output voltage V_{out}	136
4.9	Synchronous switched extraction circuit.	138
4.10	Conceptual sketch of the voltage waveform on the piezoelectric capacitance C_p for the circuit of Figure 4.9. The length of the discharge phase is exaggerated for illustration; in practice the discharge will be relatively instantaneous in most applications provided $RC \ll 1/\omega$	138
4.11	Circuit for synchronous switched extraction with DC output.	139

4.12	Contributions to total power output (P_{out}) from free-wheeling (P_{fw}) and direct conduction (P_{ps}), and their sum, as a function of output voltage, for the circuit of Figure 4.11. At $V_{out} = V_{opt}$ the contribution from free-wheeling is zero. Graph plotted in MAPLE 14.01.	140
4.13	Conceptual sketch of the voltage waveform on the piezoelectric capacitance, with the main voltages highlighted: the voltage at the start of generation, $\pm V_D$, and the voltage at the start of charge-flipping, $\pm(2V_{po} - V_D)$	141
4.14	Resonant current path for charge extraction.	142
4.15	Circuit diagram of piezoelectric fixed voltage control.	144
4.16	Operational sketch of the piezoelectric fixed voltage control circuit, including mechanical motion (top), C_p voltage (middle) and control signals for each switch (bottom).	144
4.17	Piezoelectric fixed voltage control (with charge cancelling).	145
4.18	SSHI with resistive load.	146
4.19	Scale sketch of the voltage waveform of SSHI, showing build-up to steady-state operation.	147
4.20	Parallel SSHI with DC output.	149
4.21	Conceptual sketch of the voltage waveform on the piezoelectric capacitance C_p for the circuit of Figure 4.20.	150
4.22	Series SSHI with resistive load.	152
4.23	Series SSHI with rectified DC load.	153
4.24	Sketch of the steady-state operating cycle of Series SSHI circuit.	154
4.25	Parallel SSHI circuit with synchronous extraction.	156

4.26	Sketch of the voltage on the piezoelectric capacitance in steady-state of the Parallel SSHI with Synchronous Extraction circuit.	156
5.1	Pre-biasing circuit diagram, showing optional circuit components for non-optimal output voltage operation (D_A , D_B , D_C , D_D and D_E) in grey.	162
5.2	Basic sketch of the voltage waveform on the piezoelectric capacitance for the circuit of Fig. 5.1.	163
5.3	Simulated operation over a few mechanical cycles of the circuit of Fig. 5.1, generated with <i>ORCAD PSpice</i> version 16.3. The LHS and RHS control signals refer to the closing of switch pairs S_1, S_4 and S_2, S_3 respectively. The LHS and RHS extraction control signals refer to the closing of switches S_6 and S_5 respectively. Main circuit parameters: $f=80$ Hz, $L=5$ mH, inductor series resistance $R_L=3 \Omega$, $C_p=65$ nF. . . .	164
5.4	Single-supply pre-biasing circuit diagram.	168
5.5	Sketch of voltage waveform on the piezoelectric capacitance for the circuit of Figure 5.4.	169
5.6	Operational waveforms of a single charge/discharge cycle of the circuit of Figure 5.4 generated by time-domain simulation.	169
5.7	Comparison between the pre-biasing techniques in terms of output power. Graph plotted by evaluating analytical power expressions for the two circuits for component values: $C_p=100$ nF, $Q=10$, $V_D=0.7$ V, $f=100$ Hz. Graph plotted using MAPLE 14.01 with expressions (5.13) and (5.21).	171
5.8	Comparison of circuit power gain when compared to rectified DC load against Q for $V_D = 0$. Graph plotted continuously using MAPLE 14.01 from the analytical expressions obtained for the optimised power output for each circuit.	175
5.9	Comparison of power output for circuits with DC output against open-circuit voltage V_{po} Graph plotted continuously using MAPLE 14.01 from the analytical expressions obtained. Circuit parameters were $V_D = 0.7$, $Q = 10$, $C_p = 100$ nF.	177

6.1	Pre-biasing circuit layout represented in <i>OrCAD Capture</i> 16.3 showing main circuit components and drive signal generators.	181
6.2	Simulation waveforms. Top: Main voltage waveforms showing voltage across piezo-electric capacitance, pre-charge supply voltage (V_{cc}) and output voltage (V_{out}); Middle: left and right charge pulses; Bottom: charge extraction pulses	182
6.3	Single-Supply Pre-Biasing circuit layout represented in <i>OrCAD Capture</i> 16.3 showing main circuit components and drive signal generators. Each gate drive signal (A and B) is driven by two different signal generators, one for the discharge and one for the charge cycle.	184
6.4	Simulation waveforms for single-supply pre-biasing circuit with $V_{cc} = 5$ (non return-to-zero regime). Top: current in the piezoelectric capacitance; Middle: left and right control signals (showing charge and discharge); Bottom: Voltage across the piezoelectric capacitance.	185
6.5	Simulation waveforms for single-supply pre-biasing circuit with $V_{cc} = 60.78$ (return-to-zero regime). Top: current in the piezoelectric capacitance; Middle: left and right control signals (showing charge and discharge); Bottom: Voltage across the piezoelectric capacitance.	185
6.6	Block diagram of DSP implementation for controlling gate-drive signals.	189
6.7	Oscilloscope trace showing gate drive signals (highlighted), the voltage on the piezo-electric capacitance (bottom) and the estimated position of the mechanical device (top).	190
6.8	Photograph showing the prototype pre-biasing circuit constructed on the PCB of the design of Figures A.1 & A.2.	191
6.9	Operation of pre-biasing circuit (on PCB) showing phase-shifted excitation waveform (bottom), DSP-estimated position (top) and gate-drive firing signals (top; overlaid on position waveform). Scales are arbitrary.	192

6.10	Operation of pre-biasing circuit (on PCB) under small input excitation, showing phase-shifted excitation waveform (bottom), DSP-estimated position (top) and gate-drive firing signals (top - overlaid on position waveform). Scales are arbitrary. . . .	192
6.11	Single-Supply Pre-Biasing circuit bi-directionally blocking and conducting switch implementation.	194
6.12	Experimentally measured waveforms showing piezoelectric voltage over 2 cycles (top), and an expanded view of this voltage and of the inductor current during a transition (bottom).	195
A.1	Top of PCB layout for first embodiment of Pre-Biasing Circuit, designed using CadSoft EAGLE Version 5.6.0.	210
A.2	Reverse of PCB layout for first embodiment of Pre-Biasing Circuit, designed using CadSoft EAGLE Version 5.6.0.	211

List of Tables

2.1	Storage requirements of 16-bit sampling (with 64-bit integer timestamp), and time taken to read dataset from disk at 50 MB/sec.	42
3.1	Example sample values.	83
3.2	Normalised insertion throughput for a 10 node cluster, calculated by dividing the aggregate throughput of the cluster by the single-node throughput.	117
3.3	Test results (min, mean, max) for simple aggregators, comparing the specific performance of queries between <i>Frames</i> and two other traditional implementations. All times are in seconds unless otherwise specified. (1) This test was conducted with a 400 GB file and extrapolated due to size constraints on a single system. (2) This dataset exceeded the maximum table size of the <i>MySQL</i> server.	121
4.1	Component parameters for harvester configurations.	132
5.1	Summary of power output of circuits with resistive load.	174
5.2	Summary of non-volume-constrained power output for circuits with DC load for $V_D = 0$	174
5.3	Summary of non-volume-constrained optimal power supply voltages for $Q=10$, $V_D=0.5$.	176

6.1	Comparison of model prediction and <i>PSpice</i> simulation results for the Single-Supply Pre-Biasing circuit run at 3.3 and 5 V (for interfacing to sensor nodes) and at optimal voltages for the circuit.	186
-----	---	-----

Acknowledgements

I am very fortunate to have had the expert guidance of two fine researchers during the course of this work: Dr Paul Mitcheson (PDM97) and Dr Ivan Stoianov. I am very grateful for their inspiration, knowledge and assistance. I would also like to thank Alwyn Elliott for his helpful contribution to the experimental results.

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/G070180/1, “Next Generation Energy-Harvesting Electronics: Holistic Approach” (website: www.holistic.ecs.soton.ac.uk) and by EPSRC project EP/E003192/1, “Delivering sustainable water systems by optimising existing infrastructure via improved knowledge, understanding and technology - project NEPTUNE”.

Preface

This thesis investigates two fundamental technological challenges which are critical to enable water utilities to deploy infrastructure monitoring systems with high spatial and temporal resolution: increasing the power output from a vibration-driven energy harvester based on piezoelectric transduction, and the processing, storage and querying of large volumes of data resulting from the increased level of monitoring. The work is presented as follows:

- Chapter 1 presents the motivation for this research, and background on the application area of the water industry.
- Chapter 2 is an analysis of the potential energy sources for a remote sensor in a mains distribution pipeline. It identifies the motivation for using an energy harvester based on flow-induced vibrations with piezoelectric transduction, and identifies the main limitation on the energy output of this configuration as the available damping forces that can be achieved.
- Chapter 3 discusses the development and implementation of a research database tool that is able to capture real-time high-frequency data from sensor nodes in the utility water network, and efficiently distribute the required analysis tasks amongst several computers in a cluster, and a framework for efficiently executing user-defined queries on that dataset.
- Chapter 4 presents a review of the state-of-the-art of power-extraction circuits for piezoelectric energy harvesters in a unified analytical framework that allows them to be objectively compared in terms of power output.
- Chapter 5 presents the development of a power electronic interface circuit that is able to out-perform the existing power extraction circuits by a factor of 2.

- Chapter 6 presents the results and analysis of an experimental programme that was conducted to verify the performance of the pre-biasing circuit topologies of Chapter 5.
- Chapter 7 presents the conclusions of this research and discusses the limitations to the work.
- Chapter 8 outlines the areas for further development.

List of Contributions

- Conference paper:
J. Dicken, P. Mitcheson, I. Stoianov, and E. Yeatman, “Increased power output from piezoelectric energy harvesters by pre-biasing” *Power-MEMS 2009, Washington DC, USA, December 1-4, 2009*, pp. 75-78, 2009.
14 citations.
- Conference paper (and presentation):
J. Dicken, P. D. Mitcheson, A. Elliott, and E. M. Yeatman, “Single-supply pre-biasing circuit for low-amplitude energy harvesting applications” *PowerMEMS, Seoul, Republic of Korea, November 15-18*, pp. 46-49, 2011.
7 citations.
- Journal paper:
J. Dicken, P. D. Mitcheson, I. Stoianov, and E. M. Yeatman, “Power-Extraction Circuits for Piezoelectric Energy Harvesters in Miniature and Low-Power Applications” *IEEE Transactions on Power Electronics*, Volume 27, Issue 11, pp. 4514-4529, Nov. 2012.
10 citations.

Chapter 1

Introduction

Water utilities across the world face significant operational and financial challenges in managing ageing pipeline infrastructure while providing high-quality services. A critical pre-requisite for successful asset and operational management is the availability of data with a sufficient temporal and spatial resolution. Such data can also provide optimal near-real-time response to emergency situations and prevent or mitigate the consequences of failures of key components of a water supply system. While many water utilities have been steadily increasing the level of monitoring, the significant technological barriers that currently exist to a wide-spread deployment have constrained this effort to the minimum level of regulatory requirements. Consequently, the available operational data is generally low-duty cycle, does not capture the system dynamics, has limited value and is frequently not utilised for near-real-time condition and performance monitoring of primary assets such as pumps and water transmission mains.

Effective past failure and future risk analysis enables the development of critical risk and asset management solutions. This requires the derivation of informed decisions based on historical data and using these to support future decisions. The main concern is extending the life cycle of the ageing infrastructure. 50% of pipes in the Thames Water region are more than 100 years old and 30% are over 150 years old. The motivation is to derive optimal pressure conditions which minimise changes in the pipe stress conditions, thus extending the life cycle of the infrastructure.

Distribution pipes have defined upper-limits in terms of the static pressure they are able to withstand. Pipework that is able to withstand greater static pressure has a higher associated cost due to the thicker or denser materials used [1], and the water companies are keen to minimise the pipe diameter as much as possible provided the minimum acceptable delivery pressure can be achieved [2]. Failure due to a steady-state over-pressure condition is rare [3], but a combination of age, degradation of material and the accumulated static loading can lower the effective safe working pressure of a pipe. The stress-rupture curve follows a predictable knee curve as shown in Figure 1.1 [4].

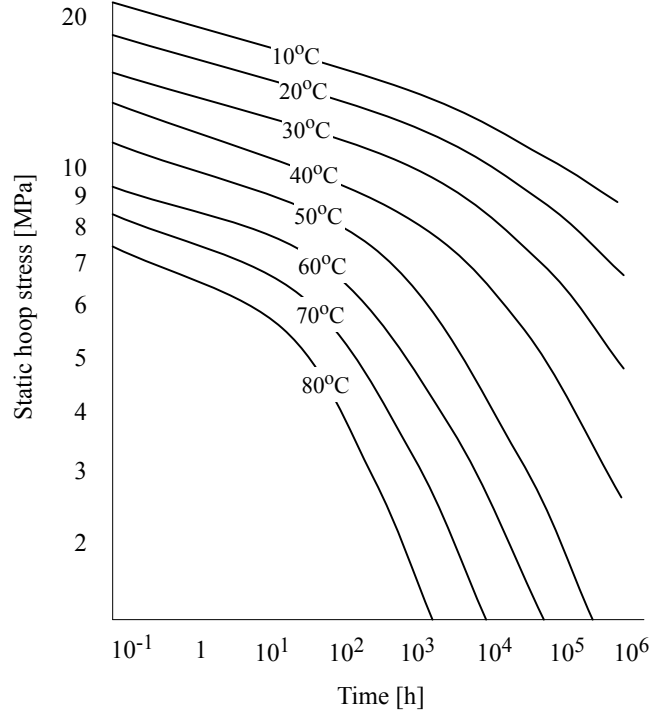


Figure 1.1: Stress-rupture curve of high-density polyethylene water pipe commonly used in the water industry (reproduced from [4]).

Cumulative pressure fatigue, calculated by a simple time-integral of pressure (and in some cases temperature) has been shown to be a reliable metric to predict future pipe failure [5, 3]. This leads to the conclusion that a significant proportion of bursts could be predicted based on this simple calculation alone. However, the large size of the dataset means that although conceptually trivial, the summation would take a significant amount of time to compute for large installations. This specific failure prediction metric could be continuously updated as new data arrives by a bespoke system, but this would imply too tight a coupling between the architecture and a specific query for a system to have general applicability.

There is an additional, and significant, risk to a mains-pressure distribution pipe from rapid changes in operating pressure and flow. This effect has received significant attention in the literature [6, 7]. Various research programmes have shown that in addition to static pressure, instantaneous rate of pressure change is a good indicator for the stress experienced by a pipe wall. The analysis of the fluid dynamics of rapid pressure fluctuations in [8] suggests that the instantaneous rate of change

of pressure can be used to model the unsteady flow conditions that arise from such occurrences. The burst detection and localisation algorithm proposed in [9] uses the propagation of the transient wave (the rate of change of pressure) throughout a pipe caused by a sudden burst.

The extreme case of this phenomenon, called water hammer [10, 11], occurs when a flow is forced to suddenly stop or change direction, and the resulting pressure wave can cause significant damage [12, 13]. In [14] (1928) it is argued that there is significant risk to a metal surface from erosion caused by rapid changes in pressure and flow rate experienced on the surface of a propellor blade; the author notes:

[...] the remarkable ability of water and other nearly incompressible fluids, under certain circumstances to produce momentary pressures far in excess of the yield-point strength of ordinary materials.

These pressure transients carry an associated cost for a water company as equipment must either be replaced more frequently or higher-rated infrastructure must be used. If data about historical instantaneous demand were readily available, a water utility would be able to use this data to characterise the system behaviour, which would ultimately facilitate near-real-time failure detection. In addition, a company may choose to expose the cost of unfavourable demand characteristics to the customer in order to recover the associated additional cost and potentially discourage such behaviour. Electrical suppliers charge large customers for reactive loads [15], since they increase the cost to the supplier of delivering a given unit of energy. Electrical suppliers are able to implement this charging structure because detailed monitoring of electrical grid infrastructure is ubiquitous. The ability to rapidly compute rate-of-change information for a historical signal, and to perform a search predicated on this value, would allow a scientist working with a water utility to quickly acquire an understanding as to the historic stress that the pipe has undergone and thereby an assessment of the likelihood of future failures.

Water is delivered to residential consumers in district-metered areas (DMAs) of typically fewer than 1500 customers. Each DMA is fed from a single (or in some cases multiple) metered inlet. At the boundary of the DMA is a pressure-reducing valve, which maintains the pressure within the DMA by modulating a variable orifice based on demand [16]. The leakage in a DMA can be

estimated by taking a flow rate reading during a period when the customer demand is known to be low, such as the early hours of the morning [17]. The assumption is that this night time demand will be dominated by leakage. Since the leakage is constant (with a dependence on pressure, which is kept constant within a DMA by pressure-reducing valves), this estimation can be made more accurate by taking many readings and discarding all but the minimum over the low-utilisation period to account for the possibility of occasional but genuine consumer demand. Currently, the water industry performs leakage studies on an ad-hoc basis when excessive leakage is suspected or reported in a particular area [9]. A system that could quickly perform these calculations to effect a continuous leakage audit would deliver a clear benefit in terms of increased automation as the need to manually collect and analyse the datasets would be eliminated.

Many of the techniques in the literature cite specific algorithmic developments that can be used to detect leak conditions; or predict them based on historic information [18, 19]. However, these techniques are rarely implemented as production-ready algorithms, and the analysis is hindered by the complexity of querying large datasets. Where implementations are provided at all, these are typically in domain-specific modelling languages, or untested prototypes. In addition, in order that these algorithms could be used in the context of a distributed database they would likely have to be re-written, requiring significant analysis and understanding of the domain mathematics. Such an undertaking is clearly infeasible in the general case, would only yield a solution applicable for a small range of problems, and would also quickly become outdated as new techniques and modelling approaches became available.

1.1 Power for Sensor Nodes

The installation of a sensor node on a utility water pipe is underground, and typically in a remote location away from utility power sources. To illustrate, Figure 1.2 shows a logger attached to a pressure-reducing valve on an underground 8-inch water main.



Figure 1.2: Manhole access to an underground pressure-reducing valve chamber.

Although the power requirements for a small logger are low, typically <1 W [20], there is a limited number of options for providing this power in the utility water network. Batteries are currently used for short-term studies, although the cost of the required continuous replacement programme means that they cannot be used for a blanket monitoring installation. Many of the conventional sources of harvested power (solar, direct vibration, thermal gradient) are unavailable, and therefore the remaining option is to harvest energy from the flow of water. There are two distinct cases:

- In locations where a pressure differential exists, such as across a pressure-reducing or static valve, a small turbine could be installed in a bypass.
- In a straight pipe, which has minimal pressure differential, an insertion turbine can be used.

In this thesis, the concept of using an energy harvesting device powered by flow-induced vibrations with piezoelectric transduction is identified as the most suitable potential source of power. However, the main drawback with piezoelectric materials when used as transducers is the limited damping forces that can be achieved [21]. The power output of practical harvesters based on piezoelectric transduction at the cm scale is in the low milliwatt range, and the low damping is a significant limiting factor to a higher power output.

A piezoelectric device is often modelled as a current source with a small shunt capacitance, and

therefore in order to increase the force, and hence damping, the voltage on the capacitance can be modified by power processing circuitry connected to the piezoelectric device. Various power processing circuits have been proposed that have the effect of synchronously modifying the voltage profile of this capacitance, such as a circuit by Shenck in [22], which discharges the capacitance at voltage maxima rather than allowing it to drain throughout the cycle into a resistive load. A class of circuits called SSH (Synchronous Switched Harvesting), which have their origins in structural damping applications, improves upon the circuit developed by Shenck further by resonantly flipping the voltage, thus leading to a steady-state operation with a higher voltage magnitude [23].

This thesis compares the state-of-art power extraction circuits, and presents a new circuit, called Pre-Biasing, which improves upon the power output increase given by the SSH circuits by controlling the voltage profile on the piezoelectric capacitance.

1.2 Research Objectives

The main research objectives that will be addressed in this thesis are:

1. Large time-series datasets
 - (a) Establish the specific data requirements of an infrastructure monitoring programme on utility water pipelines, in terms of predicted data size and type.
 - (b) Review the specific application-level requirements of a data processing system for researchers working with the water industry.
 - (c) Review the existing database technologies for general-purpose and time-series data to establish which systems or components are able to fulfil the objectives.
 - (d) Design and implement a time-series database tool to allow researchers working within the water utility network to reliably store the large volumes of data that would be generated from a large scale infrastructure monitoring programme.
 - (e) To select a restricted set of query operators via a simple mathematical query language to enable scientists to efficiently search for data of interest, and to heavily optimise the implementation of these to make searching of large datasets efficient.

- (f) To provide a means for scientists to write custom applications that transform and filter the large dataset, without having to learn about the complexities of distributed systems, or *MapReduce* architectures.

2. Energy Harvesting - piezoelectric interface circuits

- (a) Review potential sources of energy from a water distribution main to support continuous-operation of a sensor node.
- (b) Establish the damping force that can be achieved with a piezoelectric energy harvesting device when used in a water pipe like that used by the utility water network.
- (c) Investigate the limiting factor of power generation with an energy harvester based on flow-induced vibrations, with piezoelectric transduction.
- (d) Review the existing power extraction circuits that aim to increase the damping from a piezoelectric energy harvester and assess their power output within the same analytical framework.
- (e) Design and test a power extraction circuit that aims to increase power output from a piezoelectric energy harvester by increasing the damping force that can be achieved via a synchronous modification of the voltage profile on the piezoelectric capacitance.

Chapter 2

Background

2.1 Experimental Programme

An experimental data-gathering programme was conducted by the *InfraSense* lab at Imperial College London across a variety of water utility locations between 2008 and 2012 to capture pressure and flow data with a view to gaining a better understanding of the transient state of the network. The locations included pumping stations, large agricultural consumers and remote distribution main sites across the South of England.

To capture the data, sensor nodes were designed based on industry-standard PC-104 hardware (named *InfraSense^{HP}*), with 8-channel, 12-bit A-to-D converters. These were configured to sample pressure at 200 samples per second from pressure transducers connected to the supply pipes. In the pumping stations, inlet and outlet pressure was taken. Data was needed for centralised processing, to compare the readings from a number of sites and perform real-time analysis, so a simple publish/subscribe mechanism was designed to control the sample rate and operational status of the sensor nodes. In addition, GPS was used to provide an initial position fix and time synchronisation for the recorded values. Since power failure is known to cause dynamics of particular interest at a pumping station, uninterruptible battery-backed power supplies were used that were able to transparently transition the load onto backup power. Figure 2.1 shows the installation of a sensor device on a 3 MW pump sampling the pressure of the outlet at 200 samples per second.



Figure 2.1: *InfraSense^{HP}* sensor node showed affixed to 3 MW mains distribution pump in a London pumping station.

The data was streamed in near-real time to a server over 3G cellular modems to one of two data capture servers, which then stored each sample in a *MySQL 5* database server, along with the timestamp, and the unique sensor and channel IDs. A simple web-based device management tool was written to allow the user to quickly interrogate the real-time data, browse historical data and download subsets of the data for local processing. The nodes published data to the data capture servers via HTTP, since that was the only protocol that could be guaranteed to be accessible using the 3G modems. Figure 2.2 gives an overview of the operational data gathering system.

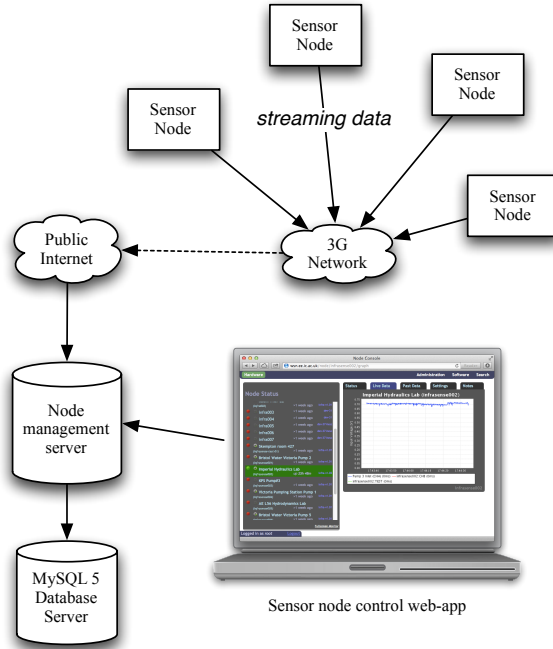


Figure 2.2: Overview of deployed system showing main hardware components.

Figure 2.3 shows a graph of flow measurements taken from the inlet to a large industrial user. The graph shows a predictable daily demand pattern which rises sharply during the morning, and falls to near-zero at night. The offline processing of such datasets can identify sudden changes in the behaviour which might be indicative of failure (short-term analysis), but would also be used for investigating the diurnal pattern in order to predict both short-term and long-term behaviour.

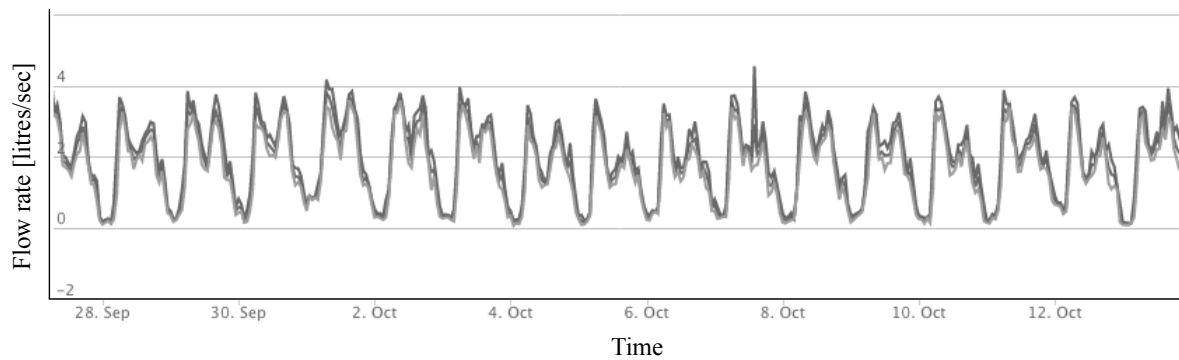


Figure 2.3: The daily demand fluctuation of a farm in the South of England, recorded in 2011.

Figure 2.4 shows a day with an abnormal operation cycle (of unknown cause¹) when the flow peaked at over 20 litres per second, 5 times the usual daily peak for this site. As discussed earlier this rapid change in flow rate would have placed considerable dynamic stress on the pipe infrastructure. In the 18 months of the dataset there are multiple similar events, showing the need for rigid data management tools which can be used to identify a typical diurnal profile versus abnormal behaviour. That would facilitate the design of improved threshold alert mechanisms, the prediction of demand behaviour for near-real-time monitoring taking into consideration different seasonal variations, and the diagnosis of failures and unusual demand profiles.

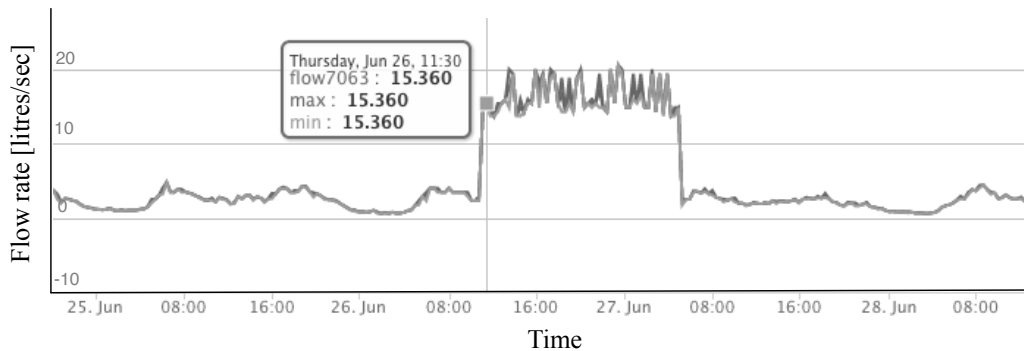


Figure 2.4: Abnormal flow rate showing large rate of change recorded on June 26th 2011.

2.2 Data Requirements

It was found that once the historical dataset reached a certain size, typically around one hundred megabytes, the time taken to perform conceptually simple operations became unacceptable. Since the entire dataset could not be loaded into most other tools, regions of interest had to be first located manually and then exported. The main limitations encountered were:

- Down-sampled views: Plotting a graph of contiguous values was efficient, but graphs spanning days and weeks required many millions of database rows to be read and aggregated, which became time consuming as the dataset grew.

¹The cause of these fluctuations is unknown, but it is speculated that they might result from the deep clean of a large water storage tank, with the large flow caused by the subsequent re-fill.

- Feature extraction: The task of filtering the dataset for candidate ranges of interest by simple searches (e.g. maxima/minima, rate-of-change, amount of time a signal was above threshold, value distribution) involved one or more lengthy scans through the entire dataset.
- Query specification: Expression of the problem using the SQL language was cumbersome and in many cases convoluted queries had to be constructed to express simple concepts.
- Calibration: Typically, the values recorded by the sensor node were the raw value from the analogue-to-digital converter. These values need to be converted into domain units (typically bar, or litres/minute) for analysis. All the queries on the recorded datasets must be performed on calibrated data, and it was found that expressing these calibrations in SQL queries when searching for data of interest led to awkward, large queries, and affected performance further as the queries were unable to take advantage of the indexes once calibrations were expressed in the query.
- While query operations were executing, which often took upwards of an hour, the database server became less responsive and dropped some incoming data.

A large-scale research deployment of sensor nodes, supplied with sufficient power, would generate a dataset which is several orders of magnitude larger than data generated from existing systems. The existing analysis methodology at water utilities relies on ad-hoc analytics with general-purpose tools such as *Excel* and *MATLAB*, which are unsuitable for datasets larger than can fit in memory, and start to perform poorly well below this limit. A limited number of queries are processed by state-of-art systems such as *Netbase*² due to the difficulty dealing with datasets of this size. As the system is scaled up and deployed across multiple sites, the combination of sampling rate and continuous data acquisition across a large number of distributed nodes will present a significant challenge in terms of volume of data and complexity of analysis and processing.

Systems that perform localised leak detection implement bespoke algorithms from high-resolution sampled data to determine the condition of a pipe [18]. This class of study is typically performed on an ad-hoc basis, and decisions can therefore only be made based on information gathered locally. However, there is an additional class of analysis that requires data from several locations, which may

²<http://www.crowderconsult.com/netbase-water-management-software/>

span large distances such as the size of a DMA, mandating the need for some form of communication system. Generalised leak estimation and localisation requires the data from a number of locations within an area [9], or within the entire network, to be aggregated in real-time to form a picture of the locations with the highest leaks. Although the data for this can be of a low resolution, the latency must be low enough so that transient behaviour is not misrepresented.

In addition to real-time aggregation requirements, scientists analysing specific burst or failure events need a significant history of high-resolution data to determine the cause. Bursts can occur at unpredictable locations, and in such cases ad-hoc sensor installations with local storage cannot be relied upon to provide the required coverage. Significant history of the pump data must be kept to track pump behaviour and mechanical degradation over time, using analysis of the frequency content of the signal [24]. In addition, until a better understanding has been reached of all of the circumstances that can lead to failure, the precise derived metrics that are needed are somewhat unknown, and a data gathering system that discarded raw data in favour of down-sampled views or other aggregations would risk making some future studies impossible.

The sample rates that are anticipated will become adopted in the water industry to manage the dynamic hydraulic conditions will be in the order of 1-200 samples/second. As the system is scaled up and deployed across multiple sites, the combination of sampling rate and continuous data acquisition across a large number of distributed nodes presents a significant challenge in terms of volume of data and complexity of processing.

2.3 Time-series data

The stream of data gathered from the programme is a time-series dataset. Each datapoint is a tuple of a timestamp and a floating-point value. The time taken to run an arbitrary analysis on a given dataset is limited by the speed at which that data can be read from disk. Computer memory is a few orders of magnitude faster than disks, and all but the most complex algorithms are therefore dependent on the speed at which the data can be fetched from the storage device by the processor. A high-quality magnetic hard drive can often read data faster than 100 MB/sec [25], but the read performance of hard drives compared to main memory is approaching 6 orders of magnitude, and

is widening still at a rate of around 50% per year [26]. This means that improvements in disk throughput alone cannot be relied upon to deliver the required increase in performance.

The monitoring programme outlined above was small compared to a full-scale study of a water network, covering only a small fraction of the total operational network. If the study were scaled up, and run for a longer period, the data requirements would grow further. To estimate this, Table 2.1 shows the storage requirements of several sampling regimes. Assuming the ideal case, where a database system is able to process data at the raw speed of the disk, then the time taken to process the entire dataset over a one year period will be the time taken to read this data, which is shown in the right-hand-side column. A conservative 50 MB/sec sustained read rate is assumed.

Table 2.1: Storage requirements of 16-bit sampling (with 64-bit integer timestamp), and time taken to read dataset from disk at 50 MB/sec.

Sample rate	1-year archive size	Read time at 50 MB/sec
10 Hz	2.9 GB	60 secs
50 Hz	14.7 GB	5 mins
200 Hz	58.7 GB	20 mins
1 kHz	294 GB	1 h 40 mins

For certain bulk operations, these query execution times might be sufficient. However, a system would only be able to perform one such query at a time at these speeds, and on magnetic storage devices the task of writing incoming data would reduce the read speed. In addition, queries that require the traversal of multiple datasets will take multiples of these data retrieval times. An efficient research tool is therefore needed, capable of storing and manipulating the large datasets that result from research programmes in the utility water work. In the next Chapter, the precise requirements of time-series database for application in the utility water network will be reviewed, and the implementation of a prototype research tool will be outlined.

2.4 Powering Sensor Nodes

The source of power for the sensor nodes in the ad-hoc sensing operation described in this Chapter was either a mains DC power supply, in locations where this was available, or 100 Ah lead-acid batteries in remote locations. In permanent installations, where mains power is rarely available, a continuous source of power is needed.

The lack of available power limits the current data acquisition systems in use by the water industry itself to roughly 15-minute sampling intervals, which fails to capture the dynamic behaviour. The sensors available broadly fall into two categories: those that store the data on some storage medium for later, manual retrieval by a site visit, and those that send data over some form of wireless link. The wireless link is most often the public GSM or 2/3G data network, as these have the greatest coverage, but due to their large power requirements these limit the length of time for which continuous data transmission can be achieved on battery power. The ability to increase the sampling rate to the range 1-200 samples/sec is therefore limited mainly by the available power, as the main component of the power usage of such sensors is wireless communication. Some loggers send measurements over SMS, although due to the high per-message cost and high, unpredictable latency, this is suitable only for very low frequency measurements, and is typically used in practice for 15-minute and longer measurement intervals.

Sensor nodes have recently been developed by A. Hoskins and I. Stoianov in the *InfraSense Lab*³ at Imperial College London to continuously acquire high-frequency, time-synchronised samples of the instantaneous pressure and flow. The node is capable of sampling at 200 samples/second for a period of eight weeks on a 3.6 V, 19 Ah battery, and saves the data to non-volatile flash memory which is then manually collected. A second generation of *InfraSense* low-power sensor node has been produced (Figure 2.5) which has a power consumption of less than 4 mW.

³<http://infrasense.co.uk/>

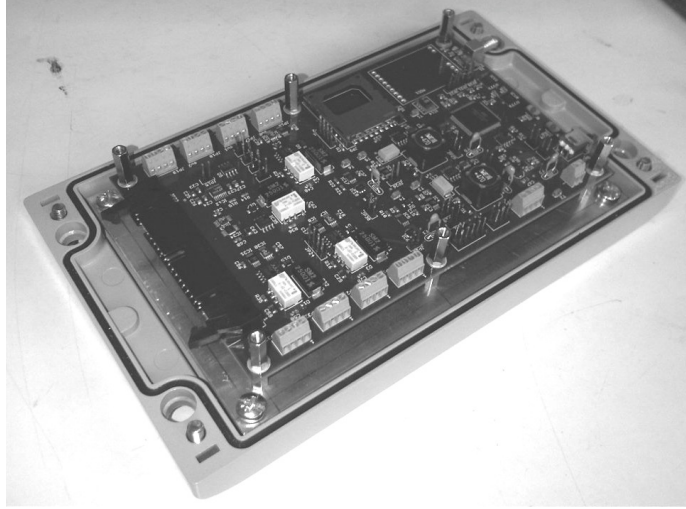


Figure 2.5: 2nd generation *InfraSense* low-power sensor node (A. Hoskins, personal communication).

However, even with this relatively low power demand, sensing deployment from battery power is limited to weeks rather than months, and continuous remote telemetry is not possible.

2.4.1 Energy Harvesting

The typical installation of a sensor node will either be in an underground access point like that shown in Figure 1.2 in the first Chapter, or buried underground with a complex joint or other equipment as shown in Figure 2.6.

The limited number of energy sources available to a wireless sensor node have been extensively reviewed and compared before [27, 28], and the mostly underground water network represents an application where many of these options are unavailable. Given the lack of viable alternatives, ad-hoc sensing programmes (over weeks rather than years) often rely on battery technology. Lithium batteries are commonly used for communication and telemetry applications due to their relatively high energy density. However, they pose a significant safety and operational risk when used underground. It is undesirable to hold a large store of chemical energy in a manhole chamber that is frequently flooded due to the risk of water ingress causing explosion. Additionally, the required



Figure 2.6: Mains water access chamber showing installation of both ABB insertion flow meter and *InfraSense* data logger.

continuous replacement programme would be costly. Therefore, due to the limited total energy available from a battery, semi-permanent installations of low-power devices are forced to operate at a very low duty cycle with low-frequency sampling. To scale up the sensing programme to cover a large proportion of the network, the significant research challenge of how to power the sensors remains.

Insertion flow meters, which are commonly used for ad-hoc flow measurements in the water industry, are known to vibrate under certain flow conditions. Consider the following extract from the manual of the ABB *AquaProbe* electromagnetic flow meter shown in Figure 2.7 [29]:

All insertion flow sensor devices are susceptible to the vortex shedding effect that can cause severe vibration of the flow sensor, resulting in damage and/or measurement instability. [...] some installations may experience unwanted vibration resonance that may further limit the maximum velocity at which the flow sensor can be used.

This vibration caused by the flow of water represents a potential source of power that could be

used to power a small sensor node. In the next section, the concept of exploiting this flow-induced vibration effect for harvesting energy from the flow of water in distribution pipes will be investigated.

2.5 Energy from Fluid Flow

There are fundamentally three kinds of energy in an incompressible fluid flow: the potential energy of the mass of the water due to gravity, the stored energy in the pressure difference between the fluid and the atmosphere, and the kinetic energy associated with the inertia of its movement. The weight of a mass of water m has a potential energy of $E = mgh$ (g is the gravitational constant and h is the height relative to some other point). A uniformly moving mass of water m has a kinetic energy $E_k = \frac{1}{2}mv^2$ (where v is the velocity of the water in meters per second). Bernoulli's principle states that these three forms of energy are conserved in a closed system (assuming negligible viscous losses). Different generator designs are required to extract energy of each type. The potential energy of a volume of water is exploited by a change in height. This technique is used in reversible hydro-electric power stations, such as *Dinorwig Power Station*, where water is stored at a height for periods of high demand, and then the change in potential energy is used to drive a turbine [30].

In closed-channel flow the fluid is constrained and fills the conduit completely, in which case its flow is driven by a pressure difference between two points. Open-channel flow, where either the fluid does not fill the conduit or the conduit is open on one side, is typically driven by gravity. In a closed-channel flow, such as a service pipe, any obstruction to the fluid flow creates a pressure differential. In open-channel flow the kinetic energy of the fluid flow can be exploited with an appropriate turbine. Modern wind turbines convert the kinetic energy of the wind into electrical energy with an alternator and a power-electronic interface to match the frequency to that of the grid. In some situations the distinction between a device extracting potential energy and one extracting kinetic energy can be blurred; for example, as water enters free-fall, potential energy is progressively exchanged for kinetic energy.

In an ideal open-channel fluid flow, with a channel of infinite diameter and a uniform fluid velocity profile, the maximum power that can be extracted from the fluid is given by:

$$P = \frac{1}{2}\rho AV^3 \quad (2.1)$$

where ρ is the density of the fluid, A the profile area of the turbine in the flow and V the flow velocity. This is limited further by the Betz coefficient (≈ 0.59), which is the highest efficiency that can be reached by an optimal generator. If the cross-sectional area of the turbine is much smaller than that of the channel then the channel can be approximated as open, but in this case frictional forces cause a radial velocity profile, with reduced velocity near the perimeter of the pipe.

Typical propeller-based turbines have a significantly lower-than-optimal efficiency. An efficiency of less than 15% can be achieved [31], with non-laminar flow and position-dependent flow rates lowering output further [32]. Vertical-axis turbines such as Gorlov's Helical Turbine [33] have reported much higher efficiencies of up to 35%, with the tradeoff of a complex insertion and operational mechanism. These limitations have limited use in operational systems.

A drawback to a turbine-based approach is the risk of sediments and particles in the flow which might cause mechanical failure of turbines. For this reason the UK water industry is looking for alternative solutions that provide power and present a minimal operational risk. Based on discussions with three UK water utilities, there is a concern that non-homogenous devices with bearings and rotating elements are fragile, and cannot easily be retrieved if the mechanism fails. An alternative to devices mounted in the main bore of the pipe is to exploit a fixed or variable pressure differential via a bypass. Pressure-reducing valves (PRV) are already used extensively in the water industry to limit the service pressure to residential customers by automatically modulating the size of an orifice depending on the pressure. There are commercially available devices that are able to exploit the pressure differential around a PRV, such as that produced by CLA-VAL [34] (Figure 2.7), which can provide 390 mW at a 6 meter pressure head and a flow of 6.5 litres/minute. The drawback of this system is that it requires this substantial pressure differential, and can hence only be installed at PRVs that are expected to produce a pressure differential of this magnitude for at least part of the day. The unpredictable daily demand profile of locations that are served by PRVs means that a pressure differential is not continuously available. The applicability of such devices may be limited further since not all sites where monitoring is desired have PRVs already installed. Although the power output from a PRV bypass device is in excess of the requirement of a small sensor node, monitoring is also required at locations that do not have PRVs installed, such as straight distribution pipes which have negligible pressure loss.

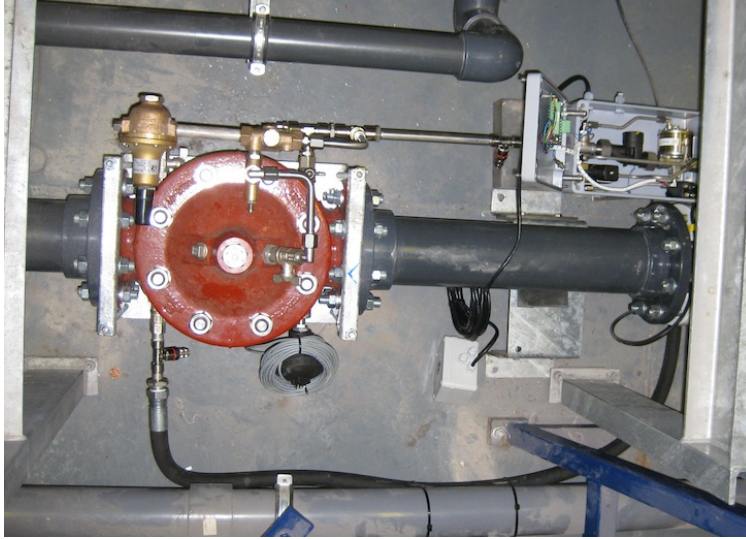


Figure 2.7: Pressure-reducing valve installation, with CLAVAL bypass turbine, in simulated recirculating flow test environment (Source: I. Stoianov, *InfraSense* lab)

There is limited information on the modes of direct vibration on the exterior wall of a water pipe, however an experimental study carried out by Cambridge University [35] shows that the fluid-pipe-soil interaction significantly dampens the various vibration modes to below useful levels. The power output of a vibration-driven energy harvested falls off sharply as the vibration acceleration decreases [36], meaning that a direct-vibration-driven energy harvester is probably only viable in locations where there is significant turbulence caused by sharp bends or turbulence caused by other significant variations in pressure.

2.5.1 Flow-induced Vibrations

When a solid body is placed in a fluid flow, under certain conditions an oscillating disturbance pattern is caused downstream of the body, as shown in Figure 2.8.

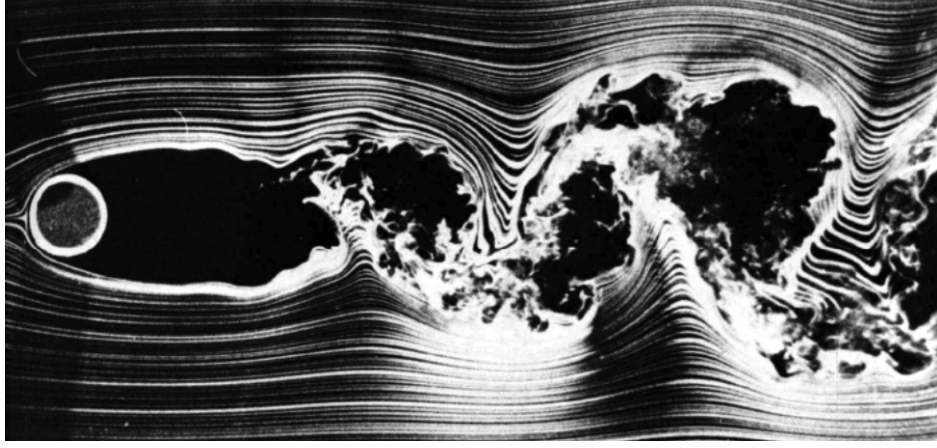


Figure 2.8: Flow-induced vibration phenomenon downstream of a bluff placed in a water flow. Flow direction is left-to-right. (Source: [37]).

The vibrations cause an oscillating force on the bluff body, principally in the direction transverse to (across) the flow. The characteristics of the fluid-induced vibration, and whether or not it occurs, is determined by the Reynolds number (Re), which is defined as:

$$Re = \frac{\rho U D}{\mu} \quad (2.2)$$

where ρ is the fluid density (water), U is the flow velocity, D is the dimension of the body (e.g the diameter of the cylinder placed in the flow), and μ the fluid's dynamic viscosity (in Pascal-seconds). At sufficiently high values of Re , as the fluid travels around the body in from point A to point S in Figure 2.9, viscous forces cause the fluid to lose momentum around its surface, forming a boundary layer around the body. This boundary layer then separates at point S , where it interacts with the higher-velocity fluid, and causes circular vortices (at point V). For very low Re numbers the viscous forces are proportionally larger than the viscous forces and flow separation does not occur.

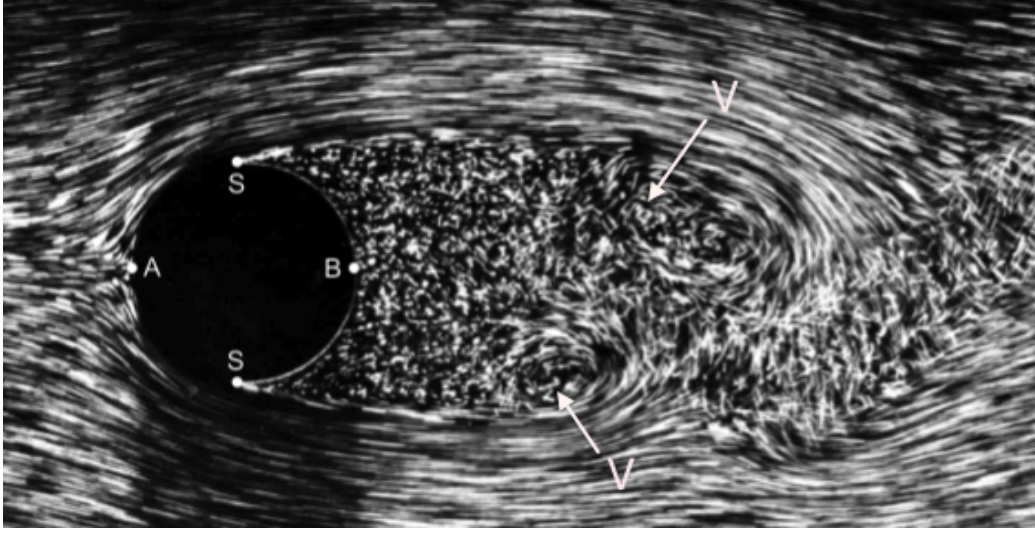


Figure 2.9: Separation of flow, from van Dyke [38]. A is the leading edge, S are points of separation, B is the trailing edge of low pressure, and V is the vortex formed by the interaction of the fluid with the boundary layer.

As the vortex grows in size, it draws fluid from the opposing boundary layer (caused by the separation at S on the opposing side of the bluff body) until it eventually redirects and absorbs the flow. This disrupts the flow to the primary vortex, which is then shed and travels downstream [39]. This travelling wake of vortexes is called a Kármán vortex street.

The amplitude of the oscillation caused by FIV can be up to 1.5-2 times the diameter of the body [40]. The frequency of oscillations that occur in a water flow of velocity v , with bluff diameter D , can be estimated by the Strouhal number:

$$St = \frac{f_s D}{v} \quad (2.3)$$

The Strouhal number is dimensionless, and is determined empirically from a given set of operating conditions and system dimensions.

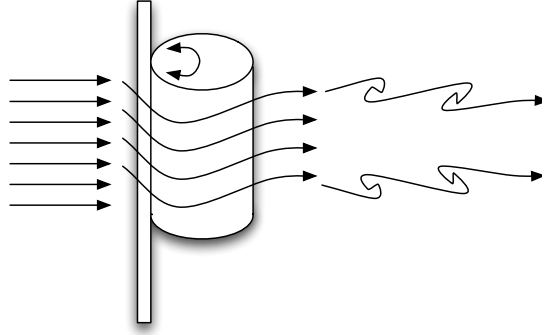


Figure 2.10: Bluff body vortex-shedding energy harvester.

2.5.2 Application

The concept of using fluid-induced vibrations to generate small amounts of power has previously been experimentally validated. In [41] (2002), T. Ghaoud and D.W. Clarke demonstrated a piezoelectric bimorph differential sensor that measures the flow velocity of the flow using the frequency of the resulting vibration. They demonstrate that since the frequency of the vortex oscillation is linearly proportional to the flow rate, the dominant oscillation frequency gives a good estimation of flow velocity. The authors estimate the principal frequency component by counting zero crossings under the assumption of a high signal-to-noise ratio.

In [42] L. Tang investigates the dynamics of piezoelectric cantilevers in axial flow. Modelling the plate as inextensible (the centreline of the plate remains un-stretched during the vibration) a three-dimensional model of the system is presented. One of the conclusions is that for a given system configuration there exists a critical flow velocity U_{Re} at which point the induced vibration takes place. Also of interest in designing a system for installation in a water pipe is the conclusion that when the fluid is confined between the solid, parallel walls of a pipe this did not cause an observed change in the vibration mode.

In US Patent 7224077 B2 [43] a system is demonstrated that uses a cylindrical bluff body placed at ninety degrees to the fluid flow (Figure 2.10). The patent is applied to large-scale energy generation, and uses an electromagnetic transduction machine, which as stated earlier limits the extent to which the device can be miniaturised.

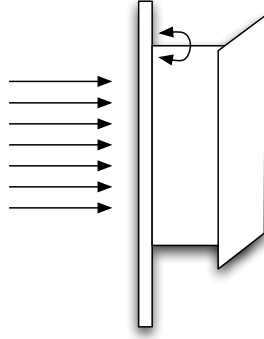


Figure 2.11: Bluff body vortex shedding energy harvester with flat plate.

The invention comprises this body connected to an axle which then derives power from with a traditional electromagnetic machine, and an impedance-matching circuit optimises the damping on the system for maximal power flow. The impedance matching circuit acts as an electrical spring whose force is proportional to displacement. The authors cite a previous invention of a similar nature (US Patent 6,424,079, the “Energy Harvesting Eel”)[44] that comprised a T-shaped object connected in a similar arrangement (Figure 2.11) with an electromagnetic machine to extract the power. The authors state:

An essential element of the idea of applying an electrical back force that is proportional to position is that this force is conservative. This means that the work that goes in equals the work that [goes out], i.e the introduction of the electrical spring can introduce resonance without a power reduction penalty.

In [45] an energy harvester is presented that generates energy from fluid-induced vibrations in a constrained flow. The authors pass a stream of water via a 8 mm diameter pipe over a piezoelectric element that is fixed at both ends inside a channel of 20 mm diameter. The flow rate is relatively high given the size of device (around 0.9 ms^{-1}), and an oscillation frequency of 26 Hz is obtained, leading to a power output of $0.2 \mu\text{W}$. The pressure differential needed to move water at this speed is high, and therefore unless deployed in a configuration where water were allowed to escape from a water main to atmospheric pressure the results obtained in this paper may not be achieved.

2.5.3 FIV in Utility Water Mains

To establish an estimated set of operating conditions for a large-diameter mains distribution pipeline, an experiment was conducted with an open-channel recirculating water flume at the Hydrodynamics Laboratory at the Department of Aeronautics, Imperial College London. The flume is configured with a honeycomb mesh to remove turbulence from the flow before it enters the main test section, which was 60 cm wide, 70 cm deep (variable), and 8.4 m long. This cross-sectional dimension is approximately equivalent to large-diameter distribution mains in use in the water industry. The flow speed is variable in increments of approximately 0.1 ms^{-1} , although the control system was open-loop.⁴

The experiment consisted of a circular tube of diameter 60 mm that was inserted into the flow from above, and mounted with a single degree of freedom in the transverse direction. The assembly was pneumatically suspended and thus had negligible friction, which was verified by measuring the damping of velocity with no water present in the flume. Figure 2.12 shows the assembly from the side.

⁴The experimental apparatus was borrowed and only available for a short time, and was principally installed for a different type of experiment by another user. It was therefore not possible to modify the experimental setup in any way without invalidating the other user's results.



Figure 2.12: Pneumatically mounted bluff assembly, with the direction of flow highlighted.

Initially a laser-based depth measurement sensor was used, but this was found to be susceptible to alignment issues and had to be frequently re-calibrated. Therefore an improvised position detection system was designed. An off-the-shelf USB webcam was used to capture images of a target image affixed to the moving assembly, which comprised of four dots of a distinct colour, as shown in Figure 2.13.

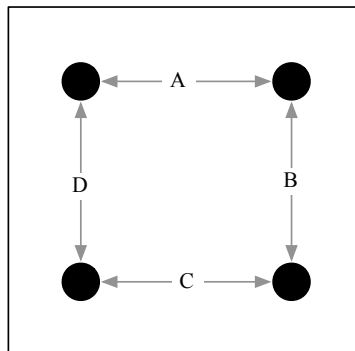


Figure 2.13: Printed target used to estimate in-plane displacement of bluff body holder assembly.

Software was written to capture an image from the camera and estimate the position of the centre

of each dot by counting the number of coloured pixels in each quadrant of the visible area, counting partially coloured edge pixels proportionally, and then averaging these locations (Figure 2.14). The lengths of the edges of the formed square were added to find the perimeter p . For calibration, two measurements were taken of the absolute position of the carriage (measured by a ruler) and the recorded value of p at either end of the extreme of motion. From these calibration points, and given the measured value of p , the position could be estimated using simple geometry. When compared to measurements taken with a ruler, the maximum displacement error using this technique was 4 mm, which occurred at the far extremity where the resolution of the camera limited the total number of pixels occupied by the coloured areas on the target image. The maximum rate at which this system would operate was 9.1 Hz, which was limited by the speed of the computer.

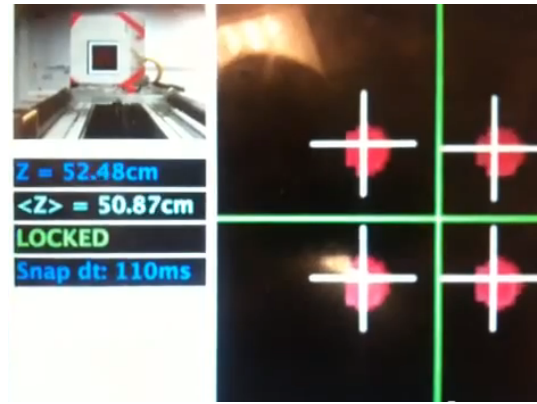


Figure 2.14: Interface of position detection software, showing raw feed from camera (top left), and position estimation of control dots shown by crosshairs (right).

A graph showing the oscillation of the bluff body at a steady flow rate of 0.25 m/s is shown in Figure 2.15.

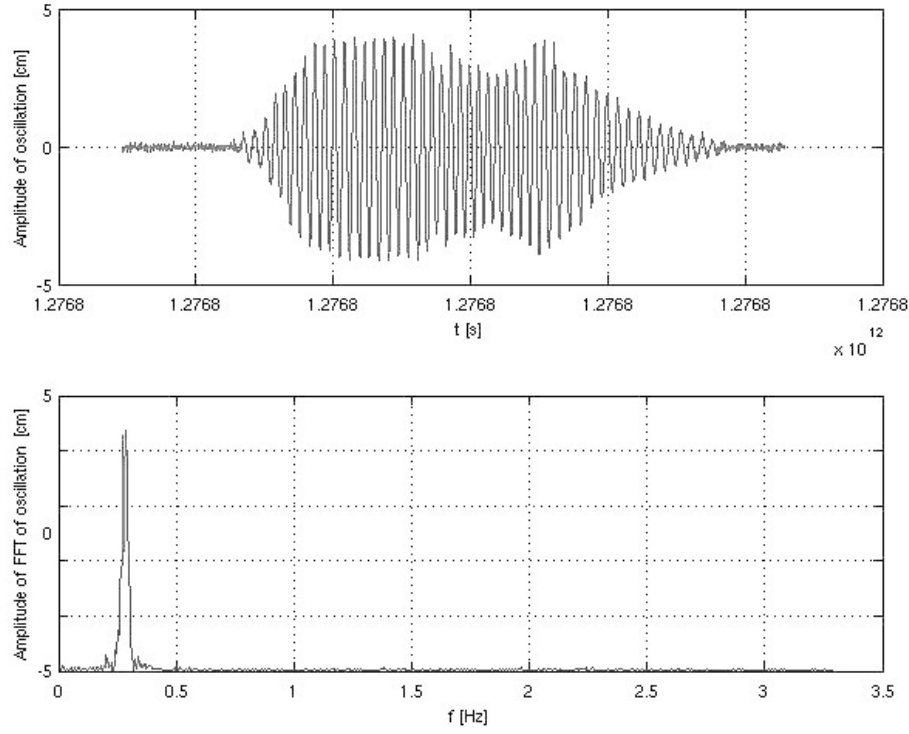


Figure 2.15: Experimental data showing oscillation of 6cm bluff body placed in open-channel water flume moving at 0.25 m/s (top), and the frequency content of this signal (bottom) showing a predominant frequency of 0.3 Hz.

It can be seen that the dominant frequency for this configuration is around 0.3 Hz.

The force acting on the bluff body can be approximated with the drag equation. The Morison equation [46] achieves a more accurate representation for the dynamic force acting on an oscillating body or flow, but in this case the mass of the bluff body used is not known, as it was not possible to remove it from the assembly. The drag equation is:

$$F_D = \frac{1}{2} \rho v^2 C_D A \quad (2.4)$$

where ρ is the mass density of the fluid (for water at 20°C this is 998.2 kg/m³), v is the velocity of the object in the fluid, A is the area of the body, and C_D is the object-dependent drag coefficient, which for a cylinder at low Reynolds numbers can be approximated as 0.5.

The area of the bluff body was the depth of the water multiplied by its diameter: $0.7 \times 0.06 = 0.042$ m. The peak velocity (from the results of Figure 2.15) was 0.12 m/s. With these values the result of expression (2.4) is a dynamic force acting on the body in the transverse direction of 0.12 N.

2.6 Piezoelectric Transduction

The best choice of transduction depends on the characteristics of the mechanical source of motion and the scale of the device [36]. Energy transduction systems using electromagnetic induction are typically rotating machines (as used in utility power generation) but less frequently oscillating coil-type machines are used, such as in the Seiko Kinetic watch [47]. The forces in an electromagnetic device typically scale between l^3 and l^4 , and electrical resistance scales inverse-proportionally to volume, whereas electrostatic forces scale much slower (between l and l^2). This means that there is an advantage to making electromagnetic machines very large. As the volume of the machine is scaled down the forces are much weaker, and at the <1 cm scale and below electromagnetic induction is a poor choice.

Piezoelectric transduction is sometimes referred to as a special case of electrostatic transduction due to the fact that electrostatic charges are accumulated on a capacitance. A piezoelectric material accumulates a charge on a material as a result of changing mechanical stress [48]. Figure 2.16 shows a common configuration of piezoelectric transducer - a piezoelectric bimorph strip, in this case modelled as a cantilever secured at one end.

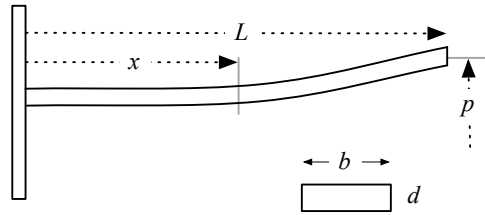


Figure 2.16: Diagram of a model of a piezoelectric cantilever.

An energy-harvesting application of vortex shedding using piezoelectric materials is presented by

Dongna Shen in [49] (also [50]). The author has developed a device aimed at the > 100 Hz frequency range that can produce $259 \mu\text{W}$ at 3.1 V which, given the volume of the device, equates to 4.6 mW/cm^3 . The authors demonstrate the device operating with an optimal resistive load of $75 \text{ k}\Omega$. Throughout the work there is reference to the fragility of the PZT (lead zirconate titanate) and that this was a significant problem due to the high amplitude of the vibration effected upon the device. To overcome this limitation other materials were used, such as MFC and PVDF, which offer higher material strength but lower power output.

The theoretical energy yield of a piezoelectric strip will now be estimated.

The piezoelectric beam acts like a capacitor, with stored energy $0.5Q^2/C$. The energy per deflection cycle of the beam is [48]:

$$E = \frac{108000}{\epsilon_o \epsilon_r} \frac{1}{L^3} \left[\frac{d_{31} E_f P}{E} \right]^2 \quad (2.5)$$

Let us assume the following values, which are typical for a PZT material:

$$\begin{aligned} d_{31} &= 370 \times 10^{-12} \text{ C/N} \\ E_f &= 60 \text{ GPa} \\ E &= 100 \text{ GPa} \\ \epsilon_0 &= 8.85 \times 10^{-12} \\ \epsilon_r &= 12 \end{aligned}$$

We shall also assume that the length of the beam is twice the length of the area of perturbation, that the beam thickness is $1/20$ of the length and that the beam height is $1/3$ the length. Therefore, by substitution into (2.5), the energy per flex of the beam is:

$$E = 5 \times 10^{-5} \frac{P^2}{L^3}$$

Given this relation some typical values can be calculated for the theoretical energy yield from a device that can be modelled as a piezoelectric cantilever. In water, a device 5 mm long under an acceleration of 3 cm/sec^2 can develop 0.032 mJ per cycle. At a vibration frequency of 10 Hz this is a power output of 0.32 mW .

2.6.1 Piezoelectric Electromechanical Damping

The limiting factor to the power output of an energy harvester utilising piezoelectric transduction is often the electromechanical damping forces that can be achieved. For a resonant device, the electrical damping should be set equal to the parasitic damping in the system, or to a level where the device is just short of hitting the end stops, whichever is greater [36]. However, piezoelectric materials typically have low electro-mechanical coupling coefficients [21], meaning practical devices often operate at below the optimal value [51].

The dynamic force acting on the oscillating bluff body encountered during the recirculating flume experiments was around 0.122 N. This is approximately the same, and in some cases slightly higher than, the blocked force achievable from piezoelectric actuators, such as the range offered by *Piezo Systems Inc.* [52]. The typical quoted maximum blocked force available for the actuator devices in their product range 0.1-0.4 N.

As soon as energy is extracted from the piezo, this blocked force will drop compared to what is achievable in open-circuit operation. These particular devices are designed for actuation rather than harvesting, and therefore the figures quoted are the blocked force values at the full rated voltage. When driven in reverse, the devices will rarely operate at this maximal voltage. In addition, the presence of rectification will limit this voltage further since the optimal voltage for a bridge-rectified output is considerably lower than the peak open-circuit voltage. In a practical harvester, given that rectification is required, it is appropriate to consider circuits which achieve a DC output while also providing active damping. Therefore it is concluded that the limiting case to the power output of a piezoelectric harvester acting as a transducer for a harvester using flow-induced vibrations will in many cases be the achievable piezoelectric damping force.

It is expected that for the design of a practical device, the following difficulties will be encountered:

- The low electro-mechanical coupling available from piezoelectric materials will limit the amount of power than can be generated from a device utilising flow-induced vibrations from water flow. This could possibly be addressed by mechanically coupling multiple piezoelectric devices, but such a configuration would require good phase alignment between the devices if they are to be connected electrically in parallel also.

- The frequencies observed during the flume experiments were in the 0-5 Hz range for the 6 cm device that was tested. A larger piezoelectric device needed to take advantage of the high-amplitude oscillations will have a larger capacitance, meaning the resonant charging circuit would require larger components (particularly the inductor) for efficient operation. Smaller devices were not investigated practically, but others have demonstrated working systems at the cm scale and below [45, 49].

2.7 Conclusion

In this chapter, two crucial research challenges have been identified, which currently limit the ability of a water utility to deploy a wide-scale continuous infrastructure monitoring programme:

1. Once data is captured, query execution using the existing tools infrastructure was too slow for researchers to make timely analysis and predictions based on the data. As the datasets grew, the length of time taken to process queries meant that practical interrogation of the dataset was infeasible. Therefore an efficient research tool is needed to enable the large datasets to be stored, filtered, processed and transformed for input into other specific research tools.
2. The lack of an available continuous source of power prevents effective deployment of sensing technology. Fluid-induced vibration from the mains water flow with piezoelectric transduction has been identified as a potential source of energy, but the damping forces available with piezoelectric materials is well below the optimal value in most applications, and the potential power output of such a configuration is low.

These challenges will be addressed in the following Chapters.

Chapter 3

Development and Implementation of a Distributed Query Architecture for Water Network Research

The task of the researcher who wishes to capture time-series data from sensor nodes deployed on civil engineering infrastructure is complicated by the size of the datasets that are accumulated, and hence the time taken to run simple operations on them. As outlined in Chapter 1, a year’s timestamped data from a single sensor at 200 Hz requires 60 GB of storage space, and to form an accurate hydraulic picture of a large system many such sensors would be used. Traditional analysis methodology has involved short-term sensing programmes and often using software tools that are designed for interrogation of small datasets such as *MATLAB*, *Excel*, and custom scripts involving bespoke flat file data formats. The limits of these tools is quickly reached as the dataset size increases, and represents a major bottleneck to extracting knowledge.

3.1 Requirements

Before the requirements of a system some bounds on the operational requirements must be placed. Locating a feature of interest in a dataset with a degree of certainty can be a computationally complex task, requiring advanced domain knowledge and bespoke algorithms. In Chapter 1 the most common causes of pipe failures were identified, and it was observed that a few key metrics can assist in the study and detection of future leaks.

These metrics were as follows:

- Minima and maxima of a range of readings over an appropriate time range is an effective estimation of the proportion of leakage within a metered area; time-scoped range queries are useful,
- The pipe infrastructure in use has clear upper limits in terms of static pressure defined by the materials used and so value-time integral is an effective predictor of future failure (i.e. sum over a given time range),
- Transient pressure (i.e. rate of change of pressure with time) is identified as another major contributing cause of premature failure, so rate-of-change calculations and the ability to locate data based on a rate-of-change predicate query would enable detection of interesting transient pressure events,

- Leakage in an area can be tracked over time by calculating the sum of a collection of flows that should sum to zero, thereby necessitating the ability to make calculations based on data from multiple streams simultaneously. Therefore simple aggregations of values over custom time ranges is needed.
- Infrastructure sizing requires access to historical distribution of data, both over all time and over arbitrary sub-ranges.
- Detection algorithms for smaller-scale leaks require custom algorithms based on frequency content, so a method of execute arbitrary functions on the data would enable this and other future developments to be implemented.

The requirements outline above are algorithmically simple, but are complicated by the fact that they all need to be applied to very large historical data streams; when applied to large datasets their runtime is frequently unacceptable. An optimised approach is needed that minimises disk seeks and reads, but still provides the same level of functionality without errors in results.

The anticipated magnitude of the dataset that must be managed places some additional requirements upon a system in terms of performance and system reliability. In Table 2.1, the size of a dataset from a real-time logger operating at 200 Hz was estimated at 58.7 GB per year if the data is sampled at a 16-bit resolution. If a conservative deployment of 100 nodes within the area of a single water utility were undertaken, over a monitoring programme lasting 5 years, then the dataset would be 29.4 TB.

To store this data, a researcher could build a large storage array by adding disk drives to a single system until the desired capacity is reached. As of September 2012, to the best of the author's knowledge the largest commercially available hard drive is the Seagate 4TB *GoFlex* drive¹. While multiple such drives can be connected to a system to form a single large storage array, the available bandwidth for data transfer and data access in a single system would soon become a limiting factor in such a system. As storage is added, the amount of time needed to perform queries would grow. The read performance of hard drives compared to main memory is approaching 6 orders of magnitude, and is widening still at a rate of around 50% per year [26]. At 100 MB/sec it would

¹Source: product literature at <http://www.seagate.com>

take nearly 12 hours to read an entire 4 TB drive. In such applications, the demand must be spread amongst many individual servers in order to achieve the desired aggregate bandwidth.

When a cluster of servers is comprised of a number of servers which are individually prone to failure, the failure rate increases faster than linearly. If any individual server failure is disastrous, then the chance of failure of the cluster as a whole (R) is:

$$R = 1 - (1 - r)^N \quad (3.1)$$

where N is the number of servers in the cluster, and r the chance of failure of an individual server (the failure modes of the individual servers is assumed to be independent².)

This expression predicts that in a given period, taking the two-year hard drive failure rate observed by Google of 10%, an 8-node cluster has an unacceptably high chance of an instance of data loss occurring of 56%. A 100-node cluster has an almost negligible (0.003%) chance of being operational after two years.

The most effective way to mitigate the high chance of failure is not to invest in more reliable components, as this has a diminishing return in terms of investment, but to architect a system that, by design, can tolerate a certain number of individual failures. Scalable distributed systems mediate the chance of overall failure by keeping multiple, redundant copies of data. Expression (3.1) can therefore be modified to take into account of the level of redundancy by modelling the chance of M failures happening in a given time period, each of probability r , as simply r^M :

$$R = 1 - (1 - r^M)^N \quad (3.2)$$

Considering the previous example again, but this time with a redundancy of 2 (two copies of each piece of data), the probability of failure is reduced considerably from 57% to 8%. At a replication level $M = 3$ the chance of failure of the same system is just 0.79%. In practice, failed components would be replaced nearly immediately by a data centre technician, dramatically reducing window of exposure, and hence the overall risk.

²In practice the failure modes of the servers would probably have slight spatial and temporal correlations, as servers are typically housed in racks, which can be subject to failures, and servers acquired at the same time may suffer the same batch manufacturing defects.

It is clear from this analysis that as data volumes grow, the demand will exceed the storage and processing capacity of a single server. Merely adding more servers is insufficient to increase the capacity of a system; the software architecture must be designed with scalability as one of the primary considerations.

3.1.1 Query Execution

It is convenient to provide a simple textual interface to a database system to allow human-readable queries to be executed. An examples the SQL language for querying relational databases [53], which abstracts the task of searching for records from the user by providing only high-level constructs such as predicates and joins.

There is an additional advantage to providing a query language when optimising the execution of a program. To find the maximum pressure over the entire dataset, an engineer might write the following imperative program:

```
max = 0
foreach (value in dataset) {
    if (value > max) {
        max = value
    }
}
return max
```

At the end of the execution of this program, the variable `max` will be set to the maximum value of any of the samples recorded in the dataset. This approach has a drawback in that its runtime is linearly proportional to the size of the dataset. Since here the underlying user intention is unclear, in the general case it is not possible for a database system to analyse and optimise such a program.

Instead of requiring that the user write an imperative program to locate data, which would require users to study database internals, the database abstracts the task of retrieving data and can be highly optimised. In a domain-specific query language the above query could be eliminated by the

system providing a highly optimised `max` operator, and express simply as:

```
max(dataset)
```

The implementation is undefined, and could even be the same as in the first listing, but the listing with the `max()` statement provides the opportunity for a domain-specific implementation in a way that the first implementation does not. The semantics of the operation are now available to, and indeed defined by, the system, and the size of the result is clear, in that the `max()` operator returns only a single value. A whole class of optimisations are now available to the system, including pre-computation of the value (indexing) and elimination of this query if it is a superfluous or repeated part of a larger query.

In this small example, of a system that provides just the `max` operator to the user, the above query could be transformed from one where computation time is proportional to the dataset size, $O(N)$, to a constant one, $O(1)$, by maintaining a separate reference to the maximum value as each data point arrives. Then, execution of the query is simply a matter of reading and returning this single value, and the dataset need not be read in full. In general the implementation is more complex; but by providing high-level algorithmic primitives to the user, optimisation is less challenging.

3.2 Existing Systems for Large Time-Series Datasets

The requirements on a system outlined will now be compared against the capabilities of existing database systems.

3.2.1 SQL Databases

Since SQL databases represent the core component of most server applications and have existed in some form or another for several decades, a large number of implementations exist, both open-source and commercial, each with subtly different tradeoffs in terms of performance, features and scalability. The application domain is a time-series database, with a single key (timestamp) and a single value (the measurement) representing each record. A simple time-series database could

in theory be implemented upon a large number of the existing databases, but as will be shown a sufficiently performant solution may be difficult to realise. Some common single-node RDBMS systems will be reviewed first, and their suitability for a high-throughput historical time-series database will be evaluated. Then more complex architectures involving distributed systems, and domain-specific databases will be considered.

MySQL is perhaps the best known general-purpose relational database system. Market research estimates the share of *MySQL* at somewhere between 25% [54] and 30% [55], which makes it the most popular open source RDBMS. It supports most of the ANSI SQL:99 query dialect [53], and stores data in databases, tables and rows. Transactions of an arbitrary scope are supported, and the databases supports row-level and table-level locking. There is a large number SQL-compliant RDBMS systems (such as *Oracle*, *PostgreSQL*, *Sybase*, *SQLite*, etc), and much of the discussion in this section generalises to these.

In order to store time-series sample data in an RDBMS a table schema like that in Figure 3.1 could be used. Here, the samples for all sensor nodes are stored together and differentiated by a unique ID assigned to each one.


source_id (BIGINT)	timestamp (BIGINT)	value (DOUBLE)
12	1349363300100	8.011
12	1349363300200	8.1061
12	1349363300300	8.0013
12	1349363300400	7.9941
12	1349363300500	8.00966
		

Figure 3.1: Example *MySQL* schema for time-series data storage.

This table schema would exhibit unacceptably slow (linear) read performance since most queries in this application domain would require a scan of the entire table, or a large part thereof. To speed up data access to a particular row based on time or value, indexes could be added to one or more columns. *MySQL* supports three basic types of indexes: unique (where each value must only

occur once), non-unique, and full-text for indexing text fields. To allow retrieval by a query using any of the columns defined above, an index should be added to `source_id`, `timestamp` and `value` such that queries searching for a particular time range or value can use the index to locate data of interest. The indexes are typically stored as trees (heaps on disk) and exhibit $O(\log N)$ complexity searching for a single key.

In order to determine the overhead associated with storing such indexes, a test was conducted with *MySQL* version 5.0.77, storing sample data collected from a utility water pipe, which took up 2599 MB in a table with no indexes³. When the three indexes defined above were added, an additional 2973 MB was used, which is just over a 100 % increase. The large size of the index is accounted for by the fact that *MySQL* has no information on the usage patterns of the index, and therefore cannot optimise its space requirements based on any assumptions about what aspects of the index will be needed by queries. It naively adds every individual sample value to the index in case it is required to satisfy a query searching for an exact match. In the case of time-ordered sample data, there will be very few duplicated timestamp values so the index degenerates to a copy of the entire dataset, stored in value order, rather than the primary-key or insertion order of the master table. There exists a few trivial optimisations that could have been performed by *MySQL* if it knew that this was sample data, such as only storing every 100th timestamp value, and then performing a linear scan between indexed values.

In terms of adaptability to the application, *MySQL* does not support any form of custom index. Automatically generated views can be defined with guaranteed consistency semantics, but these carry an additional runtime costs in terms of updating and persisting the changed view. *MySQL* is a general-purpose database, and is hence optimised for common usage patterns. While a time-series database could be implemented using *MySQL* as a storage layer, this layer of indirection might severely hurt performance.

Performance tests were conducted on commodity server hardware to establish the performance of *MySQL* when writing rows to disk. The server was able to sustain around 2.35 k row inserts per second, where a single row represents one sample. For this test, the default configuration of *MySQL* was used, which does not call `fsync()` after writing each row, meaning that there may be a delay

³The original data was up sampled to increase the size of the raw dataset by a factor of 10.

between the database acknowledging a write and the write actually being persisted. Using the batch-insert feature this rate was increased to around an average of 4.3 k writes per second. Often an inserted row in a relational database represents a user operation or transaction, in which case 4.3 k operations a second is sufficient for even moderately high-volume applications. In the case of high-frequency sample at 16-bit resolution, with a 64-bit timestamp, this equates to an equivalent raw write speed of only 42 KB/sec, which is between 2 and 4 orders of magnitude slower than commercially available hard disks are able to sustain writes.

A major requirement of the application outlined in the previous section is that the storage architecture and query execution are scalable to more than one server system. *MySQL* is principally a single-node database, although it does support master/slave replication. In order to spread storage and processing amongst several servers, application-level sharing is often used with individual stand-alone instances of *MySQL* storing partial sets of data. This approach introduces complexity in terms of locating data and recording its location, and without duplication is susceptible to data loss. If a sharded architecture were used, the significant application overhead of replicating data and balancing storage means that the database itself would be essentially an access layer for the underlying storage medium and it is therefore questionable whether any benefit would be derived from its use.

3.2.2 General Purpose NoSQL Databases

A number of scalable, distributed database architectures have emerged to fill the gap left by traditional RDBMS when it comes to vast data warehousing and querying applications.

Google's *BigTable* is a general-purpose distributed storage system for non-relational structured data, and was designed and is used exclusively internally by Google [56] for their applications, including Google search. It has been proven to scale to petabytes of data across thousands of nodes, and in all likelihood an instance of *BigTable* is the largest running database installation in use anywhere on Earth [57].

Records are indexed by row and column keys, which are strings, and each cell is individually versioned. A row can be atomically replaced, meaning clients are able to reason consistently about

the state of the system given many concurrent readers and writers. An optimisation offered by BigTable is that groups of rows with similar keys are grouped together into *tablets*, meaning a row scan over a small number of rows typically only requires access to either one, or a small number of different storage nodes.

The *MapReduce* functional programming paradigm was first used by Google in *BigTable*, and is used for many of their large processing tasks, such as building search indexes [58, 59]. The strength of *MapReduce* is that once an algorithm is decomposed into completely independent, trivially parallelisable constituents (*map* and *reduce*), the task can scale to an arbitrary number of nodes, where coordinating the distributed task is handled transparently. The *map* stage transforms every key-value pair into some other key-value pair, and then the *reduce* stage combines all values with the same key from the *map* stage to form the final result. Large throughput can be achieved for large jobs on many thousands of nodes. However, it is less suitable for incremental updates to existing calculations as it relies on operating on data in large batches for efficiency [60], and even Google is said to be moving away from the MapReduce model for this reason [61].

Although much of the design considerations and implementation details have been published, the specific implementation of *BigTable* itself is not publicly available. However, a number of other databases that are arguably inspired by *Google's* work support similar semantics.

The Apache *HBase* project is a distributed database aiming to support vast tables of data and is largely inspired by papers by Google on their *MapReduce* implementation. It offers a non-relational store (tables are independent, and no attempt is made to enforce integrity constraints) and is intended to run on the Hadoop distributed filesystem. The database is aimed primarily at scaling up large website data-stores with a high proportion of accesses being writes, such as Facebook and Twitter. It supports efficient running of large MapReduce jobs, which typically categorise web pages, or compress images. The intention of these jobs is that they will each be of a significant size, and that it is infeasible to run each job on a single computer system due its to time and space requirements.

HBase is a general-purpose distributed database, and as such there is little access to the underlying storage fundamentals unless custom code is written to interface with the HDFS filesystem directly. It is not trivial to manually move data between nodes (region servers), instead the underlying

HDFS storage system automatically replicates and distributes data in blocks. Data is stored on disk, and logically in ascending key order. It is therefore only possible to read data between two keys in that order, and no indexes are available. By careful key design, users are usually able to ensure that the data is distributed amongst servers to suit the application (either to spread the storage burden over the available servers, or to ensure that a row scan hits the smallest number of servers). A common pattern is to store a small hash of the entire key at the beginning of the key, which improves the uniformity of row distribution amongst region servers.

Data is stored in rows, each of which has a unique key and multiple columns. Each cell can contain an arbitrary amount of data, and columns are grouped physically into column families, which are always written and scanned together by the filesystem. The client interface to *HBase* makes no assumption or restriction on the data format that can be stored, values are saved and retrieved as arrays of bytes and must be serialised and deserialised by the user. *HBase* stores the full key with every column, both on disk and in the network-serialised form. This means for applications where the value in each column is probably small this causes a large additional overhead associated with storing data in terms of storage burden and network link utilisation. A test conducted with a 75 GB dataset and a minimal schema showed that the size on disk was around 341 GB, which is over 4.5 times larger, and this figure excludes the additional overhead due to replication.

Since no custom indexes are supported by *HBase*, and rows may only be read in ascending-key order, application designers are often forced to construct their own indexes. An index structure could be constructed manually by deriving a new derived key based on the field to be indexed, and storing the key of the primary record as the value with this derived key. In practice such indirection causes performance to suffer badly in a distributed system as records have to be accessed from many different systems and combined. In practice, this often means duplicating entire datasets and storing them in different key order to support iteration. Where many indexes are needed the size of the dataset can grow considerably, requiring more server hardware, backup equipment, energy and maintenance overhead.

The strength of *HBase*, in that it is designed to serve a wide-variety of applications, limits the scope of optimisation that can be performed in an application that has to store only one specific type of data. Lack of safe access to the low-level storage mechanism (in this case *Hadoop HDFS*) limits

the scope of optimisations that can be performed in terms of data locality, ordering and on-disk storage format.

HBase has the following limitations for this application:

- No assumption is made about the order of data, therefore typically each row is written one to an insertion-ordered log, and then asynchronously compacted to a sorted table some time later. All rows are therefore written to disk at least twice after insertion.
- Significant understanding of computer systems administration and some understanding of distributed systems is needed to install and run an *HBase* cluster. Therefore an application built using *HBase* as a storage layer must hide this complexity from a data scientist working on the utility water network.

MongoDB is a document-oriented database, similar to *HBase* in architecture, that stores JSON documents against string keys. Native support for a JSON-like schema-less serialisation format in terms of primary and secondary indexing and single-cell updates makes the architecture highly suitable for web applications in which JSON is a convenient format for delivery to a web browser. In contrast to many database systems that read data from disk and use a small in-memory cache to speed up frequently accessed records, *MongoDB* attempts to keep the entire dataset in memory. Tests showed that as the dataset approaches the limit of available memory, operating system page faults dramatically lowered the attainable read performance by about two orders of magnitude.

Writes are not flushed to disk until some later time which dramatically increases write performance (particularly in the case of benchmarks), at the expense of durability of the written data.

3.2.3 Infrastructure-as-a-Service

Infrastructure-as-a-Service, or informally *Cloud* computing, is a recent advance whereby users rent a fraction of the processing capacity of a much larger machine in a remote datacentre under a usage-based pricing structure. Largely made possible by advances in virtualisation technology, and the rapidly increasing hardware processing capacity, as of 2013, a server of modest specification⁴ can be

⁴1.7 GB RAM, single-core, 160 GB local storage, 64 bit architecture.

rented by the hour for around \$47 (USD) per month [62]. The ability to scale computing resources with demand is attractive for companies that do not wish to commit to expensive data centre operations or co-location of purchased server hardware before their needs are fully known. Established companies are able to use the flexibility offered by hourly pricing to meet the requirements of their peak demand, while the baseline load can be served from their existing infrastructure.

The drawback with IaaS systems is that they offer no advantage in terms of reducing the complexity of a problem to a manageable level, because it is not always possible to trivially distribute a computation amongst many computer systems. Some algorithms can be distributed and executed on many systems in parallel relatively easily, such as Monte-Carlo simulations, multiple runs of the same simulation with different initial conditions, and trivial transformation operations such as batch image filtering and analysis. However, in the general case it is not possible to distribute arbitrary computation, and the expertise needed to analyse and decompose complex algorithms for efficient parallel execution will often be out of scope of the work at hand. Therefore, utilising IaaS resources alone is not considered a solution to the storage and processing of a large dataset.

3.2.4 Offline Stream Processing

The databases and database technologies reviewed so far are general-purpose, in that they are designed to be suitable for a range of applications. They make no assumptions about the format or intention of the data they store, they merely provide an interface to reliably read what is written.

Stream processing systems such as Aurora [63], Borealis [64] and Medusa [65] aim to simplify and automate the task of running queries over a continuous stream of data as it arrives, and in general this is achieved by an automatic distribution of primitive operators amongst nodes in a network in order to put the processing as close to the source of data as possible.

As a trivial example of stream processing, consider a water network with two pipes, each with a pressure transducer. The operator wishes to know if the pressure in both of them exceeds some threshold at any instant, e.g.:

```
pressure1 > 8.5 AND pressure2 > 8.5
```

A naive system would collect the data from the two streams and then evaluate the two predicates in the above expression, presenting an alert should they both hold. An optimising stream processing system would transfer execution of the operators to as close the data source as possible in the stream network, usually to the data node itself. Each node would then only send samples that match its individual predicate.

SignalDB (MIT) is a stream processing system developed to allow operators to analyse acoustic and transient signals that was designed specifically for use in utility water pipelines. Queries can be composed of primitive operators and these are then executed on data streams as they arrive to determine the operational status of the infrastructure, and whether a leak has occurred. The system comes with primitive operators for common anomaly detection methods used in water networks, such as cross-correlation and power spectra analysis. It is restricted to continuous data, and it does not appear that the system is able to address the problem of optimising access and aggregation of stored data, or making decisions based on a combination of past data aggregation and continuously arriving data streams.

Stream processing systems perform optimisation primarily in terms of data locality, under the assumption that one of the greatest costs of operating a distributed system is communication overhead. While the application of stream-processing systems to real-time or near-real-time data in the utility water network could provide early warning notifications of events, such an application would not assist the water utility in capacity planning decisions which depend on past data such as resource sizing and evaluation of high-demand periods.

OpenTSDB is a distributed server metric database that uses *HBase* as a storage layer. It is designed to store operational metrics of server systems, such as processor utilisation and remaining disk space, for a large data centre, and the main purpose of the storage of data is for graphing, and instantaneous trends. The database is not optimised to run arbitrary aggregator queries over the dataset, rather, it is designed for the restricted subset of views that are required by a system administrator and as such pre-computes only some commonly required views in this domain (such as 7-day average graphs). In addition, the alerting mechanism uses an industry-specific infrastructure status dashboard tool (*Nagios*), and the capture interfaces are designed to record data from the available operating system counters. Custom queries could be implemented on *OpenTSDB* via

MapReduce jobs running on the underlying *HBase*, but such an undertaking would be convoluted and potentially fragile as requirements and query strategies changed.

Apache *Accumulo* is a distributed, sorted key/value store, inspired in part by *BigTable*, that is built on top of *Hadoop*. It specifically supports timestamped rows, which means it can be used to implement a time-series database. It also supports a kind of hierarchical indexing with a user-definable index range size. Indexing support is more powerful than *HBase*, as the database supports custom server-side indexers (which are provided by client code running on the database servers), and these are typically iterations over datasets that can be used to implement aggregations. Since all indexes must be built and defined by the user, in order to implement a time-series database with support for a set of aggregators upon *Accumulo* a significant amount of customisation and domain knowledge would be required.

KDB is a commercial implementation of a historical database aimed at rapid decisions on financial stock market data. It uses a query language called *q+* and claims to be able to run high-performance queries on real-time and historical data. Since *KDB* is a closed source commercial product, it is not possible to ascertain any details about its implementation.

Another commercial offering called *FAME* (Forecasting Analysis and Modelling Environment) is a time-series database, also aimed at the financial sector, that is designed for browser-based analytics applications to be built upon. The operators are primarily designed around pattern recognition, and much of the development effort appears to relate to making the system interoperable with other tools used in this sector (such as *MATLAB*).

Approximate answers to offline stream processing queries computed by the *Stanford Stream Data Manager* (*STREAM*) database system using a declarative SQL-like language [66]. The system performs an online optimisation of the precision of a particular query depending on the available system resources.

TinyDB is a distributed query execution framework that aims to minimise the power requirements of a sensor system by applying aggregation and transform functions at the sensor node itself, and by using information about the query to change the sampling regime itself [67]. It also provides a convenient SQL-like query domain-specific language for specifying filters on time-series data

streams, and the transforms on the resulting data. A query to retrieve the metrics `light` and `temp` from a set of sensor nodes once per second would look like:

```
SELECT nodeid, light, temp
FROM sensors
SAMPLE PERIOD 1s FOR 10s
```

Once this query is received by the node, the node will begin sampling once per second and terminate after 10 samples have been taken. This contrasts with other stream processing applications like *Aurora*, which assume the existence of such data and only filter and transform it at the node level before transfer.

Some of the queries identified previously require access to large amounts of historical data as the input to aggregate function, such as pressure fatigue estimation by time-integral. Although *TinyDB* optimises query execution effectively in the online case, many of the query execution strategies involve discarding data that is not of interest, which given that storage on sensor nodes is often severely limited makes future historical studies impossible.

3.3 Summary

The concluded set of functional requirements (listed in section 3.1) are conceptually and algorithmically simple in principle, but naive implementations would be unable to produce results in a timely fashion without such optimisation. Rather than implement custom algorithms for leak detection, transient classification etc, the implemented operators will be used to quickly classify large sections of data for elimination, and the result of this would be then either displayed or fed into bespoke, domain-specific tools.

The strategy that has been chosen in terms of the tradeoff between domain specificity and generality of application is as follows:

- To provide a restricted set of operators via a simple mathematical query language like that of *TinyDB* to enable scientists to efficiently search for data of interest. The implementation

of these will be optimised to make searching large datasets efficient.

- To provide a means for scientists to write custom applications that transform and filter the large dataset, without having to learn about the complexities of distributed systems, or *MapReduce* architectures.

Most of the distributed database systems surveyed that are intended for general-purpose use split the data up into a fundamental blocks, which are then distributed and replicated across multiple nodes in the cluster for availability and in some cases higher read performance. Such general-purpose databases make no assumptions about the underlying dataset, which makes them very fast for reading and writing large blocks of data - but also inhibits the types of optimisations the database itself is able to make on behalf of the user.

Code that is written to interface with these systems must do so via an abstraction layer, and in many cases this prohibits some of the more aggressive optimisations that might otherwise be available. Building a distributed time-series databases using one of the existing implementations would mean their use as a primitive storage layer alone, in which case the full benefit of the system architecture is not being realised. In addition, the introduction of domain-specific knowledge and optimisations would be complex.

The application also places the additional requirement that the answers to such queries should be delivered with very low latency. While the latency demands of billing and regulatory enforcement queries are less critical, closed-loop optimisation and leak prediction applications may require answers several times per second based upon datasets that span many years. While the optimisation of such individual application-specific queries to pre-compute partial results on historical data could theoretically yield a significant increase in speed for such queries, this optimisation would need to be done on an ad-hoc basis for each individual query and would result in significant loss of generality. A general approach to optimising a carefully chosen set of useful primitives would yield a more widely applicable solution.

The implementation of the *Frames* system will now be discussed.

3.4 *Frames* Implementation Overview

In the previous sections, the application was studied to determine what requirements it places on a research database, and what kinds of queries would enable better research studies on large-scale water industry infrastructure. Existing database implementations were reviewed and discussed with a view to assessing their suitability to store and process the large quantities of time-series data that will arise from a large-scale infrastructure monitoring programme. A number of key requirements were outlined in terms of system scalability, reliability and the interface to a user was defined in terms of the class of queries that are needed in order to answer common requirements of the historical data. It was concluded that although a viable system could theoretically be built using one of the existing systems as a storage layer, such a solution would not be optimal in terms of use of resources, and would require significant and ongoing development effort.

The following sections describe the design and architecture of a prototype distributed time-series database system, called *Frames* (Figure 3.2), with particular application to infrastructure monitoring and operation. A system is presented which has been extensively tested over a period of 18 months with real hardware.

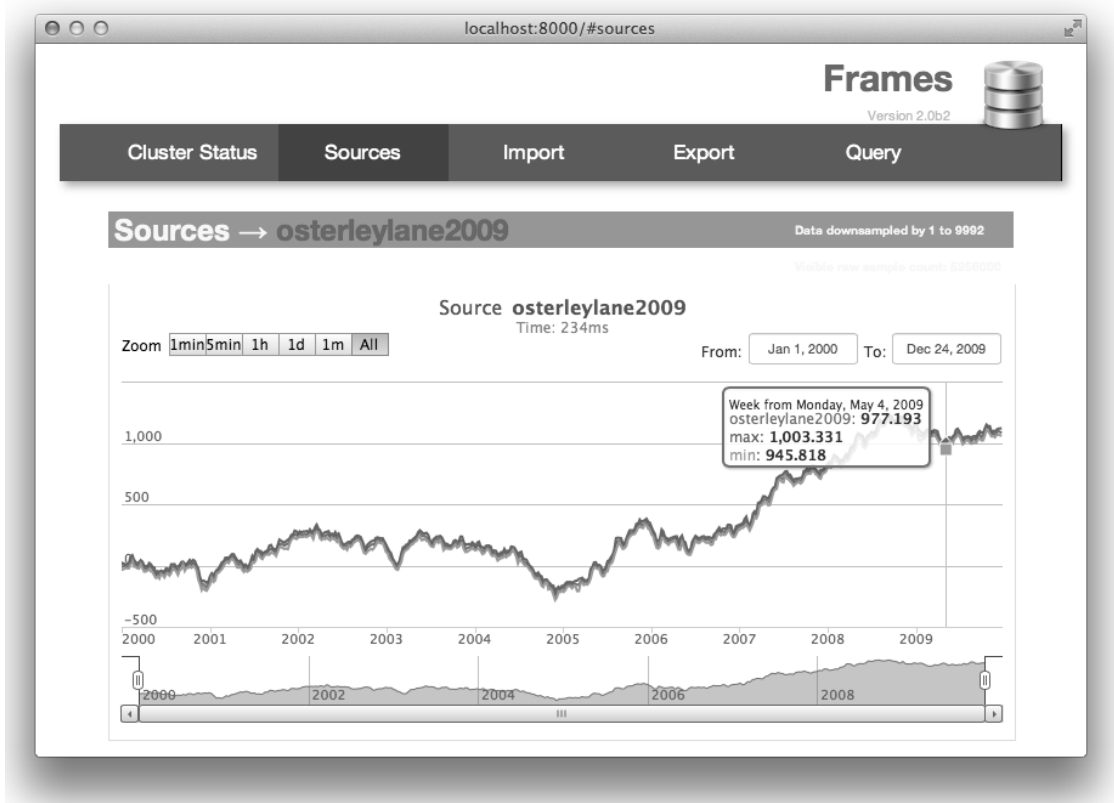


Figure 3.2: Screenshot of a live *Frames* installation.

3.5 Preliminary Concepts and Assumptions

Before the design of the system is discussed, a few practical issues, assumptions and conventions will be outlined.

3.5.1 Network Performance

The careful optimisation of network topologies, routing strategies and packet sizes *etc.* is a production issue that is typically handled by a hosting facility employed by the owner of a system. As such these factors often cannot be specified by the designer of a software system (or, if these considerations are specified, the system must still perform adequately in the likely event that they are ignored). Additionally, designing a distributed database system that responds gracefully to differ-

ing levels of performance of different nodes to the cluster (such as a cluster that spans two different countries via a high-latency link) is a research topic in its own right, but is not the topic of this thesis. Therefore, all references to a network link refer to local Gigabit Ethernet (with an assumed sub-millisecond latency), and the software is not designed to be performant over high-latency or low-bandwidth network links.

3.5.2 Timestamps

For historical reasons a convention in many programming languages is that timestamps are stored as the number of milliseconds since January 1st 1970. This is sometimes referred to as the UNIX *epoch*. Not all platforms provide a way to get a more precise timestamp than this, and many embedded systems only have a granularity of 10 ms despite reporting time in milliseconds. A 64-bit signed integer is sufficient to store every millisecond instant since January 1st 1970, for several hundred thousand years into the future, and so this was chosen as the basic data type for timestamps. To store the value, a 64-bit IEEE 754-1985 [68] floating-point format is used (with a range of $\pm 2.23 \times 10^{-308} \rightarrow \pm 1.80 \times 10^{308}$) because of the fact that the entire range of likely values can be accommodated, and the ubiquity of support in programming languages. Millisecond timestamps theoretically limit the storage architecture to sample rates of 1000 Hz which may not be sufficient for all applications, so no hard-coded assumptions will be made by the system about the actual meaning (e.g. time zone) of these values.

3.5.3 Accuracy

Often it is possible to produce an answer to a numerical query that approximates the true result, but that can be calculated in a small fraction of the time taken to compute the exact answer (or the closest floating-point representation thereof). One method is to sample a small percentage of the dataset, and then to extrapolate the answer to produce an estimated value. Another is to keep track of the convergence of a value during query execution and then terminate once the value stabilises within defined bounds. This model was applied to numerical *MapReduce* aggregations by *Yahoo!* in [69], where a query execution can be monitored while in progress by a user and interrupted once

the desired level of convergence has been achieved.

As mentioned in the previous chapter, a number of the applications of this work are related to failure prediction, where approximate answers may be acceptable. However, for other applications such as compliance or equipment sizing, any unpredictable loss of accuracy may introduce complex legal challenges, especially in the case of uncertainty in the uncertainty itself. Therefore, in this work, the system will always return answers to queries to full machine precision and no optimisation will be considered that sacrifices accuracy.

3.5.4 Protocol

The communications protocols are based on stateless, asynchronous HTTP messages. The full protocol definition used by the nodes to exchange messages with each other is given in Appendix B.2, and the protocol used by clients (such as sensor nodes) to save data and execute queries is given in Appendix B.1.

The implementation of the system will now be discussed.

3.6 Distributed Storage Architecture

The most fundamental component of a database system is the storage layer, which is responsible for persisting the data to disk, serving requests to read that data, and ensuring that specific regions of interest can be located amongst a large dataset. A time-series dataset is a stream of tuples comprising of a timestamp and some floating-point value. The fundamental performance limit of systems that read and process input data from disk as a stream were previously outlined as being the speed at which data could be either read from disk, or eliminated as not needed for a particular operation.

A single-node database that receives read and write requests is free to handle these requests as it sees fit, provided that the client sees a fully consistent view given the operations that have been performed by that client. A common optimisation made by both database systems and operating

systems is to delay writes to disk under the assumption that one large batch of writes is often more efficient than many smaller writes [70]. However, the view of the table by the database, or the filesystem by the operating system, must be consistent with the view that the client expects given the entire set and ordering of its operations. Some queries might therefore be served from memory if they have not yet been written to disk due to such an optimisation. With a distributed system, a client may be free to read and write data from any node in the cluster. In order that the system be scalable, it is not possible to contact every node in a cluster when new data is written, which introduces complexity in terms of data visibility, i.e. when a client writes some data to node *A*, and then tries to read that data from node *B* in a consistent system it expects the data to be available (a happens-before relationship). If two clients make conflicting writes to nodes *A* and *B*, there must be some strategy to detect this and ensure that the global state is consistent.

Whenever co-ordinated actions must be undertaken, the fundamental problem of distributed consensus arises. In the general case it is impossible to ensure that all of the nodes in a distributed transaction either agree or disagree that the transaction should proceed. The problem was identified by L. Lamport in [71], in which he explained by analogy to two physically separated armies intending to attack together:

We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. [...] The generals must have an algorithm to guarantee that: A. All loyal generals decide upon the same plan of action, and B. A small number of traitors cannot cause the loyal generals to adopt a bad plan. The loyal generals will all do what the algorithm says they should, but the traitors may do anything they wish.

It is proven that in the general case it is impossible for two systems to reach consensus on whether some distributed action should take place. Further, it has been shown that a distributed system may enjoy at most two of the properties Consistency, Availability and Partition Tolerance [72].

To mitigate the risk of a single node committing a transaction in error, many distributed databases use three-phase commit [73], or a variation of the Paxos algorithms [74] as a means of ensuring all

cohorts in a distributed transaction agree to commit the transaction. Three-phase commit resolves the ambiguity of two-phase commit protocols about what to do in the situation where both the master node and a cohort fail, whereby it is possible that the failed cohort had actually committed the transaction before failing [75]. The algorithm ensures that there is a deterministic upper-bound on the time within which a transactions status can be defined as successful or otherwise. The possibility that a failed node has actually committed the transaction but no other cohorts were aware of the commit instruction is eliminated in three-phase commit by the introduction of the middle (pre-commit) verification stage.

There are several drawbacks of general consensus algorithms like three-phase commit. Firstly, three-phase commit is verbose; every operation requires at least three messages in each direction. Secondly, during the operation a lock must be held (typically by the master) with a granularity that is at least as large as the area of data that is included in the transaction. In many cases, if information from a database table is needed to decide whether to commit the transaction, this might block the entire database.

The semantics of the dataset that is to be stored will now be considered with a view to informing the choice of distributed storage algorithm.

The data that arrives at the cluster from a single sensor node during typical operation is a list of tuples containing a timestamp, which is a monotonically increasing numerical value, and value, which is an arbitrary floating-point value (for example Table 3.1). There may be many sensor nodes, each with its own stream, but the streams are logically separate and have no interdependency in terms of storage.

Timestamp t	120003	120042	120089	120131
Value v	8.1	8.2	8.13	8.18

Table 3.1: Example sample values.

These samples are a capture of a physical event; as such there is no logical analogue to some of the operations that general-purpose databases must support (such as *delete* or *update*). Once a value has been recorded against a particular time instant there is no reason in normal operation to

modify that value. It is also rare for samples to arrive out-of-order, since a data logger will always record samples in the proper order. Therefore, a simplifying assumption will be made that data may *never* be stored out-of-order. Individual records (samples) are therefore classed as immutable, and a particular data stream is append-only.

Immutable data has several convenient properties in a distributed system. The transactional semantics of database systems are normally considered in terms of atomicity, consistency, isolation and durability (commonly abbreviated ACID) [76]. The storage of new data must be *atomic*; it must either completely fail and leave the system in its previous state (as seen by clients) or completely succeed and leave the system in the new state as instructed by the client. If a storage process operation were not atomic a failure of a node could leave the system in some intermediary state, or worse a state from which recovery is not possible to a well-defined state. A *consistent* system is one that progresses between from one well-defined state to another, and does not permit the system to enter an illegal state or one that violates the preconditions or type system of the database. Transaction *isolation* implies that two concurrent transaction clients are unable to see the effects of each other's transactions until they have completed, client B will either see the database entirely as it was before the effect of client A's transaction, or entirely as it should be after.

Fortunately, a signal-processing database does not need to deal with arbitrary, undefined data sets. Much is known about the data before it is inserted. As has already been stated, it is now known that the data can only be inserted (it can not be deleted) and that the stream is append-only. It is also known that there are no relationships between disjoint data sources that need enforcing. It can therefore be concluded that some functionality offered by full ACID semantics are not required by the system.

The following formal assumptions are now made:

- Each sample within a stream can be uniquely identified by its timestamp (*uniqueness of timestamp*), and
- No sample will arrive to be stored after one with a later timestamp (*monotonicity of timestamp*).

There are a few cases where these assumptions do not hold, such as in the case of a sensor node with a drifting real-time clock that is periodically reset leading to the appearance of old data, but these will not be considered and the modifications required to implement this is left as a future exercise.

We will now consider the implementation of the system in terms of a sample arriving at a particular, arbitrary, node in the cluster, and discuss how that sample is durably persisted for durable storage to the other nodes. The exact replication factor is arbitrary, but many systems in practice use a value of 3 by default⁵ as this offers a good compromise between resilience and equipment cost.

From the assumptions listed in the previous section, but without assuming that a particular node in the cluster is privy to all of the data that comprises a single stream, we can now write some properties of the distributed system:

1. Record with timestamp T is not sent to a node in a cluster before all data with timestamp $t < T$ is stored by the cluster, and
2. If a node receives a *store* request for a data record with timestamp T that is already stored, it will perform no operation, and
3. The distribution of a data record for timestamp T may be retried as many times as is necessary, and with arbitrary order with respect to other operations in the cluster; therefore:
4. If the cluster has a record with timestamp T , then it has all records $t < T$.

The useful implication of item 2 is that no special consensus algorithm needs to be implemented; a master may retry the distribution of a sample as many times as it wishes until it learns of a successful outcome. An implication of item 4 is that (notwithstanding hardware corruption) the detection of corruption or old data on a node is trivially a matter of requesting the most recent (largest) timestamp stored, and can be performed at read time.

A simplified distributed consensus algorithm for storing time-series data will now be outlined.

⁵For high-availability systems it is not uncommon to use much higher replication factors, such as the Amazon S3 storage service which is reported to use a replication factor as high as 7.

3.6.1 Implementation

Each sample stream is given a unique, arbitrary key, which is globally unique for the cluster, such as **ABC**. The stream ID, together with a timestamp is therefore sufficient to uniquely identify a single sample. Each stream has a master node in the cluster, which is responsible for tracking the location of the data in the cluster of that stream. Data is distributed such that an approximately equal amount of data exists on each of the individual nodes that form the cluster as shown in Figure 3.3. In this diagram, Node 1 is the master for a source named **ABC**, with five blocks of data located on the other two nodes, references to which are kept on Node 1. A consistent hash [77] is used to identify the master node for a stream based on the ID of the stream, meaning that there is never any ambiguity as to which node is responsible for a stream’s coordination. The master for a stream is always authoritative as to which data comprises a stream, and is thus used to differentiate between data remaindered after a failed transaction and genuine data that comprises a stream.

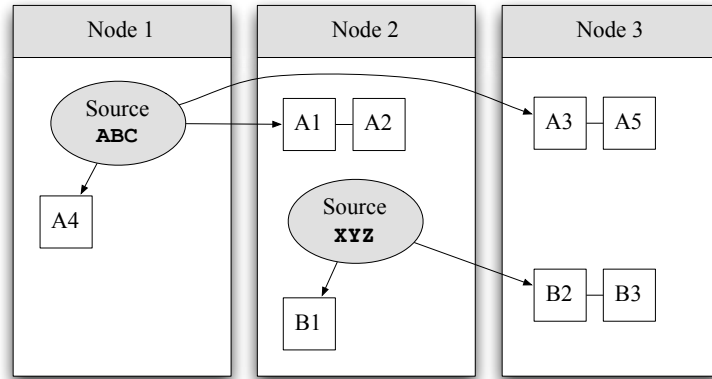


Figure 3.3: Distribution of data amongst nodes in a cluster with no replication, showing master nodes for streams A and B storing pointers to the location of data elements.

In order to simplify the task of storing and accessing the data, samples are kept in fixed-sized blocks, referred to as frames. Each frame has an identifier, which is unique to the stream it is in as shown in Figure 3.4.

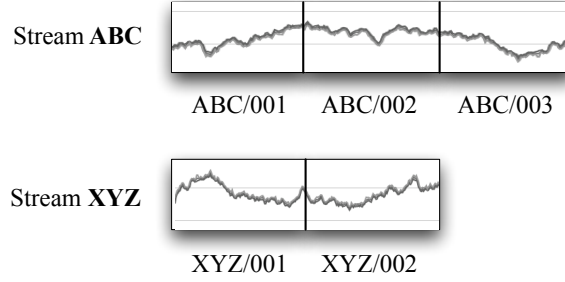


Figure 3.4: Two example streams, **ABC** and **XYZ**, split up into frames.

Typically, and unlike most database applications, all writes to a particular stream will come from a single client. Some clients may write data to multiple sources simultaneously (e.g. (X, Y) position measurements) but it is difficult to conceive of a situation where a time-series value is defined from multiple raw sensor devices. It therefore does not decrease the scalability of the system to have a mutually exclusive write lock over an entire stream enforced from a single location provided that such a lock permits concurrent reads (i.e. a read/write lock). A client wishing to read data from a source that is undergoing mutation is simply provided with the value before the write-lock was awarded, thus providing transaction isolation.

We now consider and analyse the case where node failure happens during the execution of a storage operation, and present a transactional storage algorithm that guarantees against data loss in absence of total failure. The algorithm used to persist a new sample is as follows (full protocol definitions are given in Appendix B):

1. Sample with timestamp in the range T arrives at an arbitrary node N in the cluster from the *client*,
2. If N is **not** the master for this source:
 - (a) The hash function is used to locate the master node,
 - (b) A note is made to include in the response to the *client* request that future requests can go to the correct node directly for better performance,
 - (c) The request is proxied to the master node.

3. The master verifies:
 - (a) T is strictly greater than the most recent sample previously received (otherwise an error is generated), and
 - (b) that the required cohort nodes are at the expected version⁶.
4. Master copies the new sample with timestamp T to the nodes,
5. **Master updates the location table with the locations of the data.**
This commits the transaction.
6. Asynchronously, and at some point in the future, the change to the location table for this stream is copied to other locations in the cluster for redundancy (see section 3.6.2).

The full flow diagram, showing all states and failure modes of the distributed operation is shown in Figure 3.5. It is important to note that should an operation fail at any stage before the master has committed the change to the location table, then even if data has already been sent to a node and stored by it, it will not affect the consistency of that stream and will be discovered and deleted at some arbitrary point in the future.

⁶This can be cached to save a network access, as the version can never decrease.

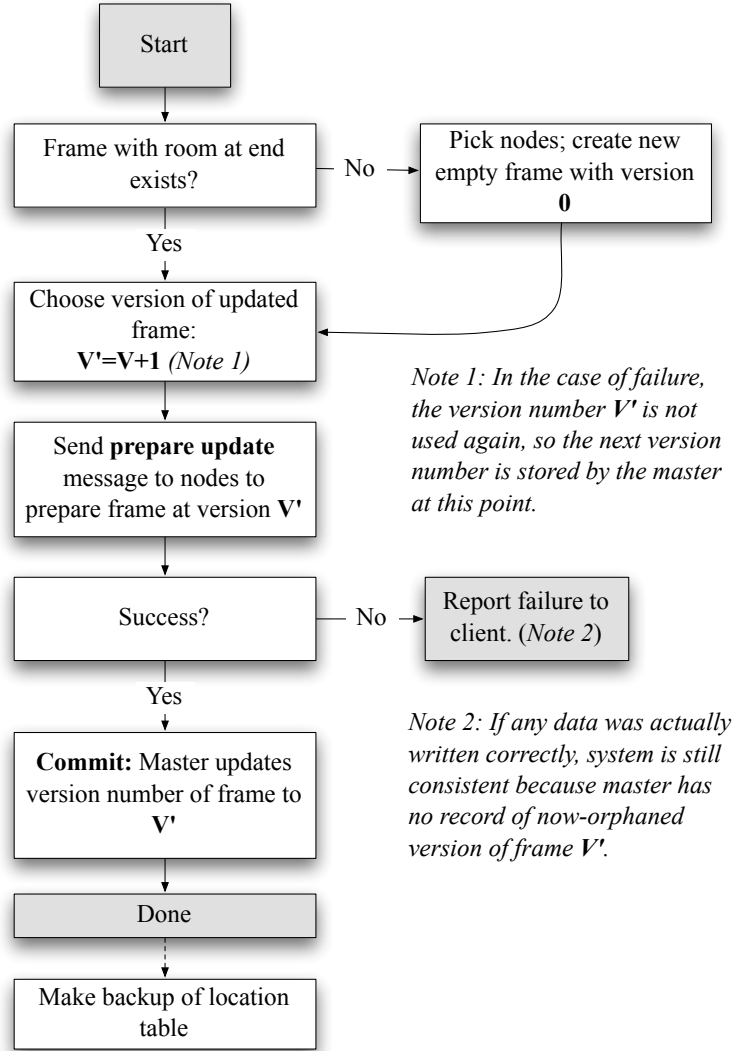


Figure 3.5: Flow diagram showing distributed append operation between master node, cohort nodes, and client.

While some sources, such as temperature sensors, may send samples in batches of one, a high-frequency sensor node such as would be needed to detect pressure transients is more likely to send samples in much larger batches. If the entire process is repeated for each individual sample, then poor latency will result as each pass of the storage algorithm requires network communication and disk access. Therefore the append process is optimised by sending data in batches. Each of the steps in Figure 3.5 can be applied to a batch of samples, and so two simple optimisations are made:

1. When a batch of samples arrives, these are persisted together in a single *append* operation,
2. The system waits a small amount of time (configurable, by default 100 ms) after a sample is received for further samples to arrive on the same stream, and then treats these as a batch and acknowledges their persistence together.

3.6.2 Failure of Master

The risk associated with a single point of failure grows super-linearly with respect to the size of the cluster. A distributed system must be able to gracefully handle the loss of at least any single node, preferably more. In *Frames* a given node may be the master for a number of sources, and the master is used to locate the data. Therefore when a given node fails, the master will become unreachable for a proportion of the streams in the database. Since the master stores the table that records the location of each block of data it therefore could represent a single point of failure. This is mitigated as follows.

After an *append* operation has completed, the stream location information (which stores the location of the data that constitutes a stream within the cluster) is asynchronously replicated to other nodes in the cluster that hold data for that stream, which is dependent on the replication factor in use. The location table is typically several orders of magnitude smaller than the data itself, so this represents a negligible overhead.

Since this happens *after* the write is acknowledged to the client, it is possible that if a master node fails between this acknowledgement and the successful distribution of the backup location table then a small amount of data will be lost. In practice, provided that the replication time lag is low, the probability of this happening is comparable to the probability of the loss of a stream as a whole due to a number of nodes failing that exceeds the replication factor.

The problem of leader election [78] is addressed by taking the set of nodes that store the first frame of data for the stream, and arbitrarily picking the one with the node *ID* that alphanumerically sorts lowest. This can be done by all the nodes that comprise this set upon realising that the master is offline. This strategy is *not* partition tolerant, however; should some nodes decide that the master for a source is offline, and others decide otherwise, then the node with the lowest-sorting name may

incorrectly elect itself as the new master.

3.7 Node-Local Append Atomicity

The part of the *append* operation performed on the node is a transition from one version (x) of a frame to another via some additional data, which we denote $V_x \rightarrow V_{x'}$.

For any guarantee of atomicity in a distributed operation we first require that the individual operations performed by the cohorts are themselves atomic. If this is not the case it is not possible to reason deductively about the state of the system from the messages received by the cohorts, and a fail-safe transactional *append* operation is not possible. The individual components of the transaction on the cohorts may happen at an unspecified time and in an unspecified order on that cohort.

The master of the transaction learns of one of the following states after attempting an *append* operation with a cohort:

1. **Known Success** $V_{x'}$

The master learns of the definite success of the cohort, in which case the cohort's operation was successful (and durable) and the version of the relevant frame is now at $V_{x'}$, or

2. **Known Failure** V_x

The master learns of the definite failure of the cohort, but in which case the cohort remains at the stable version before the transaction was attempted (V_x), or

3. **Unknown Failure** $V_?$

The master unsure as to the result of the transaction for some reason, or a specific reason.

Case 3 can occur as a result of network failure (e.g. bad link or partition), hardware failure, power loss, or any number of other causes. In order to supply an atomic guarantee the node software must never enter a state, even transiently, whereby a failure of the system (e.g. loss of power) causes data loss, i.e. we require that the frame file is always in a state from which either version V_x or $V_{x'}$ can be recovered.

Tests showed that appending an arbitrary amount of information to a file is not safe with respect to power outages during those writes. Data is written by most operating systems in blocks, which are typically 4096 bytes in size. When an amount of data is written that exceeds the page size then it is impossible to predict how much of the data will be written to the file on a filesystem that is subject to failure. In addition, writes to other areas of the file are not guaranteed to be written correctly. Journalled filesystems mitigate some of the possible low-level corruptions at the expense of increased overhead, but application-level corruption is still possible if the system is not in a consistent logical state at all times. Therefore, when data is appended, it is not safe to simply append the data to the file and *sync* the filesystem, since failure during this stage can leave an undefined amount of data written.

One solution to this atomicity problem is to have the master upgrade the cohorts one at a time. If the first node fails, then the *append* operation is rejected and the system can be recovered to a known state by restoring the frame file from another (untouched) node. There are two problems with this strategy:

1. This does not work on single-redundancy configurations ($R = 1$) without making a backup of the frame file before the transition is attempted, and
2. In higher-redundancy configurations, the operation would take much longer as each node must be contacted sequentially, and
3. If a failure occurs updating a node, the system would be unable to take further action without risking data loss as that node would be inconsistent, and a second failed update to another node could render that region irretrievable.

An algorithm was developed that firstly offers a guarantee that each node will be consistently in either V_x or $V_{x'}$, using the fact that most filesystems guarantee that the rename operation is atomic (the POSIX standard mandates this [79]), and secondly is able to return authoritatively to the master after only three filesystem operations.

Before a frame is filled (which happens when the number of samples equals the maximum for a frame), it is kept in duplicate on each node: a primary file and a secondary file, which we call ‘a’

and 'b'. When the system is quiescent, both files are identical and 'a' serves all reads. When an update arrives, the update is first applied to the secondary (b) file, and when that change is synced successfully to the underlying storage medium, the update is applied to the primary (a) file.

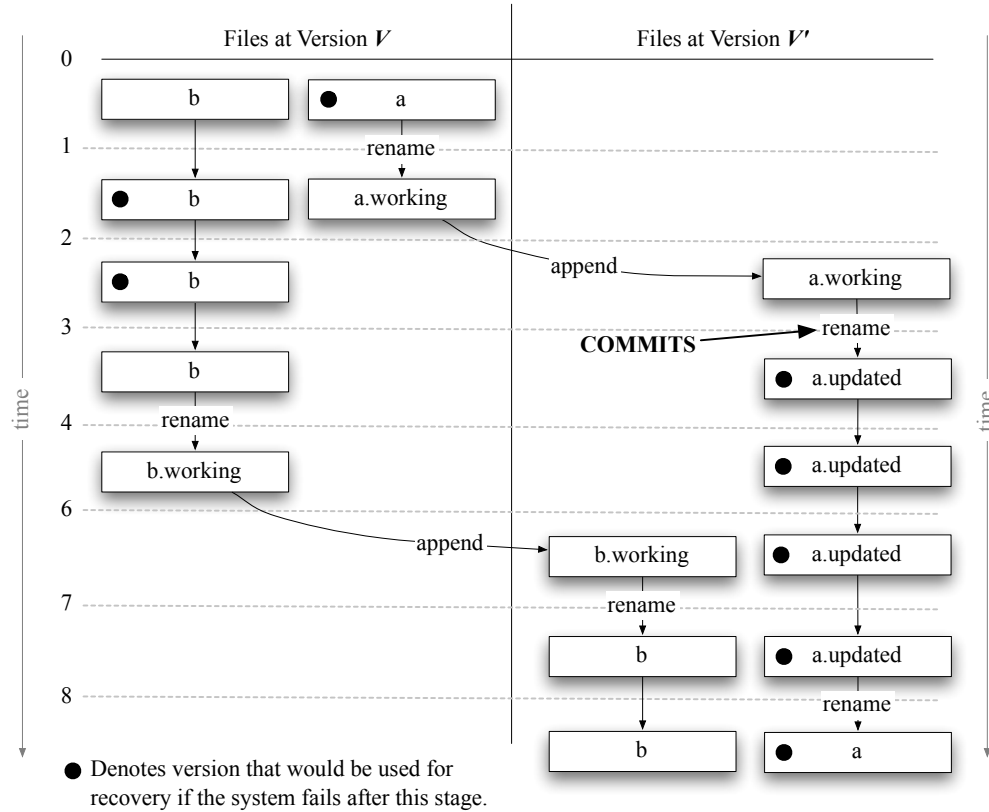


Figure 3.6: State-flow diagram of the transactional append logic, showing absence of an intermediate state where data can be lost. Each path through the diagram represents the operations performed to the slave (**b**) and master (**a**) files.

Figure 3.6 shows the sequence of operations needed to upgrade the system. In this diagram, each operation is given a clock tick between 0 and 8. Even though not every individual operation is not atomic (appending to a file is unreliable as stated), should there be a failure at any point during the process the system is able to recover to a consistent state. It is important to note that the transaction can be committed, and the master notified, after operation at clock tick 3, because should the node fail after this stage then the node can automatically recover to version $V_{x'}$ of the file by continuing the transaction from where it was interrupted.

There are two key advantages to this algorithm:

1. A node is only ever in one or other known (recoverable) states, hence it can be performed safely on all cohorts concurrently, thus decreasing latency,
2. The algorithm only adds the overhead of two filesystem rename operations to the latency of the commit path since the transaction can be marked committed after clock tick 3.

The discussion of the implementation of the prototype time-series database system so far has discussed the design and implementation of the low-level architecture, and how to reliably, consistently and concurrently append data to a stream. In the sections that follow, the methods and algorithms used to read and query data will be discussed.

3.8 Operator Definition

The framing of data introduces some complexity in terms of how to execute a query on the data, since the data is no longer located contiguously in one place. However, it also offers a convenient primitive upon which to distribute the execution of some queries. In addition, and as will be shown, the frames of samples are suitable for indexing. In this section, the property is used that since data is immutable, once a value has been computed for a particular range of data it does not need to be computed again. It will be shown that per-frame indexes can increase the performance of query execution significantly.

Other than simple mathematical functions and operators, there are no domain-specific built-in functions in *Frames* because the definition of the appropriate fluid-dynamic functions is outside the scope of this work, and their definitions would likely vary between specific applications. The only built-in aggregators are `max()`, `min()`, `count()`, `sum()`, `histogram()`, `dxdt()` and these have custom implementations and optimisations as will be shown later in this chapter. Therefore, every non-trivial operation must be defined as a plug-in (supplied via a *jar* file to any node in the cluster). In this section the way these functions are executed in parallel, and the two basic types of functions are discussed.

3.8.1 Aggregators

Aggregator functions are often called *embarrassingly parallel* functions because there is no dependency between the constituent parts of the evaluation algorithm. Many functions are associative, commutative and distributive, such as *sum* and *count*, and these can be computed first on the subsets of the dataset on each node, and then the results aggregated via further applications of the same function on the intermediate results to compute the final answer; for example:

$$\text{sum}(a, b, c, d) \equiv \text{sum}(\text{sum}(a, b), \text{sum}(c, d))$$

The *MapReduce* programming paradigm [58] requires an algorithm to be implemented in terms of a two-stage aggregation process: the *map* stage transforms values in parallel into intermediate results, and then the *reduce* (and sometimes also *combine*) stage aggregates these intermediate results to produce the result. In this work, an approach similar to *MapReduce* is taken to evaluate user aggregations in parallel on the cluster.

An user-supplied aggregator function in *Frames* provides a definition for the following Java interface:

```
interface Aggregator {  
    String name();  
    double reduce(ImmutableList<Sample> values);  
}
```

where the `Sample` type is a tuple of timestamp and value. The `name()` method is used to identify the aggregate, and is the function name that can be used in the query language.

The presence of an aggregate in the system implies two things:

1. A function with the name returned by the aggregate `name()` method is available in the query language.
2. When data is added to the system, an entry for the aggregate will be automatically built for each index block and stored in the index using the `aggregate()` method. If an aggregate is defined after data is loaded, then the indexer will compute the values in the background until the data is indexed.

3. When a query is executed that refers to the aggregate, the value is read from the index rather than by computing the value from the stream, yielding a significant performance benefit.

A example aggregate for the built-in `sum` method is shown below in *Java*:

```
final class SumAggregator implements Aggregator {
    @Override String name() {
        return "sum";
    }
    @Override double aggregate(ImmutableList<Sample> values) {
        double sum = 0.0;
        for (Sample sample : values) {
            sum += sample.value();
        }
        return sum;
    }
}
```

If this *Java* class were loaded, the function name "`sum`" would be available in the query language.

3.8.2 Windows

A large class of functions are non-associative, meaning results from independent sub-calculations cannot be so trivially combined. Many algorithms in signal processing require a sliding window, which cannot be computed on individual samples in parallel and then combined afterwards. For such functions additional logic must be implemented to handle the case where the algorithm crosses a frame border. A moving-window algorithm computes the value at each time instant based on a number N of samples before and after that value ($2N$ in total), and therefore if a node does not have all the data needed to compute the function it cannot compute a value within N of the end of its data range.

The simplest example of an operation that requires access to groups of samples is the moving-

window average filter. For each output sample, a range of samples to the left and right are read and averaged. To compute a moving-window average of size N , element x is given by:

$$e_x = \frac{1}{N} \sum_{i=x-N/2}^{i=x+N/2} e_i \quad (3.3)$$

With a single, linear dataset this can be implemented simply by iterating over the samples, accessing each sample as needed (e_i) until the operation is complete. In a distributed database not all of the samples are available to each node performing the operation, *i.e.* those samples that are across a frame boundary as the iteration reaches $N/2$ samples from the end of a particular frame.

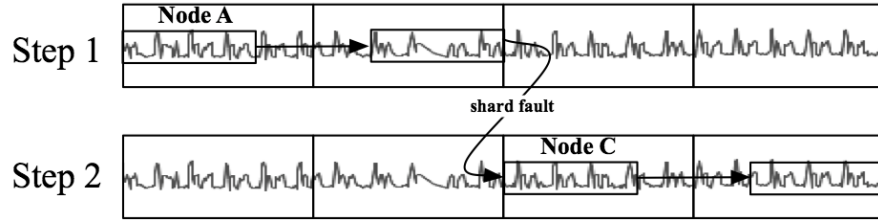


Figure 3.7: Computation of a moving-window filter on a dataset.

Figure. 3.7 gives an illustration of a moving-window filter being performed on a cluster where the data is split up into four frames, two of which are stored on Node A and two on Node C. While Node A is performing the filter it approaches the limit of the dataset stored on that node. The master node then works out which node has the largest contiguous range of frames, and the state of the operation is suspended using a continuation [80], and sent to (in this example) Node C. The continuation contains all the state of the current operation needed to resume processing on the next frame, which in this case is the contents of the array of samples of size N that was being used to compute the filter, and the value of the summation index (i). Finally, since each frame is stored in multiple locations, the process can be optimised by looking for the path the query continuation should take through the nodes that minimises the number of potentially expensive continuation moves. This technique generalises to any operation that operates in a fixed-size sliding window, and that stores a constant amount of state with respect to the size of the overall dataset.

A general, non-aggregate *Window* in *Frames* must provide a definition for the following Java interface:

```

interface Window {
    void map(Context context, Sample sample, Collector collector);
    Sample combine(long timestamp, double[] values);
}

interface Collector {
    void emit(long timestamp, double value);
}

```

Intermediate values are emitted to the *Collector*. The implementation must use the supplied *Context* object for all state, such that the system is able to transparently serialise and transfer execution to another system. The `combine()` method is then called for each unique timestamp with the different values that were produced by the `map` method,.

The `map()` and `combine()` methods of the *Window* are somewhat analogous to the *map* and *reduce* methods of *MapReduce*. The `map` method can in some instances be called in parallel on multiple areas of the dataset simultaneously so that all nodes that have data for a particular region of a stream are utilised concurrently.

Example

To illustrate the querying and index generation functionality, an example of a moving-window filter will now be used.

A simple 8th-order moving-window average filter shown is in Figure 3.8. To generate the value at time instant 0, the filter sums the eight surrounding values (four in each direction) and multiplies each value by the appropriate coefficient (shown on the y-axis).

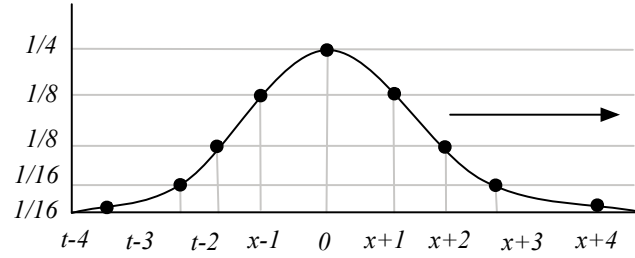


Figure 3.8: 8th-order moving-window average filter. The filter sums the value at each position multiplied by the associated coefficient, then the entire filter moves forward one place and the process is repeated.

An concrete example implementation for this algorithm is shown below. For each time instant (i.e. input sample) the `map()` method emits the block of samples that will be used by the filter in the `combine()` stage. The `combine()` method will then be invoked for each set of values for each unique timestamp that is emitted. Details of initialisation, and some unambiguous method bodies are omitted for brevity.

```

final class MovingWindowAverage {
    static final double[] FILTER_COEFFFS = {1./16, 1./16, 1./8, 1./8, 1./4, ...};

    @Override
    public void map(Context context, Sample sample, Collector collector) {
        double[] buffer = context.get("buffer", double[].class);
        shiftBack(buffer); // Move every element in the buffer back one place
        buffer[N-1] = sample; // Add sample to the end of the buffer
        // Emit all values
        for (Sample s : buffer) {
            collector.emit(s.timestamp(), s.value());
        }
        context.update("buffer", buffer); // Save the context
    }

    Sample combine(long timestamp, double[] values) {
        // Multiply the window values by the filter coefficients
        double sum = 0.0;
        for (int i=0; i<values; i++) {
            sum += values[i] * FILTER_COEFFFS[i];
        }
        return new Sample(timestamp, sum);
    }
}

```

When this example is executed the *Frames* query executor will call the `map()` function once for each sample in the dataset, and each time it does so the function will emit the previous 8 sample values to the `combine()` method.

3.9 Query DSL Structure

This section considers the design and formulation of the DSL used to express search queries to the database system. The DSL is intended to resemble other similar DSLs in use for offline stream processing systems, such as that of the SQL-like query language of *TinyDB*.

The following design considerations were used to design the language:

- Simple concepts should be expressed in simple queries.
- Concepts that are familiar to the user should have familiar representations.
- The number of new, special symbols should be minimised.

These specifications lead naturally to a query language that is similar to a mathematical expression, but with database-specific functions represented as function invocations. A math-style language will therefore be used here as a basis for the query language.

3.9.1 Mathematical Clauses

The basis of the grammar is a mathematical expression language. Most users will be familiar with the conventional way mathematical expressions are used in almost all programming languages, and even some scientific calculators, for example:

```
1 * (10 - 4.5) / 2 + sin(45)
```

The `word` token type is used to refer to variables, which represent the human-friendly identifier of a stream. For example:

```
(max(stream1) - min(stream1)) / count(stream1)
```

3.9.2 Time Range Filters

The language is now extended to incorporate the concept of a time filter, which is the most basic form of selection supported. Ideally, the number of characters needed to express simple time ranges should be as short as possible - the user should not have to completely specify the full date range of interest.

A time range filter has both absolute and repeating forms. The absolute form is always date range, such as 2001-2005, and these are evaluated and bound to a specific range of millisecond instants. The repeating form is inferred where there is any missing specifier in-between two absolute specifiers. The filter 2010 9am specifies a year, and a time, but no month or day, and is therefore a repeating instant (the instant 9am on every day in 2010).

If the query contains only absolute date specifiers in descending order with none missing, and starts with a year, then the a single time range is used:

Filter	Interpretation
2010	Every instant in the entire year 2010, excluding the first instant in 2011
2003-2005	Every instant in the years 2003, 2004 and 2005
2012/5/22	The entire day May 22nd 2012
2012/4/1-2012/4/14	The first two weeks of April in 2012
2000-now	The entire period between 2000 and the instant the query is executed

If the time range contains absolute data specifiers, but there is some missing intermediate specifier, or missing date information when a time is specified, then a repeating range is assumed. For example, if a time range contains the year and the hour, then it will produce 365 instants in that year.

Filter	Interpretation
2012 4am-6am	The two-hour period between 4am and 6am on the day the query is executed)
9am-5.30pm	Every day in history between 9am and 5.30pm
2012/05/22 9am-10am	The one-hour period between 9am and 10am on May 22nd 2012 (half-open)
2012 August Monday	Every entire Monday in August, 2012)

Time ranges are evaluated lazily, so they can theoretically produce infinite recurrences without resource exhaustion.

To help make the DSL as intuitive as possible, the time range specification grammar used is similar to an array dereference in a C-style language, and binds to the variable name:

```
sum(foo[2012])
sum(xyz[2012-05-22])
sum(bar[6am-8am])
```

An example of a potential mismatch between the user’s expectations and the evaluation of a query is in the choice of open/closed ranges. Convention dictates that ranges are half-open (inclusive start and exclusive end), such that `9am-10am` does not include the instant `10:00.00am`, whereas most users would expect the year range `2004-2005` to encompass the whole of both the years 2004 and 2005. Therefore dates always open *and* close a range, and times always open and close a range as shown below:

Filter	Interpretation
2008-2009	The whole years 2008 and 2009
2008-2009/01/01	The whole year 2008, <i>plus</i> the first day of 2009
1995 7am-8am	The 365 one-hour periods that are between 7am and 8am in 1995

3.9.3 Filters

The time-range filters outlined in the previous section offer a simple way to limit the scope of an aggregate expression by time window, or repeating time interval. A more powerful way to express the scope of an expression will now be outlined.

A predicate filter query, such as:

```
src1/1.05 > min(src1) + 4.5
```

returns a list of time ranges for which the predicate holds. In order that the user is able to use this to perform some further processing, a new operator will be introduced.

The pipe symbol (`|`) is used in UNIX shells to indicate that the results from the left-hand-side should be passed to the right hand side for evaluation. The same operator is used here. Expanding the above example, a query might be written as:

```
src1/1.05 > min(src1) + 4.5 | sum(src1*10.0)
```

Instead of returning the sample stream to the user, the above query would use it to limit the scope of the summation, `sum(src1*10)`, which would then only be applied to those time ranges that match the predicate filter on the left-hand-side.

3.10 Indexing

The previous sections outlined that in order to improve aggregate execution performance, the results would be stored in indexes. In this section, the design and implementation of an index system that is able to store and retrieve arbitrary index data for sample streams is presented.

On a raw storage device such as a magnetic (or solid-state) hard disk, data is stored and accessed by the byte offset of the data. Since this is not a particularly convenient format for a humans to locate data, a filesystem abstracts these offsets into human-friendly path names. The data on a filesystem's underlying block device is still accessible by the byte offset; the path is an index to this data that is simply more useful to a human user. Some filesystems allow more indexes to be created based on other metadata, such as the modification time or even the content of the file.

Higher-level abstractions can be built on top of lower-level storage devices. Data records stored in relational databases typically have a primary key that is auto-generated by the database, by which records can be rapidly accessed (mostly in constant or logarithmic time). In order to locate a record by a column other than the primary key a costly table scan must be performed to find the record index based on the search value. To improve the performance of such a query, databases use indexes to store the inverse relationship, i.e. a mapping from value to record index.

Different types of indexes have different properties. In-memory indexes often use trees or hash tables [81], and there are similar data structures for on-disk storage [82]. Compound indexes allow

a database to locate data based on more than one column at once. Complex compound index structures such as *quad* trees enable advanced lookups such as geo-spatial locality and collision detection [83].

As discussed in section 3.6.1, data in *Frames* is stored in fixed-length shards (called frames) in order to simplify the task of distributing and accessing ranges of samples. These frames are also a convenient foundation for indexing and query evaluation.

3.10.1 Index primitives

The potentially vast size of the datasets that will be stored and accessed means that a strategy whereby fixed, pre-defined metrics are used and stored when data arrives at the system would be a convenient implementation. However, the signal-processing functions that are used at runtime may change as these can be expanded by the user. Therefore the system must be capable of storing data into arbitrary indexes that is generated from these user-defined signal processing functions. It should also be possible to re-generate indexes based on new functions that are added while the system is in operation, without an interruption to query execution. These requirements mandate the design of a flexible approach to indexing.

Indexes in *Frames* are therefore an arbitrary collection of key-value pairs, with a header that can be used to locate the raw data in the file that the index block refers to. One index block contains the data for a particular range of samples, and the frame to which it refers. Multiple index blocks are organised into a hierarchical structure, so that one larger index block may overlap many smaller index blocks, with each frame having at least one index block covering all samples in that frame. The data stored by the index block in the *data* region is arbitrary and dependent on the query functions that are available. The structure format is shown in Figure 3.9. By default, each frame has a single index block, and 10 sub-index blocks, although this is configurable.

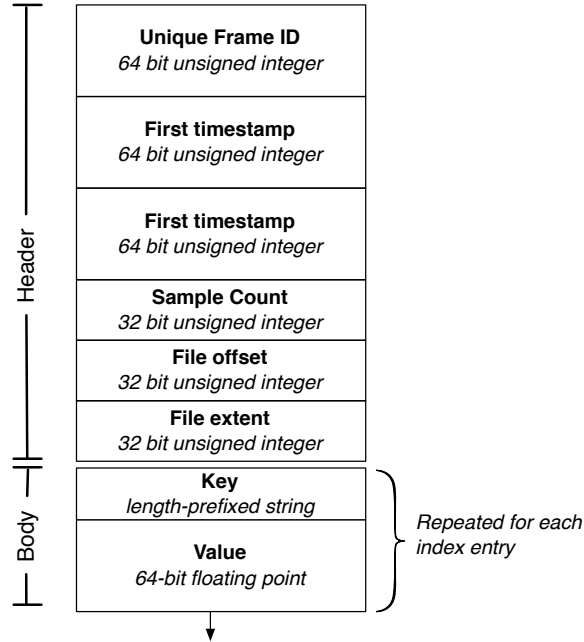


Figure 3.9: Format of index block record.

As discussed in the previous section, each stream has a different master. The master is responsible for keeping the index up to date as new data arrives, and storing backups of the index data elsewhere in the cluster for durability. The same strategy is used as described 3.6.

Indexes are consulted to satisfy predicate filters, and to eliminate data from consideration. Section 3.11.5 shows how the index blocks are used to execute particular queries.

3.11 Query Evaluation

In this section the algorithm used to execute a query will be described. Query evaluation relies heavily on index data to eliminate as much data from consideration as possible, and a query optimiser attempts to reduce the execution time of a query by replacing sub-expressions with known values and cacheing intermediate values during execution.

The execution of query can be broken up into the following stages:

1. Tokenisation, parsing and AST generation
2. Constant value replacement
3. Aggregate expression evaluation
4. Indexed stream evaluation

The four stages will now be discussed.

3.11.1 Parsing and AST Generation

The transformation of a query from a character string to an AST (abstract syntax tree) is performed using a hand-written parser. Machine parser generators such as Antlr and Yacc were not used as the query language is relatively simple and math-based, and for simple languages hand-written parsers can yield greater flexibility in terms of the interface between the parser-generated tree structure and the query execution system.

First, the query is transformed into a stream of tokens according to the rules of the grammar. The fundamental types are: **number**, **word** (such as a variable name), **functionword** (the name of a function e.g. **sin**) and **operator** (which includes symbols such as parenthesis). At this point the token stream represents a set of infix clauses. Then the infix clauses are transformed into postfix clauses (using an algorithm similar to Dijkstra's Shunting Yard [84]). Next a stack-based transform is used to convert from the postfix representation to a tree. The tree is used for analysis, but the postfix clause list is also transformed into an execution-ordered program (i.e. a dependency tree) in order to preserve information about clause dependencies.

3.11.2 Constant Value Replacement

After the AST has been generated, constant terms are replaced with their calculated values. This is a trivial optimisation: the tree is re-written such that any *constant* node is replaced with its numerical value, where a node is *constant* if:

1. It is a **number** token, OR
2. All child nodes are constant terms.

If, after replacement of constant terms with numerical nodes the entire query tree is constant, then the query result is returned immediately to the client. An example of this substitution is shown in Figure 3.10.

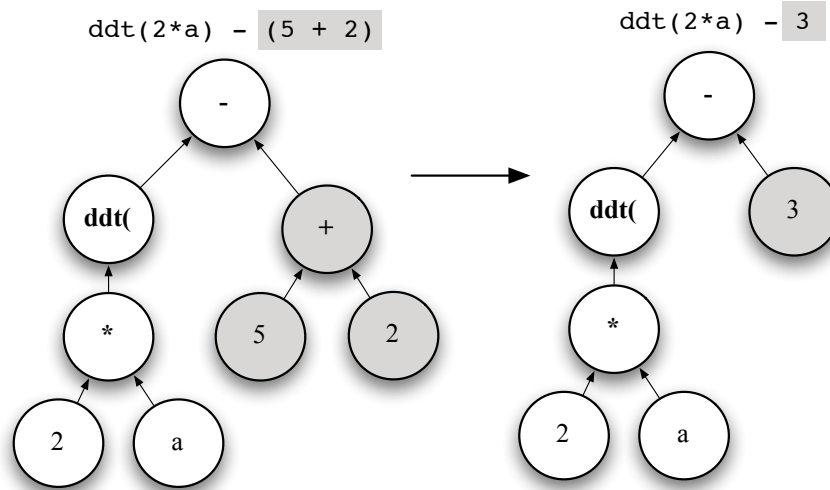


Figure 3.10: Example of automatic translation of constant sub-expression into numerical term.

3.11.3 Aggregate Expression Evaluation

Some query clauses are non-constant but trivially determinable, and have no dependence on other parts of the execution of a query. Aggregate functions often fall into this category; consider the following query:

`a > max(b in 2011)/10 + sum(c) for 1h`

This query returns all of the time ranges where the stream **a** satisfied a certain predicate clause for a period of longer than 1 hour, and has a dependency on three different data streams, **a**, **b** and **c**. Note that the terms `max(b in 2011)` and `sum(c)` are constant terms in disguise - they are simple

facts that can be calculated in parallel before the execution of the query begins⁷.

Because constant terms often have the same form, and are likely to be re-used, they are stored in the query execution cache, along with the stream(s) and time range(s) that they refer to. The part of the term that is cached is the most restrictive non-constant sub-query, so given the term $\max(b \text{ in } 2011)/10$, then $\max(b \text{ in } 2011)$ is added to the cache.

Should an *append* operation occur then the affected terms are immediately invalidated from the cache - no attempt is made to update cached values.

Figure 3.11 shows an example for the query $a[2012] > \max(a[2010])$, which returns a list of time ranges in 2012 where stream *a* is greater than its maximum value in 2010. The maximum value in 2010 is an immutable fact, and so is replaced by the optimiser before considering the execution strategy for the remaining comparison expression.

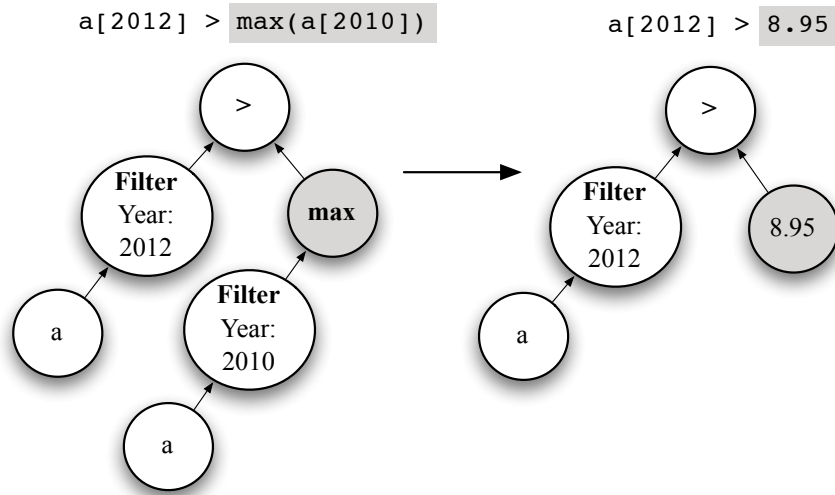


Figure 3.11: Example of replacement of reducible fact with numerical value.

⁷This may not always be the optimal strategy; a sophisticated semantic query optimiser could spot that $a \leq \max(a)$ always returns true and replace the entire expression with the constant value **true**. Such optimisations are left for future development effort.

3.11.4 Algebraic manipulation

The query executor tries to evaluate aggregators using index block data as far as possible, except where a time range predicate includes only part of an index block, in which case the corresponding data is read from disk. Some functions require some transformation before the answer to an aggregate query can be satisfied entirely from index blocks. Although associative, distributive and commutative, the `sum()` operator cannot be naively applied to the `sum` frame index held in memory. Consider the following expression:

```
sum(a - min(a))
```

In this expression, the user is asking for the sum of the difference of every value from the minimum value. If this is evaluated against the `sum` metric from N indexed blocks as-is, the result would incorrectly depend on the number of indexed blocks used for evaluation:

```
sum(a) - min(a)*N
```

Such operators require transformation of their arguments before index block-based evaluation. Constant terms that appear as an argument to plus or minus are moved outside the summation and multiplied by the underlying number of samples that have been evaluated by the indexed block, i.e.:

$$sum(a - min(a)) \rightarrow sum(a) - min(a) * count(a)$$

Arguments to multiply or divide are moved outside the summation and applied to the result with no adjustment. The above rules are applied recursively, e.g.:

$$sum(a + a * (1 + min(a)) + 1) \rightarrow sum(a) + (sum(a) * (1 + min(a))) + count(a)$$

Standard deviation presents a practical problem in that it appears to require an iteration involving each sample in the underlying dataset. It is defined as:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad (3.4)$$

where μ is the mean of the dataset. The algebraic manipulation engine is able to manipulate the `stdev()` function in order to express it in terms of indexable quantities as follows:

$$\begin{aligned}\sigma &= \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \\ \sigma^2 &= \frac{1}{N} \sum_{i=1}^N (x_i^2 - 2\mu x_i + \mu^2) \\ \sigma^2 &= \frac{1}{N} \left[\sum_{i=1}^N x_i^2 - 2 \left(\sum_{i=1}^N x_i \right) \mu + N\mu^2 \right]\end{aligned}\tag{3.5}$$

The two summation terms in (3.5), sum of squares, and simple sum, are quantities that can be pre-computed and indexed over an arbitrary range of samples. Therefore the standard deviation can be computed from index data in the same way as the other aggregators.

3.11.5 Indexed Stream Evaluation

The majority of the query execution is in the *evaluation* stage, where the parts of the query that have not already been either transformed or eliminated are computed. Query execution relies heavily on index block access, since index blocks are kept in memory whereas the access to the raw data streams requires a costly disk access. Since it is known that aggregate functions in *Frames* are associative, the query executor is able to make some powerful optimisations.

Aggregators

First, a simple example query will be considered, which could be a query itself, or a part of a larger one. Consider the following input, which executes a simple time-ranged aggregate and returns a single numerical value:

```
max(stream42[2011])
```

First, the index blocks for all the frames that are *wholly* in the time range are consulted for their answer to the `max()` query. In the above figure, this is the whole-frame index block for frames *C*, *D* and *E* as shown in Figure 3.12.

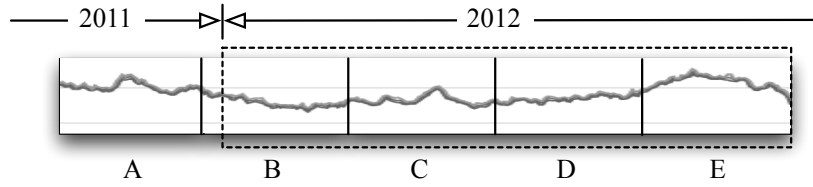


Figure 3.12: Simplified example of a data stream split up into five frames, with query bounds starting within a frame and extending past the end of data.

In this example, the index block for frame *E* returns the maximum value. Next the index block for frame *B* is consulted, as it is partially included in the time range filter. Then, this value is compared with the existing value. If this value is less than the value for the entire stream excluding that frame (which it is in this example), there is no need to read that frame from disk. If it is greater, then the frame must be read to find the maximum value in the range that is included by the query, and this value is then compared with the *E* frame value. If sub-index blocks are configured then these can be consulted by any query where the time range includes values from more than two frames.

This strategy means that the chance of needing to read any data from disk at all is small, and diminishes with the size of the dataset⁸. It also generalises to queries that specify more than one time range, although for these typically more disk reads are needed to read edge values as the individual time ranges are smaller.

Filters

The next example is more complex, and deals with the execution performed when a time range is used for a subsequent calculation. Consider the following query, which takes the average of all of the values in a dataset that satisfy the given predicate filter:

```
stream42 > min(stream42)/2 | sum(stream42)/count(stream42)
```

⁸Note that this strategy is not applied in the general case to user-defined aggregators because it cannot be determined for these whether every value contributes to the answer.

The expression on the right-hand-side of the pipe (1) determines the strategy used to evaluate the filter as follows:

- In the trivial case where the RHS does not depend on a variable, then it is returned immediately,
- The parts of the RHS that do not depend on a LHS filter variable are evaluated and replaced,
- If the RHS is an aggregation depending only on continuous variables of the LHS then a query evaluation object is constructed that performs the RHS aggregation as the filter is evaluated,
- In all other cases, the stream is filtered first (see below) to produce a stream of time ranges, and the RHS is evaluated on each of the returned time ranges. This is done in parallel, as soon as a matching time range is produced it is passed to an available node for that frame for processing.

The filter strategy, as shown in Figure 3.13 is as follows. The master begins evaluating the predicate against index blocks that match any time range selector in the LHS of the query (in this example none). As soon as a frame is found that might match (i.e. it is not excluded with certainty by the filter), this is added to a queue for processing by the cluster, and the first available node with that frame then evaluates the predicate on the data in the frame and returns the value. As values arrive, they are passed to the RHS of the expression for evaluation and aggregation to form the final result.

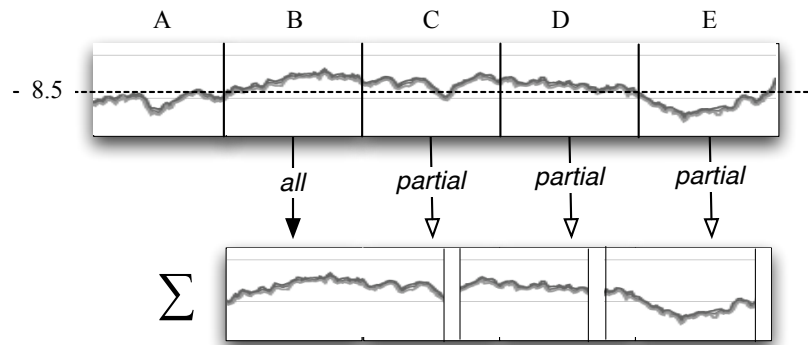


Figure 3.13: Time-range filter on a five-frame dataset, showing elimination of data and operator aggregating the data of the returned time ranges.

3.12 Validation and Performance Tests

To validate the performance of the proposed system, an experimental programme was conducted with a cluster of 10 servers at Imperial College. Each server had a single 500 GB SATA 7200 RPM hard drive, a dual-core Intel Xeon 3 GHz processor and 2 GB RAM.⁹ The local network was an uncontended gigabit Ethernet switch, and the idle inter-node packet latency was recorded at around 85 μ s. The Java VM used for the server and clients was Java HotSpot(TM) 64-Bit Server VM (build 16.0-b13, mixed mode).

3.12.1 Data persistence

The speed at which data can be stored is an important metric that is required to size a cluster of machines needed for a particular application. In multi-server, concurrent systems, the usual metrics of latency and throughput can have counterintuitive characteristics as systems are scaled and load is increased.

For a baseline for performance comparisons, a batch of tests were run with a single *Frames* server, initialised with an empty database. A simulated sensor node implementation was written that repeatedly generated a batch of random samples with monotonically increasing timestamps and persisted these to the server. Each simulated client stored data against a unique source, and each sample batch was dispatch as soon as the server acknowledged the previous one. The latency distribution, along with the effective throughput in samples per second was recorded in each case.

⁹These specifications were high-end in 2009, but significantly better hardware is now available.

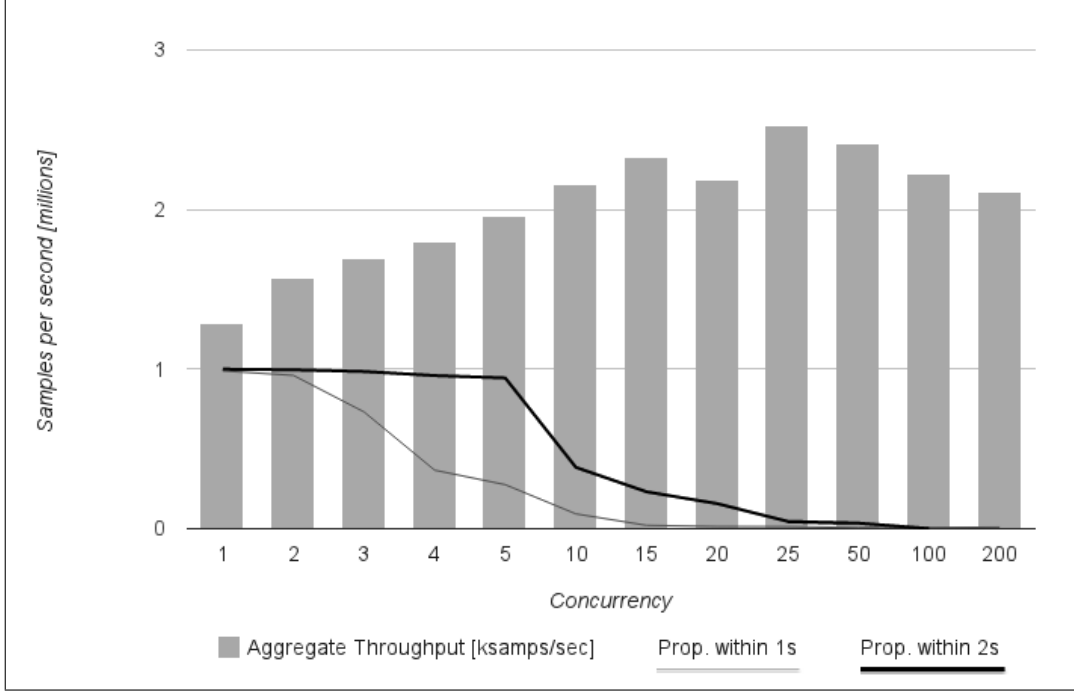


Figure 3.14: Data insert performance for a single server. Aggregate throughput across all inserting nodes is measured along with the proportion of insert operations that complete within 1 second (grey line) and 2 seconds (black line).

Figure 3.14 shows the performance characteristics of a run of tests with between 1 and 200 concurrent simulated nodes forwarding data to a single server. A single client is able to forward data at around 1.2 million samples per second (which equates to around 2.5 store operations of 500k samples per second). It can be seen that up to 5 concurrent nodes, the latency of the system does not decrease noticeably as the concurrency is increased. This is accounted for as follows: when inserting data, the server performs a small amount of CPU-bound work (indexing, validation, etc). The proportion of time this takes is around 80% of the entire operation of saving the received data. The *Xeon* processors in the servers are dual core, with hyper-threading, which means effectively 4 cores are available to the system. Therefore, before $C=5$, the limiting factor in the performance was the time taken to write the data to disk. For $C > 5$ the amount of CPU time available limited the insert performance. An additional contributing factor was likely to be the seek time of the magnetic hard disk used, but this was not measured in the tests individually.

The performance of the insert process could be improved. At present, multiple passes are made over the array of received samples for indexing and validation. If these operations were performed in a single pass, it is expected the level of concurrency at which the latency drops would be significantly higher.

The same test was repeated on a distributed cluster of 10 machines. Each simulated sensor node wrote to a unique stream as before. Figure 3.15 shows the aggregate insert performance across the entire cluster as a function of concurrency.

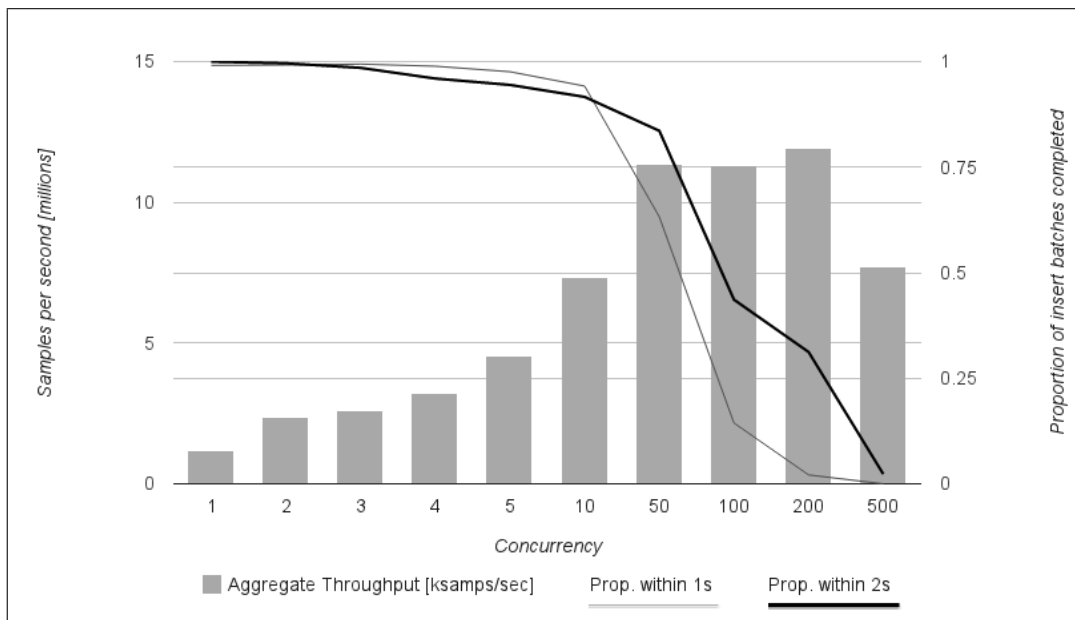


Figure 3.15: Data insert performance for a cluster of 10 servers. Aggregate throughput across all inserting nodes is measured along with the proportion of insert operations that complete within 1 second (grey line) and 2 seconds (black line).

At a concurrency of 1, the absolute performance was 1.13m samples/sec, compared to 1.30m for the single node case. In theory these should be identical, but when other nodes are available data is automatically replicated to other nodes (in this case the replication was 3¹⁰). The additional time is spent waiting for acknowledgement from the other two nodes in the replication set before the master node can commit the transaction.

¹⁰It is tempting to normalise the figures for the replication factor (which would give a performance of 3.9m samples/sec in the 10 node tests), but from the perspective of the user the replication does not represent useful throughput

In order to make a fair assessment of the 10-node case, some normalisation of the data must be performed. Table 3.2 shows the data for the 10-node tests, with the throughput normalised against the throughput recorded against a single client in this configuration.

Concurrency	Total throughput	Normalised throughput
1	1.13	1
5	4.53	4.01
10	7.32	6.48
50	11.34	10.04
100	11.42	10.11
200	11.95	10.57

Table 3.2: Normalised insertion throughput for a 10 node cluster, calculated by dividing the aggregate throughput of the cluster by the single-node throughput.

In a perfectly scalable system, it would be expected that all the values in the the **Normalised throughput** column would be equal to the cluster size N , i.e. 10. For the low-concurrency tests (where $C = 1..10$), only a subset of the servers in the cluster were involved in storing data and therefore at any one time the cluster was poorly utilised. The software is designed for clusters that are sized appropriately for the expected load, so for lower concurrency tests there was excess hardware which yielded the poor normalised throughput. At a concurrency of $C = 10$ in the table, the normalised throughput is only 65% of the expected value. This was caused by collisions of the master node allocation for each data stream. This can be seen explicitly in the graph of Figure 3.15 between $C = 2$ and $C = 3$, where the throughput does not increase as expected as the 2nd and 3rd streams were randomly placed on the same master node.

3.12.2 Query Execution Performance

A single 32 GB sample stream was generated using a random process, and inserted into a running cluster of 10 *Frames* servers that were otherwise idle, and had no other data stored in the database. Each time a query was run, the disk and application caches were flushed. The algorithm and initial conditions needed to generate this dataset are given in Appendix C. Figure 3.16 shows a down

sampled view of the data source, named `tw7`, produced by the server.

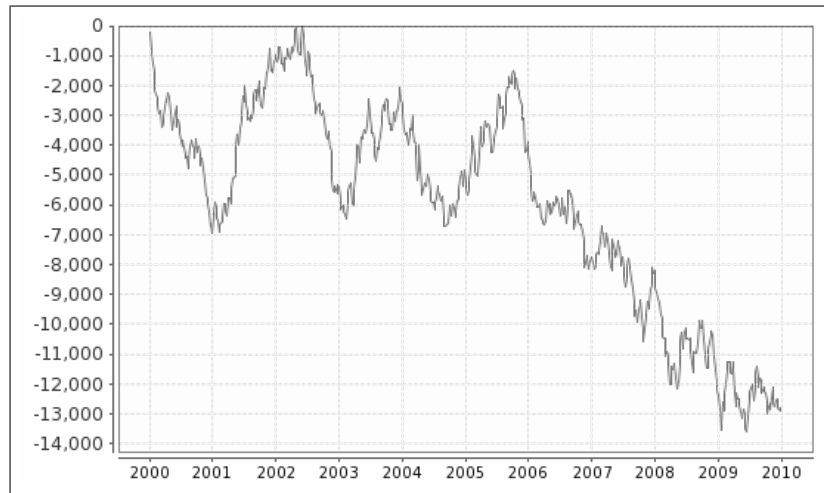


Figure 3.16: Downsampled view of 32 GB sample stream rendered by *Frames* .

To reflect real usage, in which the raw data arriving from a sensor node is likely to need calibration, the following arbitrary linear calibration was chosen:

$$100 - \text{tw7}/500$$

Calibrated this way, the range of the values in the dataset was 99.5-128.1. All operations were performed against this function rather than the raw source. The results of the tests will now be outlined.

Filter Aggregation

As outlined in previous chapters, when determining the maximum rating of a pipe, it is important to know whether the value exceeded some bound for a pre-defined minimum period, or what proportion of time was spent in certain ranges. For this, a value within 10 units of the maximum recorded value was chosen as the safe working limit. The result of the following query is the proportion of time the (calibrated) value was above this value.

```
count(100-tw7/500 > max(100-tw7/500) - 10) / count(tw7)
```

This query was run 20 times on the 32 GB dataset, and the output value was 0.2228 (i.e. 22% of the time). The worst query response time was 2.683 seconds, the average time was 1.76 seconds. Compared to a raw disk read of 32 GB at 100 MB/sec (at 327 seconds) this is a speedup of 186 times.

3.12.3 Aggregation

Accumulated static pipe stress was shown to be a contributing cause of premature failure in polyethylene pipes. Typical manufacturer ratings specify tolerances over a certain time period. The following query uses the shortcut ? operator to find any time when, in 2005, the value of the calibrated stream exceeded 110 for 2 hours:

```
(100-tw7[2004]/500) > 110 for 2h ?
```

The query was run 20 times, and the output was **true**. The maximum time taken to process the query was 0.42 seconds, the average time was 0.015 seconds. The query executed at this speed because the shortcut operator allowed the query optimiser to search the indexed blocks for ranges that match, and since an entire indexed block match the 2-hour duration, no data needed to be read from disk. If the shortcut operator is removed:

```
(100-tw7[2004]/500) > 110 for 2h
```

Then the matching time ranges are returned to the user, as shown in Figure 3.17. Note that since the exact start and end of the matching time range is needed (rather than the shortcut **true/false** response), the query now completes in around 1.2 seconds.

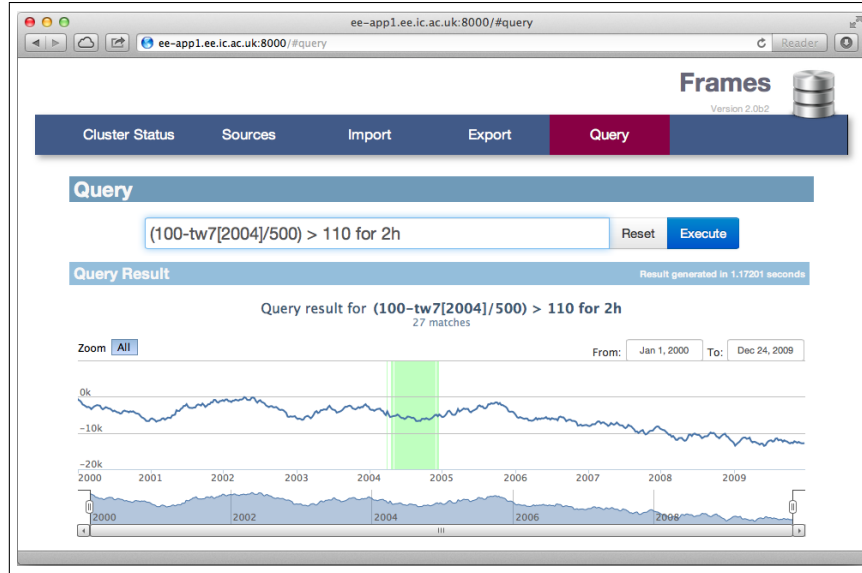


Figure 3.17: Query execution of aggregation function, showing returned time ranges from a 32 GB data stream.

Filters

In order to predict the proportion of demand that is represented during a typical time period, a user would take the average of those values within the period, and divide this by the average of values outside it, such as the following query, which calculates the ratio of increase in the average pressure during the morning period of 2006 compared with normal business hours (note that the same calibration was applied, but this is omitted in the following listing for clarity):

```
mean(tw7[2006 6am-7.15am]) / mean(tw7[2006 9am-6pm])
```

The query was run 20 times, and the output value was 0.9993 (a value close to 1 is expected since the dataset has no time dependence). The worst case execution time was 4.584 seconds, the average time was 4.30 seconds. Compared to a read of the data for 2006 in the raw dataset (at 33 seconds) this is an average speedup of 7.7 times.

3.12.4 Single-node theoretical query performance

The performance tests in this section will be used to compare the theoretical maximum performance of *Frames* against two commonly used methods of storing time-series data. The first comparison will be performed against a linear scan of a CSV file, and is used as a baseline. The second comparison is performed against a *MySQL* table with a timestamp and value column, with an index on each column and a compound index on both columns.

The dataset size is the size of the equivalent CSV file for that data. This was chosen since although CSV is not an efficient storage format, this enables the same dataset to be used for all tests, and hence a database with a more efficient format will then gain credit for the resulting increase in speed.

The set of queries used was the following, which were run in the listed order. The queries used were representative of the range of different operations, including filtering, aggregation and indexable aggregation. The name of the same stream in this example is **a**:

```
count(a > mean(a))
sum(1 - a * 2)
a > max(a) * 0.9 | count(a)
```

Dataset Size	CSV	MySQL	<i>Frames</i>
10 MB	0.23, 0.32, 0.94	0.026, 0.035, 0.329	0.0013, 0.0012, 0.0022
1 GB	22.5, 22.5, 24.8	8s, 3m 22s, 7m 43s	0.0012, 0.0019, 0.0451
800 GB	4h 50m, 4h 07m, 4h 58m ¹	(2)	0.83, 5.20, 28.9

Table 3.3: Test results (min, mean, max) for simple aggregators, comparing the specific performance of queries between *Frames* and two other traditional implementations. All times are in seconds unless otherwise specified. (1) This test was conducted with a 400 GB file and extrapolated due to size constraints on a single system. (2) This dataset exceeded the maximum table size of the *MySQL* server.

The results for the CSV file test show a predictable performance profile, and are close to the

raw speed of the disk. The results for *MySQL* show a wide variability in terms of query execution performance, which is partly due to the indexes used. *MySQL* indexes are essentially heap structures on disk, and these are able to find the extrema of values very quickly. Similarly, the joint index on the timestamp and value columns means that *MySQL* was able to optimise the time-range filtered `count` query by finding the records at either end, and then using the index to determine the number of records that lie in between. However, when the query `a > mean(a) | count(a)` was run, the performance of *MySQL* degraded significantly as it has no way to eliminate large blocks of the records from individual consideration, and so this query probably degraded to an iteration of the entire dataset. The *Frames* system was able to heavily optimise such queries by using the in-memory index block data to aggregate the data for entire frames that matched the filter without reading the raw data from disk.

3.12.5 Simulated Failure of Single Server System

In order to verify the ability of the developed system, an experimental programme was carried out where failures were simulated by either forcibly terminating the software process, or disconnecting the network cable manually, during operation. It was concluded that the network disconnection is a more accurate test of real-world failures, as even forcibly terminated tasks may be able to flush remaining network and filesystem buffers before the execution is interrupted.

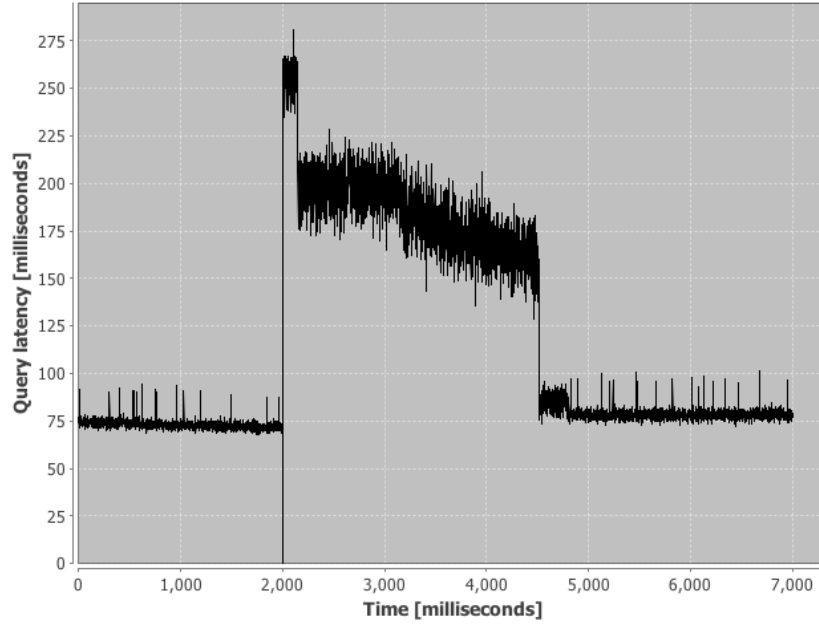


Figure 3.18: Chart showing the latencies to queries during a simulated node failure event. The simulated failure takes place at $t = 2000$.

Figure 3.18 shows the latencies that were recorded during a simulated failure experiment. The queries run were simple aggregators that required a small number of frames to be read from different nodes in the cluster. The interpretation of the results is as follows. At $t = 2000$ the network cable was unplugged to one of the nodes with data for the stream that was being used. As can be seen, there is a short period where a few requests fail (these are represented by zeros in the graph) because the master is unable to reschedule frame fetch tasks that have already been sent to the ‘failed’ node. Ideally, these requests would be retried by the master, but this is not implemented at present. There is then a brief period (around 100 ms) where the master does not decide that the node has gone down, and continues to try to contact it to serve requests, yielding significantly worse query performance. At around $t = 2150$, the node is marked as down by the master, which then initiates a recovery process whereby data from remaining nodes is re-distributed. As can be seen in the chart, this results in considerably worse query execution performance during this period, with significantly decreased determinism. At around $t = 4500$ the re-distribution is completed, and the master has atomically switched the frame location references to point to the new data location. Query execution then resumes normally from about $t = 4800$, except with slightly higher

latency. It was not immediately apparent what caused the increased latency; this was probably due to differences in fragmentation on the hard disk of the new vs. the old (failed) servers. The dataset in this case was only 100 MB, meaning that recovery was swift. Recovery for larger datasets takes a proportionally longer time, but follows a similar pattern to Figure 3.18.

Chapter 4

Analysis of Power-Electronic Interface Circuits for Piezoelectric Energy Harvesting

Inertial energy harvesters turn the mechanical motion of a proof mass into electrical energy in order to power a load circuit. There are three main transduction methods for converting the mechanical work into electrical energy: electromagnetic, electrostatic and piezoelectric. The electromagnetic approach, as commonly used in conventional energy generation schemes, has been implemented at miniature and micro-scales for energy harvesting devices by utilising finely-wound or printed coils and permanent magnets. These devices have been successful particularly at the centimetre scale and above [85, 86]. The electrostatic force has been implemented using moving-plate capacitors primed with an external source such as a battery [87], or primed with an electret [88, 89]. Piezoelectric devices include centrally loaded beams [90], and cantilevers [91].

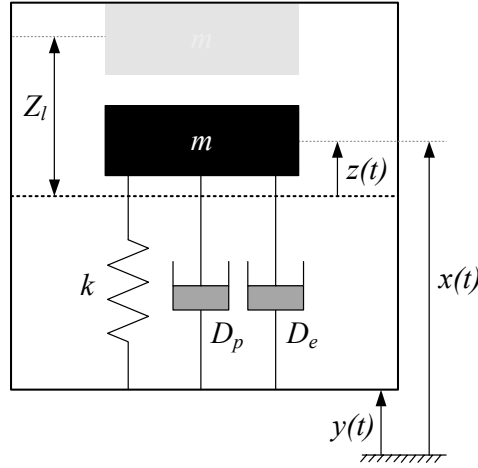


Figure 4.1: Inertial energy harvester with parasitic damping and displacement-constrained travel. D_p and D_e are the parasitic and electrical, Z_l is the internal range of motion, $y(t)$ is the instantaneous absolute position of the harvester, $x(t)$ is the instantaneous absolute position of the proof mass m , and $z(t)$ is the instantaneous position of the proof mass relative to the harvester.

For the inertial energy harvester device shown in Figure 4.1 to operate optimally (*i.e.* with maximum possible power density), the damping force presented by the transducer must be optimised [36]. As an example, when operating the generator at resonance, the electrical damping force D_e should be made equal to the parasitic (mechanical) damping force D_p , or to a level which just limits the travel range of the proof mass to $\pm Z_l$ (the maximum displacement amplitude in which the mass may travel before hitting the end stops), whichever is the greater. Under many operating conditions and transducer designs, the required damping forces cannot readily be reached using conventional

load circuits (resistors or bridge rectifiers) connected to piezoelectric transducers. This is primarily because of the modest electromechanical coupling coefficients achievable [21, 51].

If the amplitude Y_o of the external displacement is small compared to the internal travel range of the mass, Z_l , then the system can operate with a high Q-factor, i.e. a low total damping, before the end stop limits are reached. In such cases, if the parasitic damping is low, high Q operation will maximise power output, and this will require low electrical damping; consequently a high piezoelectric coupling coefficient is not required. However, these conditions are generally found only for high-frequency applications, and it is increasingly recognised that most practical applications of energy harvesters provide vibration at modest frequencies, in the range 1 - 100 Hz. Low-amplitude displacement operation also presents a practical limitation in some circuits in that the open-circuit voltage must be large enough to overcome the diode on-state voltage drops. Whatever electrical damping is realised results from providing real power to the electrical load connected to the transducer. If a simple resistive or rectifying load is used, the maximum damping for a piezoelectric device is small.

4.1 Voltage Profile Modification

The key to increasing the amount of power from any vibration-driven energy harvesting device is to maximise the force-distance integral, i.e. the work done by the mechanical source of motion. Various load circuits have been proposed and demonstrated which aim to increase the power that can be extracted from a piezoelectric device, and these essentially function by placing additional charge on the piezoelectric element in a synchronous fashion, which acts to increase the force that the transducer presents to the mechanical system. The static mechanical force produced by the piezoelectric element is proportional to the charge on the device. If this voltage is controlled such that the force produced opposes the direction of mechanical motion then the mechanical vibration source must do more work to move the piezoelectric element the same distance, and by increasing this voltage to the limit of the electrical breakdown of the device the output power is maximised. Therefore the optimal voltage profile is a square wave that changes polarity when the piezoelectric device reaches each extremity of motion.

In [22], Shenck observed for the first time that by modifying the charge profile on the piezoelectric element, rather than optimising the value of a purely resistive load, an increase in power could be obtained. A circuit is presented that synchronously extracts the full charge from the piezoelectric capacitance, such that at the start of the next half-cycle the current source is not required to waste energy discharging the capacitance, resulting in a net increase in power output.

The synchronous extraction circuit of Shenck does not achieve the optimal voltage profile, although it does better than a purely resistive load. Figure 4.2 compares three voltage waveforms for a piezoelectric harvester under sinusoidal excitation. It can be seen that compared to the purely resistive load, the synchronous extraction circuit achieves a voltage profile that is closer to the ideal. Whereas the mechanical vibration source spends part of the cycle discharging the capacitance, and hence wasting energy, the sign of the voltage waveform of the synchronous extraction is always the same as the optimal profile.

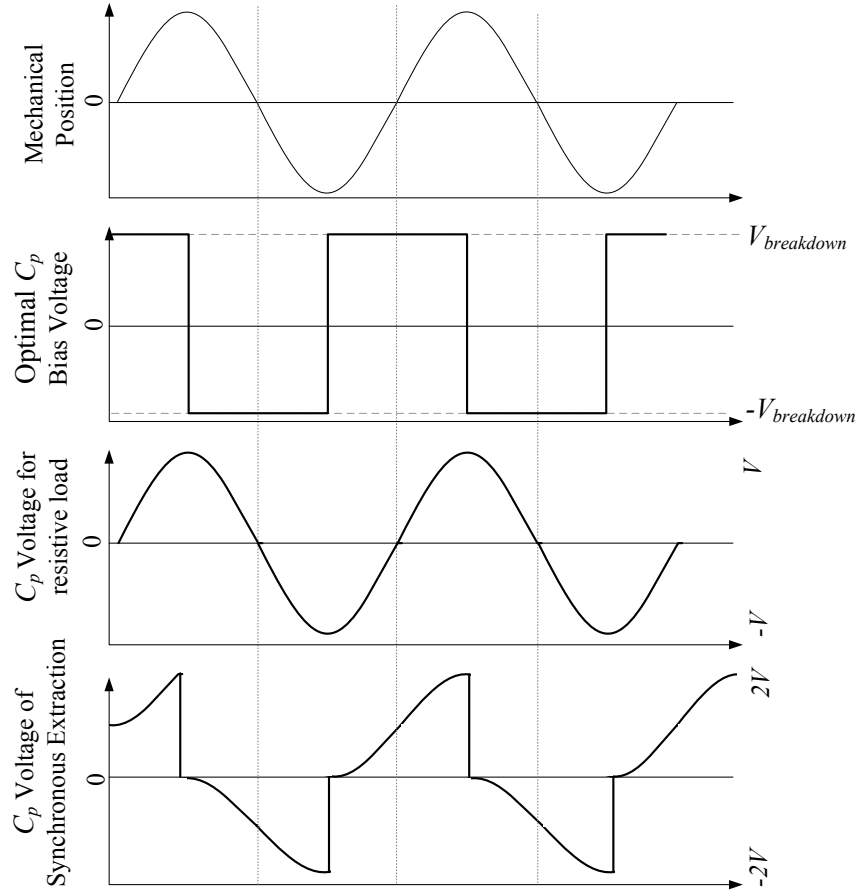


Figure 4.2: Comparison sketch of piezoelectric capacitance voltage profiles; 1) reference mechanical position (nominal amplitude), 2) optimal voltage profile in order to maximise damping and hence energy generation, 3) voltage profile of purely resistive load, and 4) voltage profile of Shenck's synchronous extraction circuit.

4.2 Modelling of the Electromechanical System

The force presented by the piezoelectric transducer to the mechanical system is influenced by the impedance and operation of the electrical circuit that is connected to it. The transducer interface circuit can therefore be designed to allow modification of the damping force, and different types of circuit are capable of modifying the electrical damping by different amounts. A simple equivalent circuit model of a mass-spring-damper microgenerator damped by a piezoelectric element with a

purely resistive load is shown in Figure 4.3 [92].

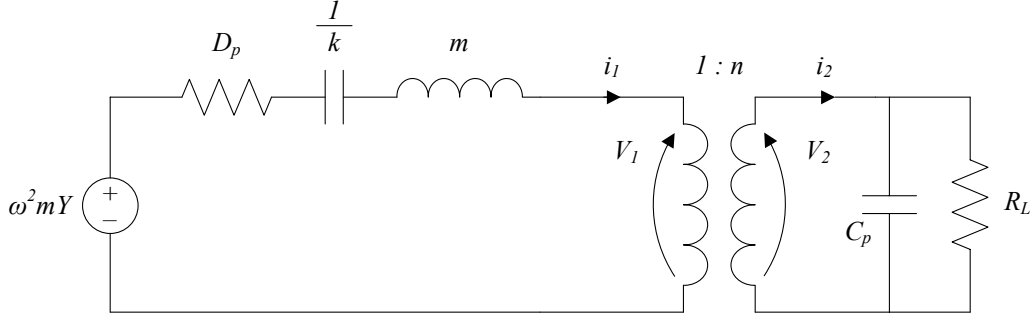


Figure 4.3: Microgenerator with piezoelectric transduction.

In Figure 4.3, the circuit on the primary side of the transformer represents the mechanical system (voltages correspond to forces and currents correspond to velocities [93]). The transformer represents the piezoelectric transduction mechanism and the secondary side circuit represents the electrical load and properties of the piezoelectric transducer, including its in-built capacitance C_p . In this case, an external load resistor is attached to the transducer. Due to the relatively low coupling coefficient of most piezoelectric materials, the transformer is a step-up device with a high turns ratio. This has the effect that the combined impedance of C_p and R_L , when referred to the primary side, is very low, so that V_1 is low and effectively a small mechanical force is presented by the load. Thus the current in the primary, corresponding to proof mass relative velocity $\dot{z}(t)$, is largely unaffected by connections made on the secondary side. The transformer can therefore be approximated by a current-controlled current source, and Figure 4.3 redrawn as the circuit shown in Figure 4.4.

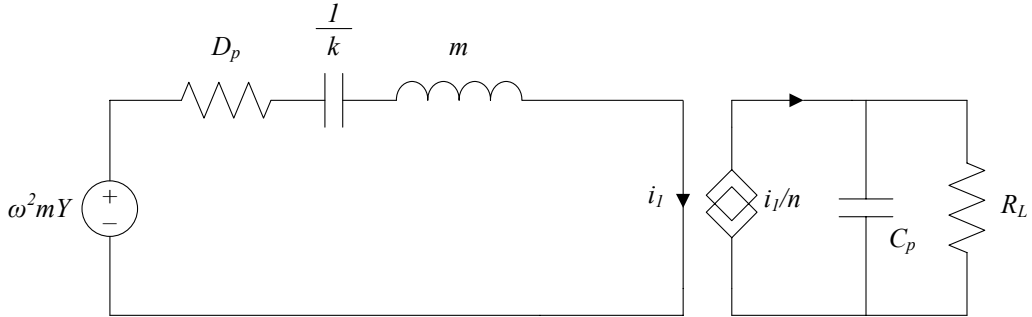


Figure 4.4: Microgenerator equivalent circuit with low electromechanical coupling.

Therefore, a piezoelectric energy harvester with a level of electromechanical coupling low enough that any level of damping from the electrical side makes negligible difference to the velocity of the proof mass can be simplified to the circuit of Figure 4.5.

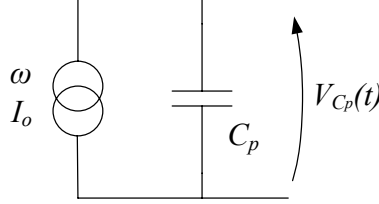


Figure 4.5: Simplified piezoelectric model. The open-circuit amplitude of $V_{C_p}(t)$ is V_{po} .

Where sophisticated circuits are able to overcome the low damping limitation described above, the approximation that I_o is unaffected by the electrical load will begin to break down. However, we will neglect this limitation in the analysis that follows, for purposes of simplicity and clarity. For a wide range of useful applications, the output power can be improved substantially by modified circuits before the change to I_o becomes significant. Furthermore, even where I_o is significantly reduced by the enhanced damping, the relative performance of the various circuits - all referenced to I_o - will be unaffected. The task of the interface circuit is therefore to extract the maximum power from a sinusoidal current source with an inherent shunt capacitance.

4.3 Methodology

Before discussing the candidate interface circuits, we must first place some bounds on the analysis that is to follow. The circuits analysed have many degrees of freedom, including diode on-state voltage drop V_D , inductor Q factor ($\omega L/R_s$ where R_s is the series resistance), the on-state drain-to-source resistance of the MOSFETs $R_{DS_{on}}$, the size of the piezoelectric element, and the mechanical excitation amplitude and frequency of the harvester. Typically the power generated in the piezoelectric can be increased and the losses in the circuit reduced by allocating additional volume to each constituent part of the harvester system. As there is generally a constraint on total harvester volume, optimally choosing all of the component values within the volume constraint is a complex task. Fully optimising the system in a holistic way across all of these variables for each candidate

circuit is beyond the scope of this thesis; however, sensible component values must be chosen to allow fair comparisons between the circuits to be made.

The power consumed by control circuitry has also not been considered. This control usually involves maxima/minima detection, gate drives and some simple timing, and so its power consumption is similar across the range of architectures studied. This will affect the relative performance of circuits with and without such a requirement, particularly at very low output power levels, although not the comparison of the circuits' performance in terms of the Q factor of their inductive current paths.

In the case that diode voltage drops are either negligible or are ignored, it is found that the power output of each circuit can be normalised to a factor of $I_o^2/\omega C_p$ and written solely as a function of the Q-factor of the current paths containing inductors:

$$Q = \frac{1}{R} \sqrt{\frac{L}{C}} \quad (4.1)$$

This allows a comparison between the performance of all circuits to be made which holds regardless of excitation characteristics and the volume of piezoelectric material used. However, when diode drops are taken into account, such normalisation is not possible due to the non-linearity making a first-order expression in Q unavailable. For these comparisons, component values were taken from the best available off-the-shelf products at the time of writing for three chosen sizes of energy harvester. The configurations of harvester and component parameters used are summarised in the Table 4.1. The on-state resistance of any switches is assumed negligible compared to the series resistances of inductors, and the parasitic capacitance is negligible compared to C_p , which is realistic for existing off-the-shelf parts.

Table 4.1: Component parameters for harvester configurations.

Config.	Vol.	R_s	L	C_p	V_{Don}
1	1 cm ³	0.47 Ω	470 μ H	65 nF	0.5 V
2	3 cm ³	0.141 Ω	1.41 mH	585 nF	0.4 V
3	10 cm ³	47 m Ω	4.7 mH	6.5 μ F	0.3 V

The inductors were chosen from a range of products available to maximise the Q factor of the current paths in each given volume, and the diodes are Schottky diodes. For simplicity, all diodes are modelled as an ideal rectifier in series with an ideal voltage source. Where a specific value of

this voltage source is needed for plotting graphs, it is chosen from Table 4.1 above.

To validate this model of a diode, a simulation was conducted against a ZC5800 diode model in *PSpice* with the last circuit presented in this work, the Single-Supply Pre-Biasing circuit. The typical on-state voltage of this component at the peak current of 100 mA is around 0.28 V, so the ZC5800 was compared to a 0.28 V voltage source and ideal diode. It was found that the total discrepancy of the final power output from the circuit was less than 1 % for a range of different operating conditions. This is due to the fact that compared to some applications, the piezoelectric discharge currents are relatively high (in the 1-100 mA range), and therefore the diode was always forced either completely into conduction or completely blocking - the diodes spent almost no time conducting at the lower end of the conduction region. Additionally, since the frequency of operation of the charge and discharge circuitry (where the diodes are used) is low (10-100 Hz) the effect of junction capacitance, which typically dominates at high frequencies, is minimal.

Of the various circuits analysed, some have the capability to push energy into a storage element such as a capacitor in order to provide a stable DC voltage for a load, while others provide a time-varying voltage. This important difference in functionality is taken into account when final comparisons of the circuits are made.

All of the circuits are analysed here within the same analytical framework to allow fair comparisons to be made between them. Where circuits have previously been investigated by others, the analysis is briefly repeated here, but with particular attention to such factors as the effect of diode drops for circuits with low piezoelectric output voltages. All of the derived closed-form solutions have been verified against *PSpice* time-domain simulations and the results agree within a few percent.

4.4 Purely Resistive Load

The simplest load that can extract real power from the piezoelectric transducer is a resistive load as shown in Figure 4.6.

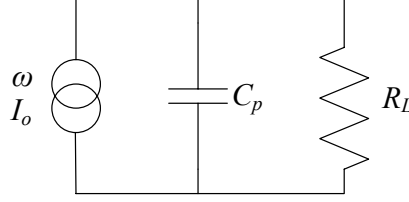


Figure 4.6: Piezoelectric energy harvester connected to a purely resistive load.

Under excitation by a sinusoidal current source with a magnitude I_o , a resistive load of magnitude R_L will dissipate a power P_{R_L} , given by $P_{R_L} = I^2 R_L$:

$$P_{R_L} = \frac{1}{2} \left(\frac{I_o^2}{1 + R_L^2 \omega^2 C_p^2} \right) R_L \quad (4.2)$$

By differentiating (4.2) with respect to R_L , the optimal value that maximises P_{R_L} is found to be $1/\omega C_p$, at which the circuit has an output power of:

$$P_{ORL} = \frac{1}{4} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.3)$$

This value of P_{ORL} will be used as a base case for comparison with the other circuits.

4.5 Tuned-Out Shunt Capacitance

An obvious approach to extract the maximum power from the current source in Figure 4.5 is to add a shunt inductor to tune out the piezoelectric capacitance. If this is done then the circuit looks like an ideal voltage source. However, this option is not realistically achievable because of the very large inductor values required. For example, in an energy harvesting application with a typical vibration frequency of 100 Hz, and a piezoelectric capacitance of 100 nF, the required resonant inductor would be about 25 H. Consequently, this option cannot be exploited in practice, and is not considered further here.

4.6 Rectified DC Load

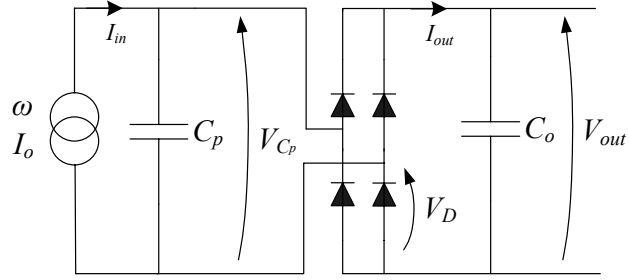


Figure 4.7: Full-bridge rectifier with output smoothing capacitor.

A more useful output circuit for piezoelectric microgenerators is a standard bridge rectifier with a smoothing capacitor (Figure 4.7). The circuit's application in energy harvesting was originally analysed in [94] where the diodes were assumed to be ideal. Here we analyse the circuit as having diodes with a fixed on-state voltage drop (V_D). This circuit provides useful additional functionality over the circuit in Figure 4.6 as almost all electronic loads require a DC input. The output capacitance C_o will typically be much larger than the piezoelectric capacitance C_p , and so its voltage can be approximated as constant over a cycle. The current waveform I_{out} will be discontinuous, and only non-zero when the magnitude of the voltage on the piezoelectric capacitance V_{C_p} exceeds $V_{out} + 2V_D$. The voltage waveform on the piezoelectric capacitance in relation to the piezoelectric current source is shown in Figure 4.8:

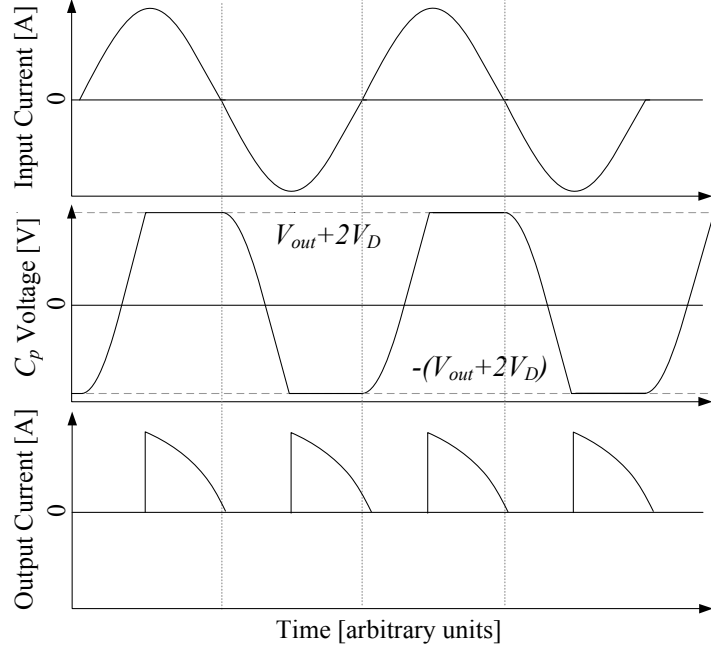


Figure 4.8: Scale sketch of piezoelectric input current, piezoelectric capacitance voltage and output current (labeled I_{out} in Figure 4.7) in the presence of a bridge rectifier with a fixed output voltage V_{out} .

It is useful to introduce a quantity V_{po} which is the open-circuit voltage magnitude of the piezoelectric capacitance as shown in Figure 4.5. This is given by:

$$V_{po} = \frac{I_o}{\omega C_p} \quad (4.4)$$

By finding the angle at which conduction into the output stage begins, the power output of the circuit as a function of the output voltage V_{out} can be determined:

$$P_{out} = \frac{2}{\pi} I_o V_{out} \left[1 - \frac{V_{out} + 2V_D}{V_{po}} \right] \quad (4.5)$$

The optimal output voltage of this circuit, found by differentiation of (4.5) with respect to V_{out} , is:

$$V_{out_{opt}} = \frac{1}{2}(V_{po} - 2V_D) \quad (4.6)$$

The maximum power is then found by substituting (4.6) into (4.5), as:

$$\begin{aligned} P_{max} &= \frac{(I_o - 2V_D\omega C_p)^2}{2\pi\omega C_p} \\ &= \frac{1}{2\pi} \left(1 - \frac{2V_D}{V_{po}}\right)^2 \left(\frac{I_o^2}{\omega C_p}\right) \end{aligned} \quad (4.7)$$

Unless the diodes are replaced with synchronous rectifiers, the circuit only produces power if the change in voltage due to the current source is high enough to cause current to flow to the output stage, that is $V_{po} > V_{out} + 2V_D$. Therefore to extract power from the circuit at all as $V_{out} \rightarrow 0$ we require that:

$$V_{po} > 2V_D \quad (4.8)$$

In the ideal case that the diode drop is negligible, the power output simplifies to $P_{max} = I_o^2/2\pi\omega C_p$, which is less than the optimal resistive load case (P_{ORL}) by:

$$\frac{P_{RECTDC}}{P_{ORL}} = \frac{2}{\pi} \approx 0.64 \quad (4.9)$$

Clearly, when the diode drops are included, this ratio is even lower. Thus, whilst this circuit is more useful than the optimal resistive load for practical applications, there is a trade-off in that less power can be converted.

4.7 Synchronous Switched Extraction

A third circuit, originally proposed by Shenck [22], is shown in Figure 4.9. Instead of a direct connection to the piezoelectric device, the resistive load is connected via a switch that closes when the voltage on the capacitor is maximal in either polarity (i.e. at the zero crossings of the current source). Assuming that there is no leakage through the switch while it is open, at that instant all the energy on the capacitor is transferred to the load, and the process is repeated in the negative half cycle as shown in Figure 4.10. Note that in this circuit the switch must be capable of conducting and blocking in both directions and therefore cannot be a single MOSFET. However, the effect could be achieved with a pair of series MOSFETs with suitable gate drives. We assume from now on that all switches have this capability, and will hence model them as perfect switches.

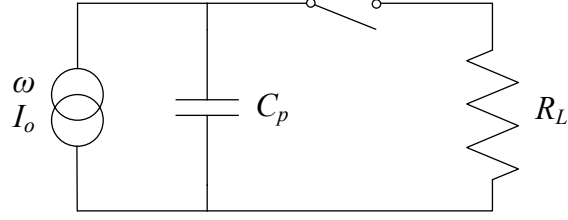


Figure 4.9: Synchronous switched extraction circuit.

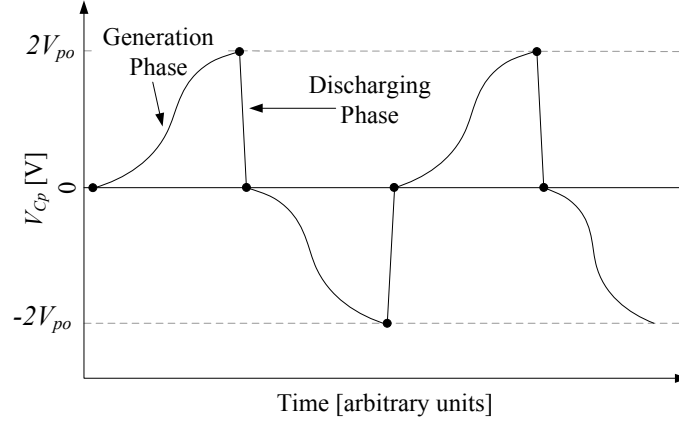


Figure 4.10: Conceptual sketch of the voltage waveform on the piezoelectric capacitance C_p for the circuit of Figure 4.9. The length of the discharge phase is exaggerated for illustration; in practice the discharge will be relatively instantaneous in most applications provided $RC \ll 1/\omega$.

A sinusoidal current source of amplitude I_o and frequency ω produces a charge per half cycle $Q_{\frac{1}{2}} = 2I_o/\omega$. This charge, placed on C_p , has an energy:

$$E = \frac{1}{2} \frac{Q^2}{C_p} = \frac{2}{C_p} \left(\frac{I_o}{\omega} \right)^2 \quad (4.10)$$

The power dissipated in a resistive load R_L is found by multiplying this expression by $2f$:

$$\begin{aligned} P_{max} &= 2f \left(\frac{2I_o^2}{C_p \omega^2} \right) \\ &= \frac{2}{\pi} \left(\frac{I_o^2}{\omega C_p} \right) \end{aligned} \quad (4.11)$$

Compared to the base-case optimal resistive load circuit of Figure 4.6 this represents an improvement in power generation, for a modest increase in complexity, by a simple ratio of:

$$\frac{P_{SSE}}{P_{ORL}} = \frac{8}{\pi} \approx 2.55 \quad (4.12)$$

4.8 Synchronous Switched Extraction with DC Output

As we have seen from the circuit in Figure 4.9, there is an advantage of a factor of 2.55 over the optimal resistive load case when extracting energy from the circuit at zero crossings of the current source.

As previously discussed, a DC output is required to power a typical electronic load. A new synchronous switched circuit is therefore proposed (Figure 4.11) with a rectifier and smoothing capacitor. This circuit combines the synchronous switched extraction circuit of Shenck with an inductor and free-wheeling output stage to efficiently charge an output voltage source, and aims to achieve the same energy gain of 2.55 while providing a DC output. A 5th free-wheel diode is added to the standard bridge-rectifier configuration to provide a small efficiency gain in the free-wheeling stage by lowering the on-state voltage drop.

The output stage in this case is the output filter of a buck switch-mode power supply, which allows energy to be transferred efficiently from C_p to the output capacitor C_o , which again behaves as a constant-voltage source.

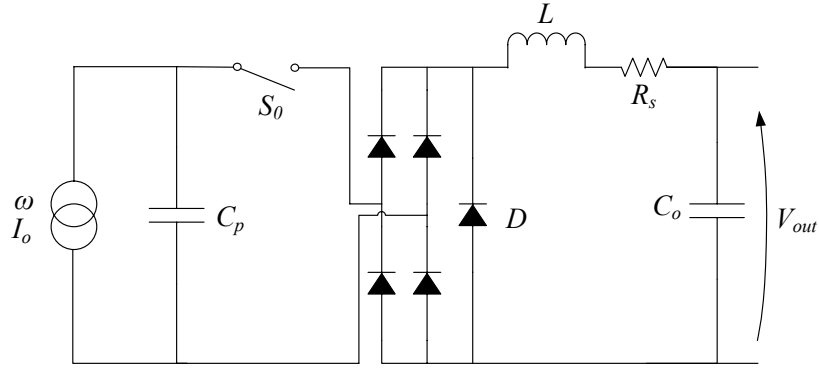


Figure 4.11: Circuit for synchronous switched extraction with DC output.

4.8.1 Optimal energy transfer

In any circuit that transfers energy between a capacitor and a voltage source through an inductor, such as the $C_p \rightarrow S_0 \rightarrow L \rightarrow C_o$ current path of Figure 4.11 (with C_o being modelled as a voltage

source), there can be one or two parts to the energy transfer process. The first part is the transfer of energy through the path between the capacitor, inductor and voltage source, and the second part is when the capacitor is fully discharged and the inductor current then free-wheels (in this case though D). If the value of the voltage source (given the initial voltage on the capacitor) is carefully chosen, all the energy will have been removed from the capacitor at the same point as the inductor current falls to zero, meaning the free-wheeling stage does not occur. If the current path resistance is zero, then the initial voltage on the capacitor, $V_{C_p}(0)$, must be exactly twice the value of the voltage source in order for it to fully discharge. If the voltage on the capacitor is higher than this, then free-wheeling will occur. If it is lower then a voltage will remain on the capacitor (approximately $2V_{out} - V_{C_p}(0)$) when the current has dropped to zero.

It can be shown that in the presence of losses, maximum energy is delivered to the voltage source if the voltages are such that the circuit operates at the limit of the onset of free-wheeling, and this is the strategy chosen for the circuits analysed here. Figure 4.12 shows the results of a simulation of the normalised output power vs. V_{out} , and the contributions to this power from free-wheeling and direct conduction.

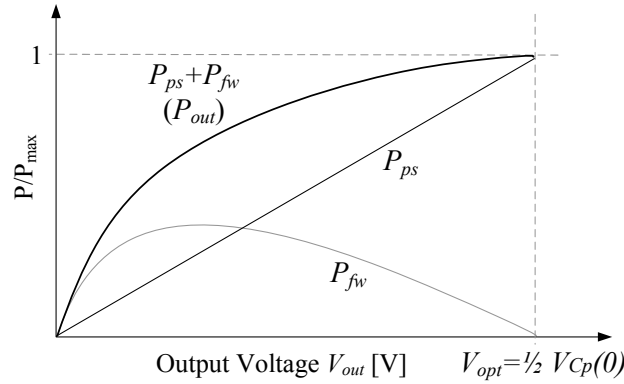


Figure 4.12: Contributions to total power output (P_{out}) from free-wheeling (P_{fw}) and direct conduction (P_{ps}), and their sum, as a function of output voltage, for the circuit of Figure 4.11. At $V_{out} = V_{opt}$ the contribution from free-wheeling is zero. Graph plotted in MAPLE 14.01.

4.8.2 Detailed analysis of synchronous switched extraction

The circuit of Figure 4.11 will now be analysed. At the start of the cycle (shown in Figure 4.13), the current source drives charge into the capacitor C_p with switch S_0 open. As in the previous synchronous extraction case, S_0 is closed at the point when the voltage on C_p is at its maximum. This causes current to flow through the full-bridge rectifier and inductor to the output capacitor. If the circuit is operating on the limit of freewheeling occurring, the voltage on C_p will be clamped at V_D at the end of the extraction process. It is worth noting that were D not present, the voltage after energy extraction would instead be $2V_D$.

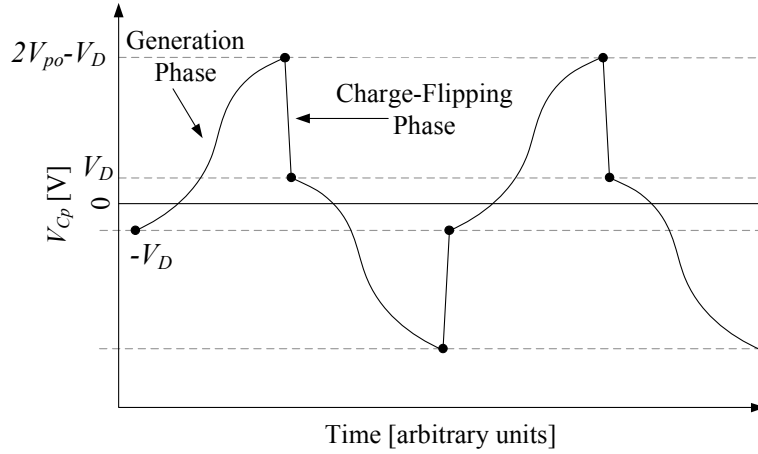


Figure 4.13: Conceptual sketch of the voltage waveform on the piezoelectric capacitance, with the main voltages highlighted: the voltage at the start of generation, $\pm V_D$, and the voltage at the start of charge-flipping, $\pm(2V_{po} - V_D)$.

The inductor is modelled as having inductance L with series resistance R_s , and the capacitors are assumed to be free of leakage. It is also again assumed that the output voltage is constant and is thus modelled as a voltage source. The main current path is a series RLC circuit with two diodes as shown in Figure 4.14.

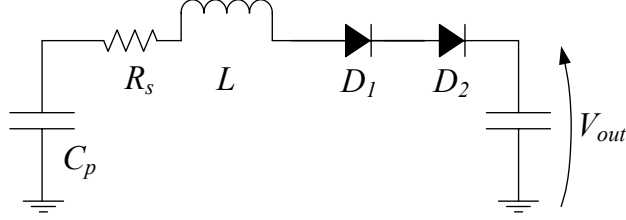


Figure 4.14: Resonant current path for charge extraction.

The Q factor of the current path does not depend on the diodes as these are assumed to have a constant voltage drop; the power loss due to the diodes is accounted for by a lowering of the effective voltage magnitude over which the oscillation occurs. The Q factor is therefore the same as a series RLC circuit:

$$Q = \frac{1}{R_s} \sqrt{\frac{L}{C_p}} \quad (4.13)$$

The damped resonant frequency of the path is given by the standard result:

$$\omega_n = \sqrt{\frac{1}{LC_p} - \frac{R_s^2}{4L^2}} \quad (4.14)$$

Given that D clamps C_p at V_D at the end of the charge-flipping phase, the voltage at the end of the piezoelectric generation phase is $2V_{po} - V_D$. The period for which the switch should conduct is half a cycle of the damped natural frequency ω_n (*i.e.* π/ω_n), the end of which coincides with a zero value of inductor current. The fraction of voltage magnitude conserved on the capacitor of an RLC oscillator with Q factor Q after one half cycle of operation is found by taking the exponential envelope of the damped sinusoid and evaluating it at π/ω_n , or one half-cycle:

$$\gamma = e^{-\frac{\pi}{2Q}} \quad (4.15)$$

Therefore the final voltage on the piezoelectric capacitance C_p will be:

$$V_{final} = (V_{out} + 2V_D) - \gamma [(2V_{po} - V_D) - (V_{out} + 2V_D)] \quad (4.16)$$

As explained in Subsec. 4.8.1, in order to remove as much energy as possible from the piezoelectric capacitance we require that in a resonant half cycle, enough energy is removed from C_p to reduce the voltage to exactly V_D . This implies that (4.16) puts an upper bound on the value of V_{out} of:

$$V_{out} \leq \frac{(2V_{po} - 3V_D)\gamma - V_D}{1 + \gamma} \quad (4.17)$$

If V_{out} is greater than this value, then it is impossible to extract all of the energy stored in C_p though the action of the resonant circuit. Assuming V_{out} is set exactly as per the limit of the inequality in (4.17), then the energy output from the circuit per half cycle is:

$$E_{\frac{1}{2}} = V_{out}C_p(2V_{po} - 2V_D) \quad (4.18)$$

By approximating γ as close to 1 and hence taking a truncated series expansion $\gamma \approx 1 - \pi/2Q$, and with V_{out} set to the maximum value permitted by (4.17), this expression multiplied by $2f$, and after some rearrangement, gives the power output from the circuit:

$$P_{out} \approx \frac{2}{\pi} \left(\frac{I_o^2}{\omega C_p} \right) \left[\left(1 - \frac{\pi}{4Q} \right) - 3 \frac{V_D}{V_{po}} \left(1 - \frac{\pi}{6Q} \right) + 2 \left(\frac{V_D}{V_{po}} \right)^2 \left(1 - \frac{\pi}{8Q} \right) \right] \quad (4.19)$$

This expression indicates clearly the loss of power from two factors - the finite Q of the inductor, and the diode voltage drop. If both factors are neglected, the output power is the same as the synchronous switched extraction into a resistive load of Figure 4.9. Without neglecting losses, to obtain any power in the circuit of Figure 4.11 we require that $V_{po} > V_D$.

4.9 Piezoelectric Fixed Voltage Control

The circuits analysed so far achieve different performance in terms of power output ultimately because the piezoelectric current source pushes charge into different average voltages. To maximise the energy generated, the circuit must therefore optimise this voltage profile, towards the optimal profile shown in Figure 4.2. We can now see that it is intuitively reasonable that the rectifier circuit should perform the poorest, the directly connected resistor is better and the synchronous circuits perform the best. The rectifier circuit clamps the peak voltage on the piezoelectric capacitance so that the current source never operates into the maximum voltage, while the synchronous circuits maintain all of the charge on the piezoelectric capacitance until the voltage peaks.

Expanding this concept, various circuits have been developed that increase the output voltage into which the generated charge is supplied by adding charge to the device at suitable points in the motion cycle. All these circuits require synchronous switching of some type, and thus have some monitoring and control overhead.

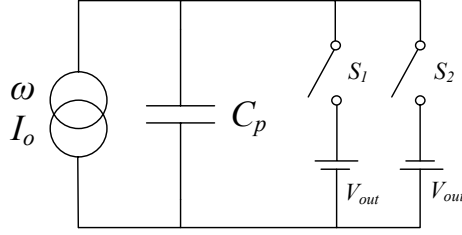


Figure 4.15: Circuit diagram of piezoelectric fixed voltage control.

A simple example is shown in Figure 4.15, which was originally introduced for applications in structural damping [95], where the inefficiencies due to a lack of inductors in current paths are not a concern. In this circuit the generator is connected to one of the two output voltage sources continuously, by either S_1 or S_2 , except for two very short periods per cycle when the current source has a zero crossing. At that moment, the polarity of V_{out} is near-instantaneously switched using S_1 and S_2 so that the current source is always charging a supply, as shown in Figure 4.16.

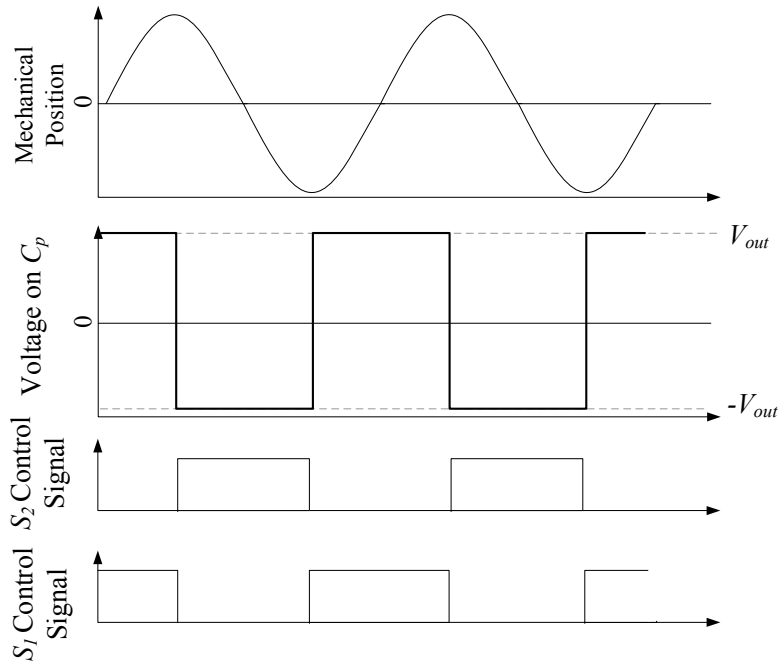


Figure 4.16: Operational sketch of the piezoelectric fixed voltage control circuit, including mechanical motion (top), C_p voltage (middle) and control signals for each switch (bottom).

Since the charge produced by the generator during each half cycle is $2I_o/\omega$, the energy supplied per cycle is $4V_{out}I_o/\omega$. However, each polarity reversal at the current zero crossings (corresponding to a displacement maximum) drains $2C_pV_{out}$ of charge from the supply, reducing the net energy gain per cycle by $4C_pV_{out}^2$. The value of V_{out} that optimises the net power is $I_o/2\omega C_p$, giving a net average power of:

$$P_{max} = \frac{1}{\pi} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.20)$$

This is worse than the optimal resistive load by a factor of $2/\pi$.

An improvement to the circuit is proposed here by dividing the polarity reversal into two stages (Figure 4.17): discharge through an additional switch S_o , followed by charging to the opposite polarity.

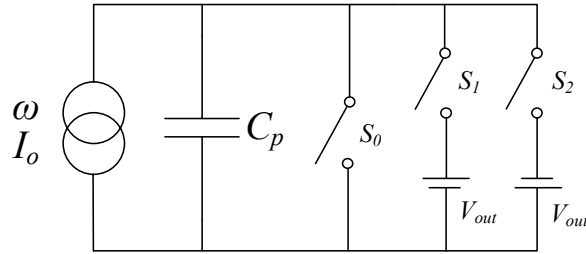


Figure 4.17: Piezoelectric fixed voltage control (with charge cancelling).

This reduces the lost energy by half, and doubles the net power and the optimal V_{out} . The operational waveform is visibly the same as that of Figure 4.16, except that between the transition between the switch control signals, an additional signal momentarily closes S_o .

This modification improves the power output by a factor of two, to $4/\pi$ times better than the optimal resistive load case. However, this is still low compared to the synchronous switched output circuits discussed above. To gain a power advantage over these previous synchronous circuits, it is necessary to apply the extra charge to the piezoelectric device in as lossless a manner as possible, which is the purpose of the class of circuits that will be described next.

4.10 Parallel SSHI with Resistive Load

The SSHI (Synchronous, Switched Harvesting with Inductor) technique reported by Guyomar et al. in [96] introduced the key advance over Shenck's circuit of using a switched inductor to flip the charge on the capacitor twice per cycle. Since the extra charge does not have to be drawn from an external supply, the losses can be minimal - limited primarily by the finite Q of the path containing the inductor.

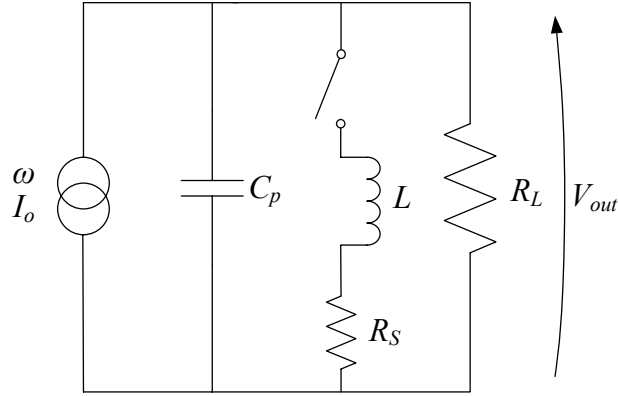


Figure 4.18: SSHI with resistive load.

Figure 4.18 shows an SSHI circuit with a resistive load, with the inductor losses occurring in R_s . The output voltage V_{out} is shown in Figure 4.19. Gradually the energy stored in the capacitor C_p builds, increasing the voltage, until the power output increases to a point where the system reaches steady state.

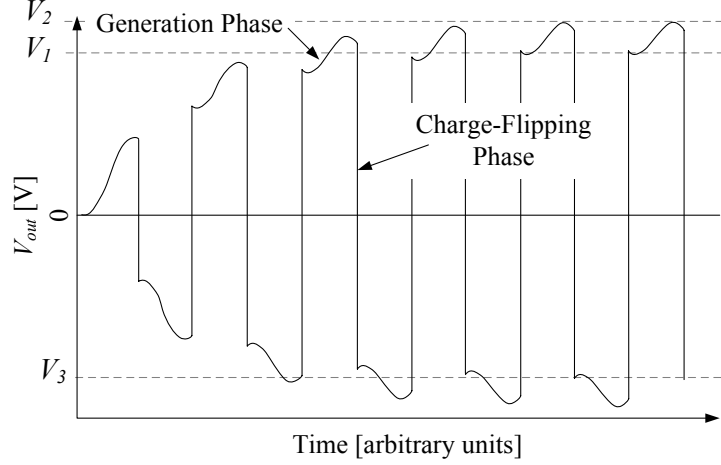


Figure 4.19: Scale sketch of the voltage waveform of SSHI, showing build-up to steady-state operation.

There are two phases of operation:

1. The generation phase, during which the current source charges C_p and the load resistor removes some charge. The voltage on C_p goes from $V_1 \rightarrow V_2$.
2. The charge-flipping phase, where the charge stored on C_p is flipped in magnitude through the inductor, and the voltage on C_p goes from $V_2 \rightarrow V_3$. This phase occurs over a very short time period.

After the system is started from rest the output magnitude gradually rises until steady state is reached at $V_3 = -V_1$ as shown in Figure 4.19.

4.10.1 Generation phase $V_1 \rightarrow V_2$

During the charging phase some energy is dissipated by the load R_L . The total charge passing through the load is:

$$q_{R_L} = I_{R_L} t \approx \frac{V_1}{R_L} \frac{\pi}{\omega} \quad (4.21)$$

The net charge supplied to C_p is then $2I_o/\omega - q_{R_L}$, and the change in voltage is simply this divided by C_p , giving:

$$V_2 = V_1 \left(1 - \frac{\pi}{\omega R_L C_p} \right) + 2V_{po} \quad (4.22)$$

4.10.2 Charge-flipping phase $V_2 \rightarrow V_3$

The charge-flipping phase will be much shorter than the generation phase, so that the energy dissipated by the load resistor R_L during that time can be neglected. The switch is closed for a time π/ω_n (where ω_n is given in (4.14)), *i.e.* until the current in the inductor returns to zero. Using the definition of γ in (4.15) the voltage after flipping from a voltage V_2 is a function of the Q factor of the RLC circuit as follows:

$$V_3 = -V_2\gamma \quad (4.23)$$

4.10.3 Steady state

Solving (4.22) and (4.23) for the case where $V_3 = -V_1$ yields an expression for the steady-state voltage V_{1ss} :

$$V_{1ss} = \frac{2V_{po}}{1/\gamma - 1 + \pi/\omega R_L C_p} \quad (4.24)$$

Approximating the output power as V_{1ss}^2/R_L , the optimal load can be derived as:

$$R_{Lopt} = \frac{\pi}{\omega C_p (1/\gamma - 1)} \quad (4.25)$$

Using the approximation $\gamma \approx 1 - \pi/2Q$, the output power is:

$$P_{max} = \frac{\gamma}{\pi(1-\gamma)} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.26)$$

$$\approx \frac{2Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.27)$$

Compared to the optimal resistive load this is an improvement of:

$$\frac{P_{P_SSH\!I_RL}}{P_{ORL}} = \frac{1}{\pi} \frac{\gamma}{1-\gamma} \quad (4.28)$$

$$\approx \frac{8Q}{\pi^2} \quad (4.29)$$

Thus even a modest Q provides a significant power gain. What is now needed is a method which realises the benefits of this circuit but which also gives a DC output.

4.11 Parallel SSHI with DC Output

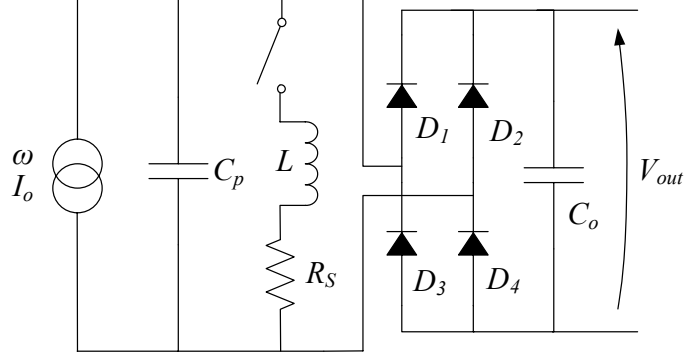


Figure 4.20: Parallel SSHI with DC output.

Fortunately, the parallel SSHI circuit can be straightforwardly adapted for a constant DC output with the addition of rectification as shown in Figure 4.20. This circuit was originally presented in [97], but was analysed in more detail in [98], and is analysed here for consistency with our analytical framework. The rectifier and output capacitor clamp the output voltage of the piezoelectric element to $\pm(V_{out} + 2V_D)$. As in the previous circuit, the switch is closed briefly at the zero crossings of the piezoelectric current to reverse the polarity on C_p . The voltage waveform on C_p is shown in Figure 4.21.

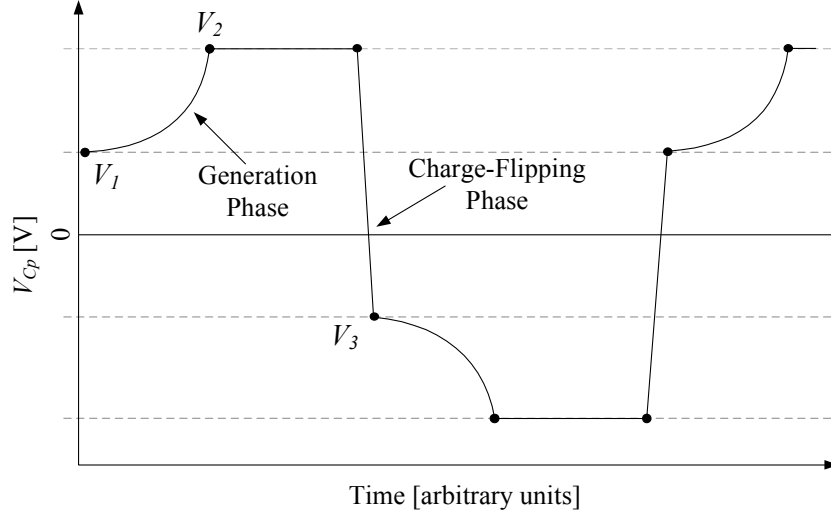


Figure 4.21: Conceptual sketch of the voltage waveform on the piezoelectric capacitance C_p for the circuit of Figure 4.20.

As can be seen, the piezoelectric voltage is clamped at a maximum of V_2 . After charge flipping this gives $V_3 = -\gamma V_2$, which in steady state equals $-V_1$ so that $V_1 = \gamma V_2$. To bring the voltage back to the clamped level in order to deliver charge to the output, some charge must be placed on C_p to compensate for the inefficiency in the flipping process. Thus:

$$V_2 - \gamma V_2 = \frac{1}{C_p} \int_0^\tau I_o \sin(\omega t) dt \quad (4.30)$$

where τ is the time to reach the clamped voltage. From (4.30) we can obtain:

$$\cos(\omega\tau) = 1 - \frac{V_2}{V_{po}}(1 - \gamma) \quad (4.31)$$

The energy supplied to the output per half cycle is:

$$\begin{aligned} E_{\frac{1}{2}} &= I_o V_{out} \int_\tau^{\pi/\omega} \sin(\omega t) dt \\ &= C_p V_{po} V_{out} \left[2 - \frac{(1 - \gamma)(V_{out} + 2V_D)}{V_{po}} \right] \end{aligned} \quad (4.32)$$

This energy is maximised for an output voltage of:

$$V_{out} = \frac{V_{po}}{1 - \gamma} - V_D \quad (4.33)$$

for which the output power is given by:

$$P_{max} \approx \left[1 - \frac{\pi}{2Q} \frac{V_D}{V_{po}}\right]^2 \left(\frac{2Q}{\pi^2}\right) \left(\frac{I_o^2}{\omega C_p}\right) \quad (4.34)$$

This equation shows a key advantage of the circuit, which is that the open-circuit output voltage V_{po} does not need to exceed the diode drop voltages, the minimum instead being V_D times $\pi/2Q$. If V_{po} is substantially greater than this minimum, so that the first factor in (4.34) can be neglected, then the output power is $8Q/\pi^2$ times higher than in the optimal resistive load case. For the high-Q case, where the diode drop can be neglected and $\gamma \approx 1 - \pi/2Q$, the optimum output voltage $V_{out} \approx (2Q/\pi)V_{po}$. Since a sinusoidal current source of amplitude I_o supplies an average current (when rectified) of $2I_o/\pi$, this implies an average power from the source of:

$$\begin{aligned} P_{source} &\approx V_{out}(2I_o/\pi) \\ &\approx \frac{4Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p}\right) \end{aligned} \quad (4.35)$$

Since the power supplied to the output is half this level, clearly the optimised circuit is at best 50% efficient, as would be expected from an impedance-matched source-load pair.

4.12 Series SSHI with Resistive Load

An alternative implementation of the synchronous switched harvester is to place the switching inductor in series with the load, as shown in Figure 4.22. This circuit was reported in [23], and analysed again in [99] in the context of structural damping.

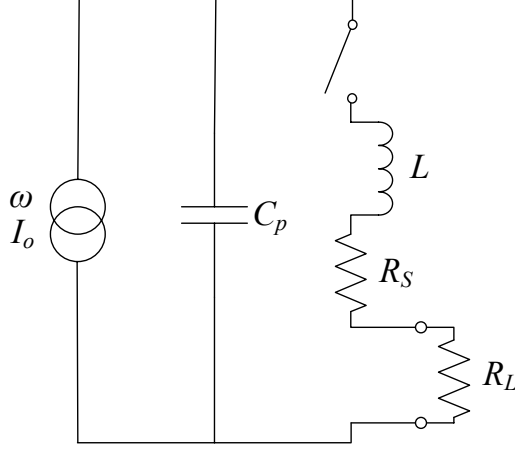


Figure 4.22: Series SSHI with resistive load.

Again, the switch is briefly closed at the maxima of displacement to reverse the polarity of the voltage on C_p , but in this case power is only extracted into the load during this charge-flipping phase. The waveform on C_p will be the same as in Figure 4.19, with a transient period until a steady state is reached. The loss of voltage due to dissipation in the charge reversal equals the voltage increase provided by the source, as was the case in the previous charge-flipping circuits.

For this circuit, during the generation phase current is only supplied to C_p , not to the load, so that (using the previous notation) $V_2(1-\gamma') = 2V_{po}$. Here γ' is the efficiency of the RLC circuit as before, but where the Q of the charge-flipping circuit now includes the load resistance R_L , *i.e.* $\gamma' = e^{\pi/2Q'}$ and:

$$Q' = \frac{1}{R_s + R_L} \sqrt{\frac{L}{C_p}} \quad (4.36)$$

The energy dissipated in the two resistors is therefore:

$$\begin{aligned} E_{loss} &= \frac{1}{2}C_p V_2^2 - \frac{1}{2}C_p V_1^2 \\ &= \frac{1}{2}C_p (2V_{po})^2 \frac{1+\gamma'}{1-\gamma'} \\ &\approx \frac{8Q'}{\pi} C_p V_{po}^2 \end{aligned} \quad (4.37)$$

This energy will divide between the two resistors in proportion to their magnitude since both conduct the same current, so the load energy per half cycle is:

$$E_{\frac{1}{2}} \approx \left(\frac{R_L}{R_L + R_s} \right) \left(\frac{8Q'}{\pi} \right) C_p V_{po}^2 \quad (4.38)$$

It is straightforward to show that this is maximised for $R_L = R_s$, for which $Q' = Q/2$, giving a maximum power of:

$$\begin{aligned} P_{max} &= \frac{\omega}{\pi} E_{\frac{1}{2}} \\ &= \frac{2Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \end{aligned} \quad (4.39)$$

which is the same as in the parallel SSHI case of Figure 4.18.

4.13 Series SSHI with DC Output

The Series SSHI circuit can also be adapted for a fixed DC output by adding rectification, as shown in Figure 4.23. In this circuit, proposed by Lefeuvre et al. [100], the charge flipping and energy extraction again occur simultaneously. This circuit is similar in operation to the DC Synchronous Extraction circuit of Figure 4.11, except that in this case there is no possibility of free-wheeling due to the placement of the switch in series with the inductor. Therefore the switch must be opened at zero-crossings of inductor current, so instead of discharging the capacitor to zero each half cycle as with the DC Synchronous Extraction circuit, the charge on the capacitor is flipped resonantly through the output stage as shown in Figure 4.24.

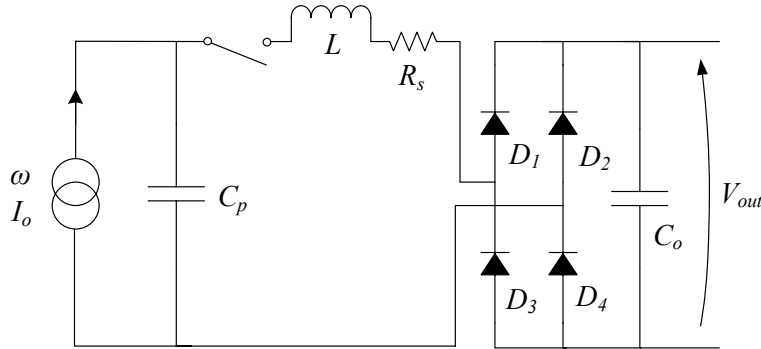


Figure 4.23: Series SSHI with rectified DC load.

A variation on this circuit presented in [101, 102] uses a transformer instead of an inductor to decrease the effective on-state voltage drop of the diodes. It is also possible to modify the output voltage by the addition of a further flyback output stage as described in [103]. This allows the first

capacitor to remain at the optimal voltage for operation of the SSHI technique, whilst giving a chosen output voltage, and this does not modify the upper limit on output power for this circuit. A further study of the switching duty cycle required to maintain an optimal voltage on the storage capacitor is presented in [104].

When the switch is closed, charge flows through either D_1 and D_4 or D_2 and D_3 , and the output capacitor C_o , being in series with these, has charge added to it at a voltage V_{out} . We again assume C_o is large enough so that V_{out} does not vary during charge flipping.

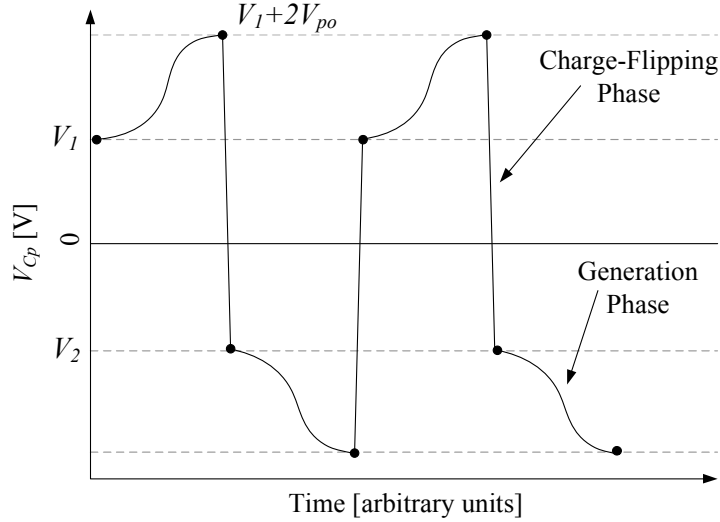


Figure 4.24: Sketch of the steady-state operating cycle of Series SSHI circuit.

The waveform on C_p for this circuit (Figure 4.24) has the same form as in the previous cases. We define V_1 as the voltage at the beginning of the generation phase, with $V_1 + 2V_{po}$ the voltage just before charge flipping. At this point the voltage across the inductor is the difference between $V_1 + 2V_{po}$ and $V_{out} + 2V_D$ and it is this voltage whose polarity is reversed, with an efficiency γ , so that the voltage V_2 after flipping is:

$$V_2 = (V_{out} + 2V_D) - \gamma((V_1 + 2V_{po}) - (V_{out} + 2V_D)) \quad (4.40)$$

The steady-state value of V_1 is found by setting $V_2 = -V_1$:

$$V_{1ss} = \frac{2V_{po}}{1 - \gamma} - (V_{out} + 2V_D) \left(\frac{1 + \gamma}{1 - \gamma} \right) \quad (4.41)$$

While it appears from (4.41) that V_{1ss} can be negative, the need for $V_1 + 2V_{po}$ to be greater than $V_{out} + 2V_D$ introduces the simple requirement that the open-circuit piezoelectric voltage V_{po} must be greater than $2V_D$ for the circuit to function. This requirement can be a significant limitation for energy harvesting applications where the motion amplitude is low.

The energy added to the output capacitor on each half cycle of motion is given by the voltage change on C_p during flipping times C_p (to give the charge transferred), times V_{out} . Using the high-Q approximation $\gamma \approx 1 - \pi/2Q$, this gives:

$$E_{\frac{1}{2}} \approx \frac{8Q}{\pi} C_p V_{out} [V_{po} - (V_{out} + 2V_D)] \quad (4.42)$$

From this the optimum output voltage simplifies to:

$$V_{out_{opt}} = \frac{V_{po}}{2} - V_D \quad (4.43)$$

The output power is then $(\omega/\pi)E_{\frac{1}{2}}$, *i.e.*:

$$P_{max} \approx \left[1 - \left(\frac{V_D}{V_{po}} \right)^2 \right] \frac{2Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.44)$$

This gives the same value as the parallel SSHI circuits if $V_{po} \gg V_D$. A variation of this circuit is proposed in [105] with only one diode in the conduction path, so V_D is halved compared to the Series SSHI case.

4.14 Parallel SSHI with Synchronous Extraction

The previous circuit combined the advantages of charge flipping for increased damping, synchronous charge extraction and a DC output. However, a disadvantage is that in order for it to operate, V_{po} must be greater than $2V_D$. A new circuit is now proposed, called Parallel SSHI with synchronous extraction (Figure 4.25), which aims to retain all the advantages of the previous circuit and at the same time overcome the minimum requirement on V_{po} . This circuit introduces additional complexity in that there are now two control variables: the output voltage, V_{out} and the proportion of the stored energy on C_p to extract during each half cycle.

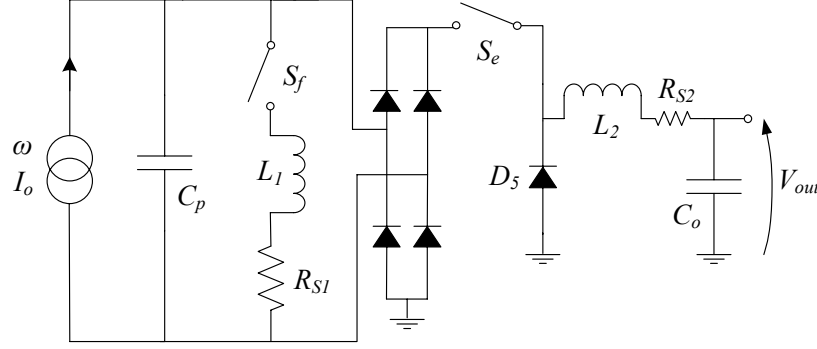


Figure 4.25: Parallel SSHI circuit with synchronous extraction.

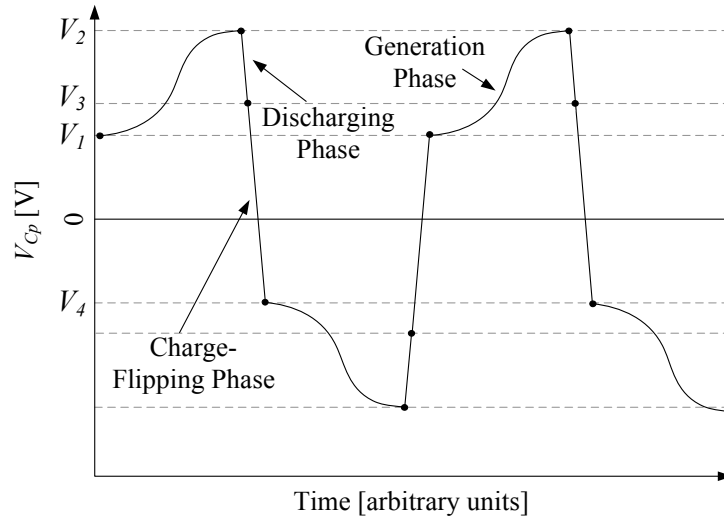


Figure 4.26: Sketch of the voltage on the piezoelectric capacitance in steady-state of the Parallel SSHI with Synchronous Extraction circuit.

We define V_1 as the voltage on C_p just after charge flipping has occurred, and V_2 as the voltage after generation has taken place, as shown in Figure 4.26, with $V_2 = V_1 + 2V_{po}$. We now let α_e be the control parameter representing the proportion of the voltage on the capacitor C_p left after extraction through S_e (taking values between 0 and 1), and V_3 is the voltage on C_p after the discharge takes place, such that $V_3 = \alpha_e V_2$. Let α_f be the proportion of the voltage magnitude left after the charge flipping takes place, so the magnitude of the voltage after the flipping $V_4 = -\alpha_f V_3$. In steady-state operation $V_4 = -V_1$. Solving these expressions at $V_1 = -V_4$ we find that the steady-state condition

of V_1 is:

$$V_{1ss} = 2V_{po} \frac{\alpha_e \alpha_f}{1 - \alpha_e \alpha_f} \quad (4.45)$$

4.14.1 Charge-flipping stage

The circuit has two Q factors: Q_f (of the charge-flipping circuit) and Q_e (of the extraction circuit). The value of α_f in (4.45) is based only on the Q factor of the inductor of the flipping path Q_f , such that $\alpha_f = e^{-\frac{\pi}{2Q_f}} = \gamma$. The damped resonant frequency of this path is:

$$\omega_{n_f} = \sqrt{\frac{1}{L_1 C_p} - \frac{R_{s1}^2}{4L_1^2}} \quad (4.46)$$

Therefore S_f should be closed for a time π/ω_{n_f} .

4.14.2 Optimal discharging voltage ratio α_e

The proportion of voltage left after discharging (α_e) is a control parameter, set by the discharge switch opening when the capacitor C_p voltage (starting from V_2) has fallen to that proportion of its initial value (*i.e* V_3). When this happens there will be a current in the inductor, which will free-wheel into the output power supply V_{out} . To find the optimal amount of energy to extract each cycle we can ignore the efficiency of the output stage since the energy extracted is not affected. The change in energy per half cycle on the capacitor during discharging is:

$$E_{\frac{1}{2}} = \frac{1}{2} C_p V_2^2 - \frac{1}{2} C_p V_3^2 \quad (4.47)$$

$$= (1 - \alpha_e^2) \frac{1}{2} C_p V_2^2 \quad (4.48)$$

Substituting V_2 for its steady-state value found from (4.45), and differentiating with respect to α_e we find that this expression is maximised when $\alpha_e = \alpha_f$.

4.14.3 Free-wheel energy recovery

The differential equation describing the voltages along the free-wheel current path is:

$$L_2 \frac{di(t)}{dt} + R_{s2} i(t) + V_D + V_{out} = 0 \quad (4.49)$$

Solving this to find an expression for the inductor current with respect to time $i(t)$ gives:

$$i(t) = \left(i(0) + \frac{V_D + V_{out}}{R_s} \right) e^{-\frac{R_s}{L}t} - \frac{V_D + V_{out}}{R_s} \quad (4.50)$$

The energy transferred to the power supply while the current is positive (it cannot go negative due to the presence of the diode) is given by the integral:

$$E_{fw} = V_{out} \int_0^{t|i=0} i_{fw}(t) dt = \frac{L_2 V_{out}}{R_s^2} i_{fw_o} R_s + \frac{L V_{out}}{R_s^2} (V_{out} + V_D) \ln \left(\frac{V_{out} + V_D}{i_{fw_o} R_s + V_{out} + V_D} \right) \quad (4.51)$$

Where i_{fw_o} is the current at the start of free-wheeling.

4.14.4 Maximising power output

The energy from the capacitor C_p into the power supply during the discharge conduction (non-free-wheeling) stage is:

$$E_{out} = V_{out} \int_0^{t_{S_e}} i_{discharge} dt \quad (4.52)$$

$$= 2V_{po} \frac{(1 - \alpha_e) C_p V_{out}}{1 - \alpha_e \alpha_f} \quad (4.53)$$

Substituting $\alpha_e = \alpha_f$ for maximum power extraction, and replacing V_2 with the steady-state value from (4.45), this gives a power output of:

$$P_{out} = I_o V_{out} \left(\frac{2}{\pi} \right) \frac{\alpha_f - 1}{\alpha_f^2 - 1} + \frac{\omega}{2\pi} E_{fw} \quad (4.54)$$

The final voltage on the piezoelectric capacitance after discharging is:

$$V_{final} = (V_{out} + 2V_D) - \gamma (V_2 - (V_{out} + 2V_D)) \quad (4.55)$$

The power output of this circuit is maximised when V_{out} is set to a value such that $V_{final} = \alpha_e V_2$ and no free-wheeling occurs, *i.e* the switch S_e is closed for exactly one half cycle. Substituting V_2 for the steady-state value found using (4.45), and solving for V_{out} gives:

$$V_{out_{opt}} = 4V_{po} \frac{\gamma}{(1 - \gamma^2)(1 + \gamma)} - 2V_D \quad (4.56)$$

Substituting this into (4.53) gives a maximum power output of:

$$P_{max} = \frac{8}{\pi} \frac{\gamma}{(1+\gamma)^2(1-\gamma^2)} \left(\frac{I_o^2}{\omega C_p} \right) - \frac{4}{\pi} \frac{V_D I_o}{1+\gamma} \quad (4.57)$$

Using the expansion $\gamma \approx 1 - \pi/2Q$ this expression simplifies to:

$$P_{max} \approx \left[1 - \frac{\pi V_D}{Q V_{po}} \right] \frac{2Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (4.58)$$

i.e the same as for the previous circuit.

In the next chapter, a new class of circuits, invented by the author, Pre-Biasing will be studied and compared to the existing circuits reviewed here. A full comparison of all the circuits studied, including the preferred version of the Pre-Biasing circuit called Single-Supply Pre-Biasing can be found at the end of the next chapter.

Chapter 5

Improved Power Output by Pre-Biasing

The circuits of Chapter 4 achieve an increase in performance by synchronously applying charge and then removing it from the piezoelectric transducer, so that the voltage on the capacitance is modified while the external motion moves the device and thus drives charge into the circuit. The benefit from these techniques is that they all, to varying degrees, achieve a piezoelectric voltage profile that is closer to the ideal profile of Figure 4.2, i.e. a square wave with the opposite polarity to the direction of motion.

The pre-biasing circuit (Fig. 5.1), first proposed by this author in [106], is a generalisation of a number of the techniques analysed so far. The setting of the piezoelectric voltage and energy extraction occur as two independent, fully controllable phases.

5.1 Pre-Biasing

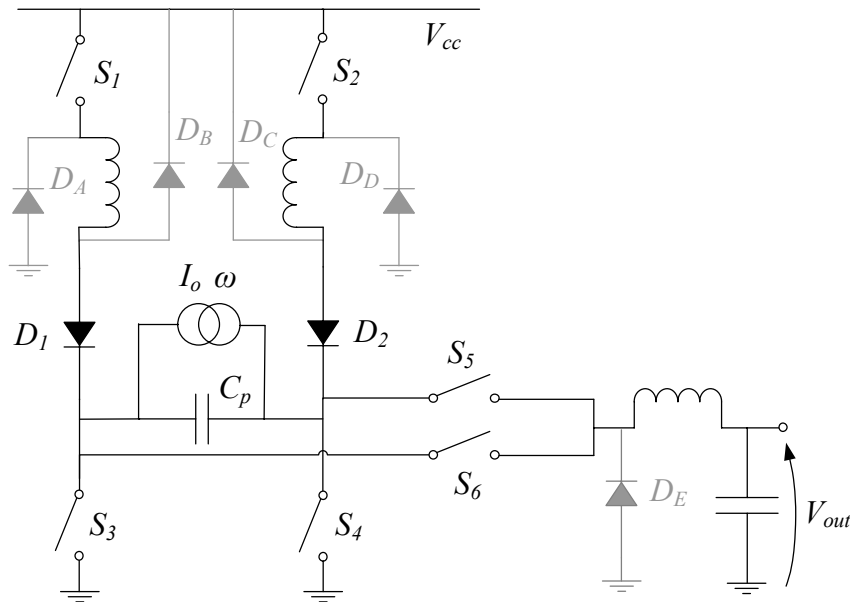


Figure 5.1: Pre-biasing circuit diagram, showing optional circuit components for non-optimal output voltage operation (D_A, D_B, D_C, D_D and D_E) in grey.

The circuit is comprised of a modified H-bridge, which allows charging of the piezoelectric capacitance in either direction, and a synchronous buck output stage, which allows the extraction of

an arbitrary amount of energy from C_p . Free-wheeling paths exist to return energy to the power supply for operating modes where the switches must open when there is still current in inductors. A sketch of the voltage on the piezoelectric capacitance during operation is shown in Fig. 5.2. The exact voltage placed on the piezoelectric capacitance can be controlled, as can the exact amount of energy removed each cycle by the discharge circuit.

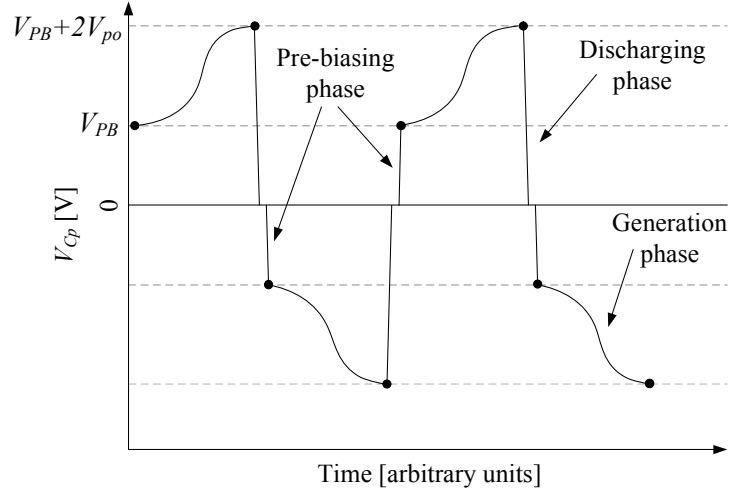


Figure 5.2: Basic sketch of the voltage waveform on the piezoelectric capacitance for the circuit of Fig. 5.1.

Figure 5.3 shows a more accurate simulated operation of the circuit over a few mechanical cycles. The four required control signals, comprised of two extract pulses and two charge pulses, are shown, with the current from the internal piezoelectric current source and the effect on the piezoelectric voltage.

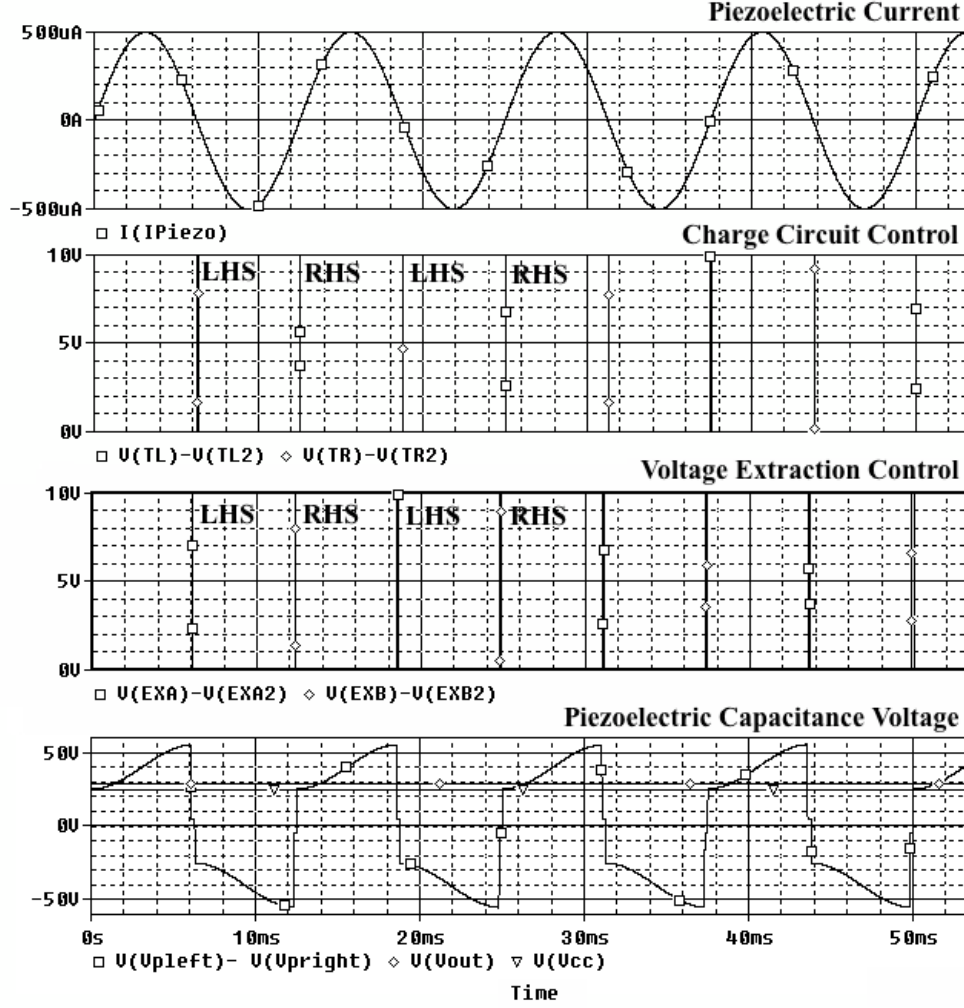


Figure 5.3: Simulated operation over a few mechanical cycles of the circuit of Fig. 5.1, generated with *ORCAD PSpice* version 16.3. The LHS and RHS control signals refer to the closing of switch pairs S_1, S_4 and S_2, S_3 respectively. The LHS and RHS extraction control signals refer to the closing of switches S_6 and S_5 respectively. Main circuit parameters: $f=80$ Hz, $L=5$ mH, inductor series resistance $R_L=3$ Ω , $C_p=65$ nF.

Previously this circuit was analysed as a function of the overall efficiency that could be achieved in charging and discharging the piezoelectric capacitance. Here a detailed analysis is performed for the first time in terms of the component values and hence Q-factor.

There are two possible modes of operation for this circuit:

1. **Optimal voltage control** - the power supply (V_{cc}) and output (V_{out}) voltages are set according to the optimal operating condition - *i.e* energy transfer occurs without free-wheeling occurring. Therefore the diodes $D_{[A-E]}$ are not used to conduct free-wheeling currents, although they can be retained to deal with any free-wheeling current that results from timing inaccuracy. Pre-bias voltages can be in the range $\approx 0 - 2V_{cc}$.
2. **Custom output voltage** - a non-optimal regime that allows an arbitrary output voltage to be set by allowing energy transfer through free-wheeling paths. Diodes $D_{[A-E]}$ are therefore required to conduct free-wheeling currents. Pre-bias voltage is limited to V_{cc} because of the clamping of D_B and D_C .

For consistency with the analysis of the previous circuits we will only consider the optimal operating regime for this circuit, *i.e* that which we term optimal voltage control.

The circuit operates in three phases. First the pre-bias is applied by closing either S_1 and S_4 , or S_2 and S_3 until $V(C_p) = V_{PB}$. Then the switches are opened and the piezoelectric capacitance floats, gaining more charge from the current source I_o , when motion occurs, until it is at a voltage $V_{PB} + 2V_{po}$. Then the entire charge is removed by the closing of S_5 and S_3 or S_6 and S_4 , after which $V(C_p) = 0$. The process then repeats in the negative half-cycle.

5.1.1 Discharging Phase

Using the previous notation, the voltage on C_p after discharging is:

$$V_2 = V_{out} - \gamma(V_{PB} + 2V_{po} - V_{out}) \quad (5.1)$$

As previously discussed, this final voltage must be zero, with no free-wheeling occurring, and so V_{out} is set to:

$$V_{opt} = \frac{(V_{PB} + 2V_{po})\gamma}{1 + \gamma} \quad (5.2)$$

By substituting (5.2) into the energy delivered to the power supply, which is the amount of charge transferred times the change in voltage $V_{out}C_p(V_{PB} + 2V_{po})$, the energy into the output per half cycle is:

$$E_{\frac{1}{2}} = C_p(V_{PB} + 2V_{po})^2 \frac{\gamma}{1 + \gamma} \quad (5.3)$$

So the discharge efficiency η_d can be written as:

$$\eta_d = 2 \frac{\gamma}{(1 + \gamma)} \quad (5.4)$$

5.1.2 Charging Phase

If the piezoelectric capacitance starts at 0 V (having been perfectly discharged) then the pre-bias voltage after half a resonant cycle (lasting π/ω_n) is:

$$V_{PB} = (V_{cc} - V_D)(\gamma + 1) \quad (5.5)$$

Since V_{PB} is a control parameter, the required V_{cc} to achieve a given V_{PB} is:

$$V_{cc} = \frac{V_{PB}}{\gamma + 1} + V_D \quad (5.6)$$

The energy expended by a power supply at V_{cc} charging a capacitor to a voltage of V_{PB} in a half resonant cycle is:

$$E_{ps} = V_{cc} V_{PB} C_p \quad (5.7)$$

$$= V_{PB} C_p \left(\frac{V_{PB}}{\gamma + 1} + V_D \right) \quad (5.8)$$

Therefore we can write the charge efficiency η_c as:

$$\eta_c = \frac{\frac{1}{2} C_p V_{PB}^2}{E_{ps}} \quad (5.9)$$

$$\approx \frac{1}{1 + \frac{\pi}{4Q} + \frac{2V_D}{V_{PB}}} \quad (5.10)$$

From (5.7) and (5.3), the per-cycle energy output is:

$$E_{\frac{1}{2}} = C_p (V_{PB} + 2V_{po})^2 \frac{\gamma}{1 + \gamma} - C_p V_{PB} \left(\frac{V_{PB}}{\gamma + 1} + V_D \right) \quad (5.11)$$

From this expression we find the optimal pre-bias voltage to be:

$$V_{PB_{opt}} = \frac{2\gamma V_{po} - \frac{1}{2} V_D (1 + \gamma)}{1 - \gamma} \quad (5.12)$$

Therefore the maximum power output from this circuit is:

$$P_{max} = \frac{\omega C_p}{\pi} \left[\frac{\gamma}{1 - \gamma^2} 4V_{po}^2 \left(1 - \frac{1 + \gamma}{2} \frac{V_D}{V_{po}} \right) - \frac{V_D^2}{4} \frac{1 + \gamma}{1 - \gamma} \right] \quad (5.13)$$

If V_D is set to 0 this becomes:

$$P_{max}|_{V_D=0} = \frac{4}{\pi} \frac{\gamma}{1 - \gamma^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.14)$$

$$\approx \frac{4Q}{\pi^2} \left(1 - \frac{\pi}{4Q} \right) \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.15)$$

Compared to the optimal resistive load this is a significant gain of:

$$\frac{P_{PB}}{P_{ORL}} = \frac{16}{\pi} \frac{\gamma}{1 - \gamma^2} \quad (5.16)$$

$$\approx \frac{16Q}{\pi^2} \left(1 - \frac{\pi}{4Q} \right) \quad (5.17)$$

This increased power output is arguably offset somewhat by the considerable complexity of the circuit due to the high number of components required, and the control overhead (which is not included) will reduce the figure obtained.

5.2 Single-Supply Pre-Biasing

For a given Q factor the previously analysed pre-biasing circuit exhibits the highest power output of those presented in chapter 4. However, the circuit requires three inductors and six switches, making a high Q factor harder to realise in that circuit than one with fewer inductors. An embodiment of the pre-biasing method presented in [107] called single-supply pre-biasing (Fig. 5.4), invented by the author, will now be analysed, which reduces the component count to four switches and one inductor. In addition, the existence of diode drops in the previously proposed circuits causes a reduction in efficiency when the generated voltage on the piezoelectric element is low. These diodes are either for rectification or to provide freewheeling paths. Whilst in theory the switches in any of the previously analysed circuits could be implemented using synchronous rectifiers, the single-supply pre-biasing circuit is particularly suitable for synchronous rectification due to the low component count, and the fact that the switches are always operated in synchronous pairs.

The circuit topology is similar to that presented in [108], with the fundamental difference that the pre-biasing circuit does not invert the voltage in one stage. The circuit of [109] is functionally similar except that it uses diodes in conduction paths, and like the SSHI circuits all the energy is flipped through the output stage, both of which increase the loss.

This circuit has a single voltage source from which the pre-charge is taken and into which the generated energy is returned. As will be shown in Section 5.2.1 there exists a value of this voltage source that eliminates free-wheeling currents entirely for both pre-charging and extraction, whilst allowing the piezoelectric capacitance to return to 0 V at the end of each half-cycle as shown in Figure 5.5.

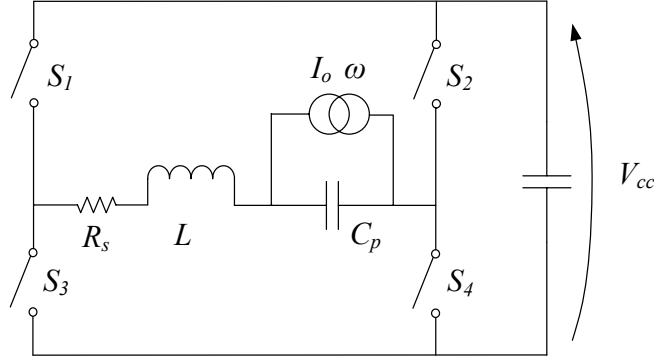


Figure 5.4: Single-supply pre-biasing circuit diagram.

The basic principle of operation for a half cycle of motion of the piezoelectric beam is as follows: during the pre-biasing phase S_1 and S_4 are closed for exactly one half period of the resonant cycle of the inductor L and the piezoelectric capacitance C_p , allowing the voltage on C_p to rise. In the subsequent generation phase the proof mass moves, thus increasing the voltage in the same polarity. At the voltage maximum, the same switch pair is closed again to discharge the piezoelectric to 0 V. The other pair of switches (S_2 and S_3) is used for the other half cycle.

A graph of the operation of the circuit, over a single discharge/charge cycle, is shown in the simulation trace of Figure 5.6. A single discharge/charge cycle is shown. It can be seen that the discharge and charge pulses produce current waveforms in the same direction. The energy balance of the power supply (top trace) shows the power supply receiving a larger amount of energy from the discharge cycle than is used to charge the piezo during the charge cycle, yielding a net energy generation.

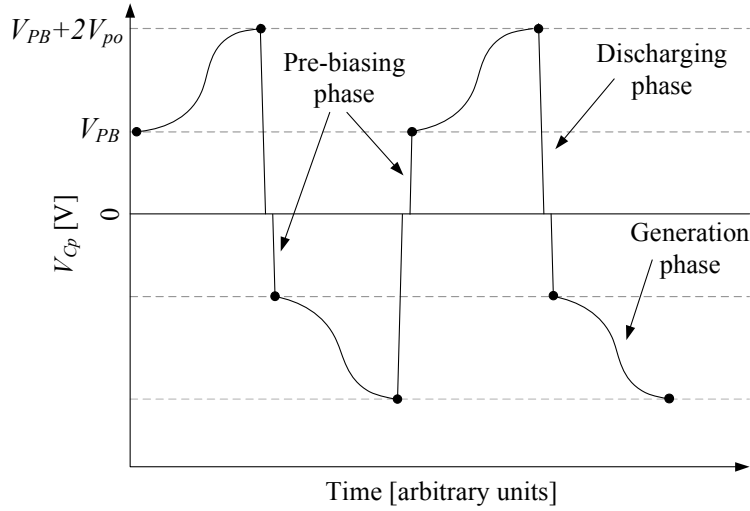


Figure 5.5: Sketch of voltage waveform on the piezoelectric capacitance for the circuit of Figure 5.4.

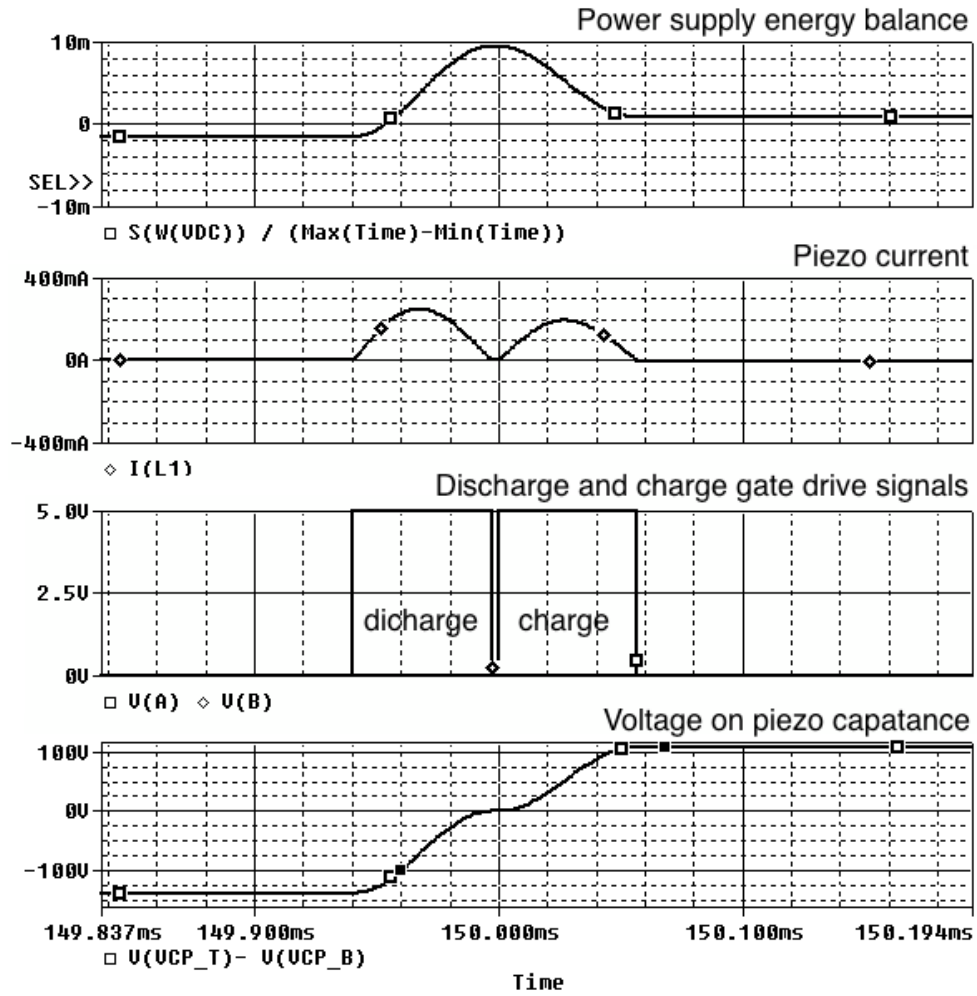


Figure 5.6: Operational waveforms of a single charge/discharge cycle of the circuit of Figure 5.4 generated by time-domain simulation.

5.2.1 Analysis

The circuit will now be analysed in terms of two operational regimes; the first provides the simplest implementation, and the second provides the maximum possible power output at the expense of some implementation complexity.

Zero-return operational regime

The simplest mode of operation is to allow the current in the inductor, and the voltage on the piezoelectric capacitance, to return to zero after each displacement half cycle. The operational cycle is similar to the pre-biasing circuit of Figure 5.2 in terms of the effect on the piezoelectric element, except that the same current path is used in charging and discharging with the advantage that fewer components are required. Since there are no free-wheel paths, the switches must open only when the current in the inductor is zero, after a time π/ω_n . The voltage after charging, V_{PB} , is given as a function of the loss coefficient γ and the power supply voltage:

$$V_{PB} = V_{cc}(\gamma + 1) \quad (5.18)$$

The voltage remaining after discharging, V_{rem} , is given by:

$$V_{rem} = V_{cc} - (V_{PB} + 2V_{po} - V_{cc})\gamma \quad (5.19)$$

The principle of operation of the circuit in this regime is that for a given harvester amplitude V_{po} , there exists a value of V_{cc} such that the voltage on C_p after discharging is exactly 0 and the switches therefore open under zero current, with no freewheeling path in operation. Solving $V_{rem} = 0$ for V_{cc} :

$$V_{cc} = 2V_{po} \frac{\gamma}{1 - \gamma^2} \quad (5.20)$$

From this, in charging C_p an energy $C_p V_{cc}^2 (\gamma + 1)$ is borrowed from the power supply, and in discharging $V_{cc} C_p (V_{cc}(\gamma + 1) + 2V_{po})$ is returned. The power from the circuit is therefore the

difference between these, multiplied by $2f$, with V_{cc} substituted with the value found in (5.20):

$$P_{out} = 8fC_pV_{po}^2 \frac{\gamma}{1-\gamma^2} \quad (5.21)$$

$$= \frac{4}{\pi} \frac{\gamma}{1-\gamma^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.22)$$

$$\approx \frac{4Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.23)$$

The circuit in this operational mode therefore has the same limiting power output as the pre-biasing circuit, but it can be seen that whereas the pre-biasing circuit cannot produce net power below $2V_{po} = V_D$, the single-supply circuit both operates and produces more power over the entire range, with the two expressions converging at sufficiently high values of V_{po} . Figure 5.7 shows the performance of the proposed circuit compared to the pre-biasing circuit over a low-voltage operating range.

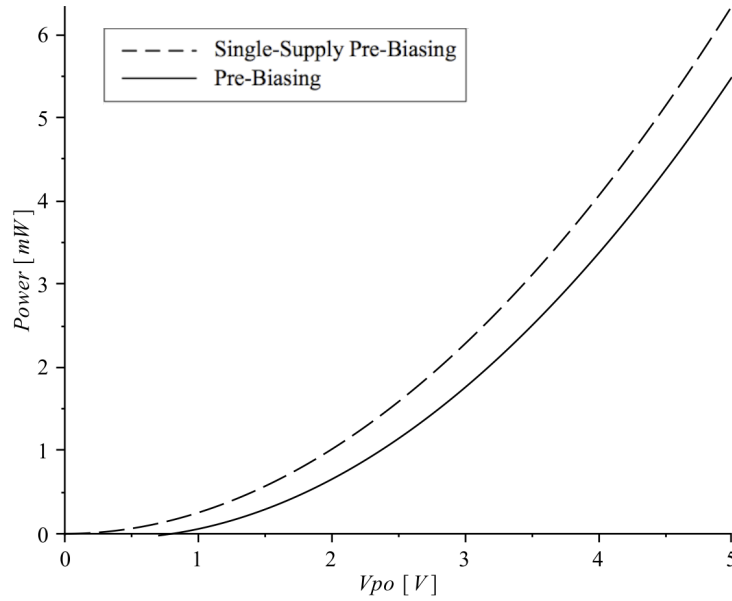


Figure 5.7: Comparison between the pre-biasing techniques in terms of output power. Graph plotted by evaluating analytical power expressions for the two circuits for component values: $C_p=100$ nF, $Q=10$, $V_D=0.7$ V, $f=100$ Hz. Graph plotted using MAPLE 14.01 with expressions (5.13) and (5.21).

General Operational Regime

The above analysis is concerned with the simplest operating regime, which minimises control circuitry overhead and complexity, but requires a voltage source at a pre-defined voltage dependent on the excitation condition. Difficulties can arise in practice when trying maintain a pre-set voltage exactly, so V_{cc} will typically vary from this value even if control circuitry attempts to keep it constant. In many cases it is undesirable to connect further voltage-modifying stages, in which case the supply voltage V_{cc} is constrained externally. If the voltage source V_{cc} is not set exactly as per (5.20), then the voltage will not return to zero after each half-cycle. The circuit of Figure 5.4 will now be analysed assuming that V_{cc} is a control parameter.

The prebias voltage V_{PB} is given as a function of the loss coefficient γ and the power supply voltage. Charging is started from a voltage that remains on the piezoelectric capacitance from a previous cycle V_{rem} (this has the same sign as the charging voltage in this half-cycle):

$$V_{PB} = V_{rem} + (V_{cc} - V_{rem})(\gamma + 1) \quad (5.24)$$

During a half cycle of mechanical motion $2V_{po}$ is added to the piezoelectric capacitance. Then C_p is discharged in a resonant cycle into the power supply at V_{cc} to a voltage V_2 :

$$V_2 = V_{cc} - (V_{PB} + 2V_{po} - V_{cc})\gamma \quad (5.25)$$

In steady-state operation $V_2 = -V_{rem}$, so solving for V_{rem} gives the steady-state voltage at the start of each cycle V_{start} :

$$V_{start} = \frac{V_{cc}(\gamma^2 - 1) + 2\gamma V_{po}}{\gamma^2 + 1} \quad (5.26)$$

To calculate the energy transfer, the steady state condition is used. Each half-cycle, an amount of energy:

$$E_{out} = C_p V_{cc} (V_{PB} - V_{start}) \quad (5.27)$$

is delivered from the power supply V_{cc} in prebiasing the piezoelectric capacitance. When the capacitance is discharged back into V_{cc} from $V_{PB} + 2V_{po}$ an amount of energy

$$E_{in} = C_p V_{cc} (V_{PB} + 2V_{po} + V_{start}) \quad (5.28)$$

is returned¹. The power from the circuit is the difference between these half-cycle energies. Substituting V_{start} with the value found in (5.26) and multiplying by $2f$ gives an expression for the steady-state power output when V_{cc} is set to an arbitrary value:

$$P = 4fC_pV_{cc} \left(\frac{(V_{po}(\gamma + 1)^2 + V_{cc}(\gamma^2 - 1))}{\gamma^2 + 1} \right) \quad (5.29)$$

To find the value of V_{cc} that maximises this expression, it is differentiated and solved to give:

$$V_{cc_{opt}} = \frac{V_{po}}{2} \left(\frac{1 + \gamma}{1 - \gamma} \right) \quad (5.30)$$

with a corresponding maximum power of²:

$$P_{max} = fC_pV_{po}^2 \frac{(\gamma + 1)^3}{(\gamma^2 + 1)(1 - \gamma)} \quad (5.31)$$

$$= \frac{1}{2\pi} \frac{(\gamma + 1)^3}{(\gamma^2 + 1)(1 - \gamma)} \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.32)$$

$$\approx \frac{4Q}{\pi^2} \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.33)$$

As γ approaches 1 (high Q) this expression converges with the power output of the zero-return operational regime in (5.21). As γ approaches 0, then this expression approaches a factor of two increase over (5.21), meaning that in situations where the inductor Q-factor is low then this operating regime is preferred over the zero-return regime.

¹Note that the V_{start} term in this expression is positive unlike previous similar expressions due to a double negative.

²Note that if the V_{cc} for the zero-return regime in (5.20) is substituted into this expression, then the power output for that regime (5.21) can be found.

5.3 Comparisons and Conclusions

Closed-form solutions for the optimised power output of each of the analysed circuit topologies, including the two variations of the pre-biasing circuit, have now been derived in terms of a common factor that now enables clear comparison between the circuits. Table 5.1 shows the relative performance of the circuits that operate into a resistive load, normalised to the simple optimal resistive load case of Fig. 4.6. Table 5.2 compares the circuits with a DC load against the optimised resistive load and bridge rectifier, assuming a zero on-state device voltage drop. For convenience we define:

$$P_{ref} = \left(\frac{I_o^2}{\omega C_p} \right) \quad (5.34)$$

The comparison was performed in terms of an inductor Q factor independent of the volume of

Table 5.1: Summary of power output of circuits with resistive load.

Circuit	$\frac{P_{max}}{P_{ref}}$	$\frac{P_{max}}{P_{ORL}} @ Q = 10, V_D = 0$
Optimal Resistive Load	1/4	1
Sync Extraction	$2/\pi$	2.55
Parallel SSHI with Resistive Load	$\frac{1}{\pi(1/\gamma-1)}$	7.5
Series SSHI with Resistive Load	$\frac{1}{\pi} \frac{1+\gamma}{1-\gamma}$	16.24

Table 5.2: Summary of non-volume-constrained power output for circuits with DC load for $V_D = 0$.

Circuit	$\frac{P_{max}}{P_{ref}}$	$\frac{P_{max}}{P_{ORL}}$	$\frac{P_{max}}{P_{RECTDC}}$
Rectified DC Load	$1/2\pi$	$2/\pi \approx 0.64$	1
Synchronous Switch Extraction (DC)	$\frac{4}{\pi} \frac{\gamma}{1+\gamma}$	2.34	4
Fixed-V Control	$1/2\pi$	$2/\pi \approx 0.64$	1
Fixed-V Control CC	$1/\pi$	$4/\pi \approx 1.27$	2
Parallel SSHI DC	$\frac{1}{\pi(1-\gamma)}$	8.76	$4Q/\pi$
Series SSHI DC	$\frac{1}{2\pi} \frac{1+\gamma}{1-\gamma}$	8.12	$4Q/\pi$
Parallel SSHI (with Synchronous Extraction)	$\frac{8}{\pi} \frac{\gamma}{(1-\gamma)(1+\gamma)^3}$	9.39	$4Q/\pi$
Pre-Biasing	$\frac{4}{\pi} \frac{\gamma}{1-\gamma^2}$	16.14	$8Q/\pi$
SingleSupply Pre-Biasing	$\frac{4}{\pi} \frac{\gamma}{1-\gamma^2}$	16.14	$8Q/\pi$

the device. The Series SSHI technique is clearly superior in this comparison, out-performing the optimal resistive load by a factor of 16.24 for $Q = 10$. Figure 5.8 shows the power gain over the rectified DC load circuit of Fig. 4.7 if the diode on-state voltage drops are neglected, and the inductor Q factor is equal to 10. From this it is clear that there are two distinct classes of circuit, those that do not depend on Q , and those that have a linear dependence. The first class have a constant power output that depends only on the excitation condition $I_o^2/\omega C_p$. The second class of circuits has a linear dependence on Q , and for these the power gain over a simple bridge rectifier tends to $4Q/\pi$ for moderate values of Q (around 10 or more). The pre-biasing circuits too have a linear dependence, and have a higher power gain over the rectifier of $8Q/\pi$. When the piezoelectric fixed voltage control circuit (which performed worse than the optimal resistive load) was fitted with a charge-cancelling switch that operated before the main charging path, the power output was doubled. This similarly is the cause of the increased power output of the pre-biasing circuit, which can achieve the same bias voltage as the SSHI with synchronous extraction but only half the energy has to travel through the inductive paths as the entire charge is never flipped. A further

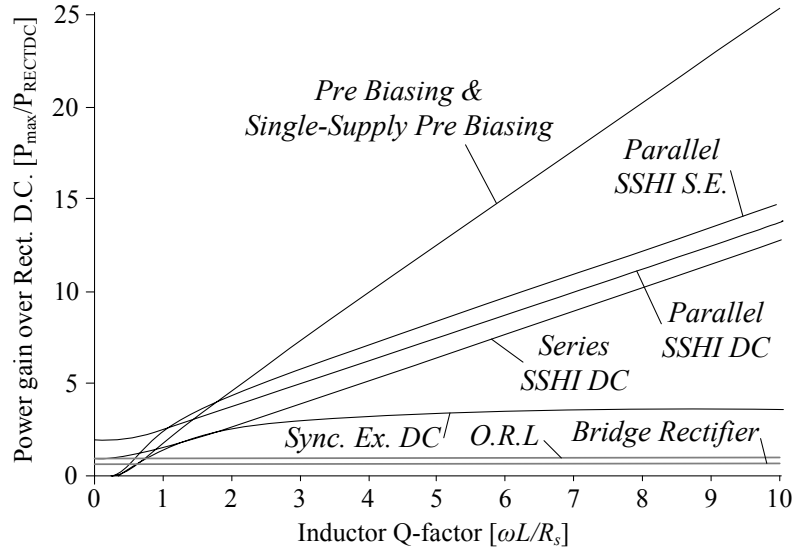


Figure 5.8: Comparison of circuit power gain when compared to rectified DC load against Q for $V_D = 0$. Graph plotted continuously using MAPLE 14.01 from the analytical expressions obtained for the optimised power output for each circuit.

comparison for those circuits with a DC output can be performed in terms of the voltage that must be supported if the circuit is operated at its optimal voltage so that free-wheeling currents

do not occur. Table 5.3 shows these voltages as a function of the excitation voltage V_{po} for each circuit, sorted in order of maximum value of any of the voltages the circuit is required to support. In the case of low input excitation, a low output voltage (particularly where this is less than 1 V) may mean that the harvester requires further DC/DC power converter stages before a load can be connected.

Table 5.3: Summary of non-volume-constrained optimal power supply voltages for $Q=10$, $V_D=0.5$.

Circuit	Voltage normalised to V_{po}
Rectified DC Load	$V_{out} = 0.48$
Series SSHI DC	$V_{out} = 0.5$
S.S.E. DC	$V_{out} = 0.9$
Single-Supply Pre-Biasing	$V_{cc} = 6.3$
Parallel SSHI S.E.	$V_{1ss} = 5.4, V_{out} = 6.8$
Parallel SSHI DC	$V_{out} = 6.9$
Pre-Biasing	$V_{PB} = 11.8, V_{out} = 81, V_{cc} = 6.4$

In order to perform a fair comparison between the circuits with a fixed volume, the Q factor of the inductor must be reduced by a factor of 3 in the original pre-biasing circuit, as it has 3 inductors which must share the fixed volume. Figure 5.9 shows a comparison of the power output against the piezoelectric open-circuit voltage V_{po} for the devices with a DC output if this is done (plus the optimal resistive load for reference). It can be seen that of these the single-supply pre-biasing circuit of Fig. 5.4 performs best over the entire range, while the original pre-biasing circuit's performance is negatively affected by the reduced inductor Q factor.

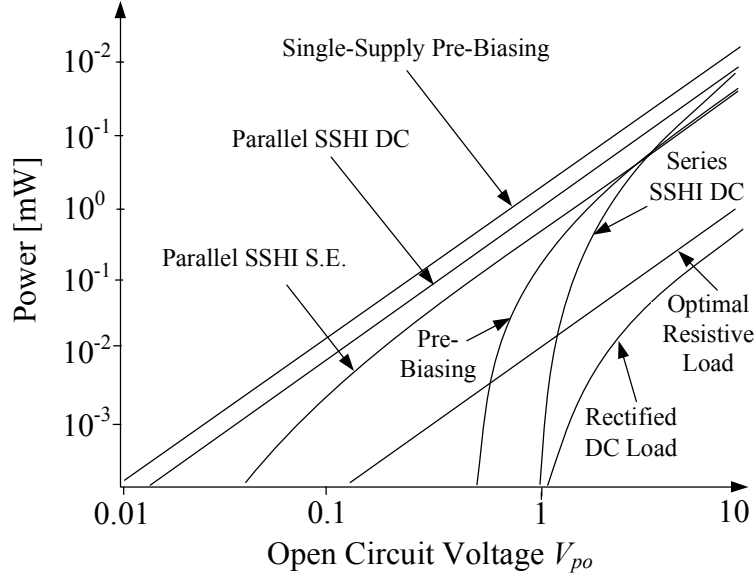


Figure 5.9: Comparison of power output for circuits with DC output against open-circuit voltage V_{po} . Graph plotted continuously using MAPLE 14.01 from the analytical expressions obtained. Circuit parameters were $V_D = 0.7$, $Q = 10$, $C_p = 100 \text{ nF}$.

This Figure excludes the necessary overhead of control circuitry. The better performing circuits at low values of V_{po} all require overhead power for the associated control circuitry, which means their gross output power must be greater than the minimum required for this function. If the control can be achieved within $1 \text{ } \mu\text{W}$, then taking as an example the SSPB circuit operating at 100 Hz , with an inductor Q of 10 and a C_{po} of 100 nF , the total power can only exceed $1 \text{ } \mu\text{W}$ for V_{po} above about 0.16 V .

5.4 Further Work

The pre-biasing circuit topology discussed in this chapter demonstrates a large theoretical power increase over other circuit topologies, at the expense of significantly increased control complexity and component cost. Before a device capable of harvesting energy from water flow can be developed, there are some remaining issues that have not been addressed in this work. The pre-biasing circuit requires a small amount of power for control electronics, without which the piezoelectric element is

open-circuit and therefore no power is delivered to the output. A self-start bootstrapping circuit is needed, which would accumulate enough energy to start the control circuit. It could do this by initially rectifying the output from the piezoelectric element directly, and would therefore need a way of disconnecting itself once the main circuit had activated to avoid grounding the piezoelectric capacitance during normal operation. A fair evaluation of the best-performing circuits studied in terms of a fixed volume, including the volume taken up by a bootstrapping circuit, would be an interesting project.

The optimisation of the circuit for the application domain is a complex process, and is out of the scope of this work. For example, the desired power output profile of the circuit depends on the desired sensor communication regime. Also, the typical flow rates in the water pipes vary considerably over a diurnal cycle. If continuous communication is desired then the circuit should be optimised such that it delivers power across a wide range of flow rates, at the possible cost of achieving the highest overall daily energy generation. If a large amount of energy storage is available, in the form of batteries or capacitors, or if periodic communication is acceptable, then the optimisation might focus on gathering as much energy over the day as possible at the expense of a lower worst-case power delivery. This regime might mean long periods of inactivity, with a few short burst of very high generation.

The control strategy used in the experiments was a zero-crossing detector that used a low-pass-filtered version of the position signal. This performed poorly when the frequency was varied more than around 50 % in either direction from the frequency the routines were designed for. In a utility water main, the diurnal pressure variation is considerable, so in order to extract the maximum amount of energy the pre-biasing circuit would need to be able to operate over a much wider frequency range. More sophisticated signal processing and analysis could yield a control strategy that would be able to operate over such a wider frequency range. In addition, other control regimes, such as one in which the system deliberately avoids pre-biasing the piezoelectric capacitance in situations where it would not yield a net power gain, have not been considered.

Chapter 6

Experimental Validation of Pre-Biasing Techniques

The theoretical results of Chapter 5 predict that the pre-biasing circuit topology can yield a significant increase in power output compared to other techniques. In practical experiments involving micro-scale devices, the deviation from predicted results can be multiplied by effects that are particularly dominant at the micro-scale, such as stray capacitance, RF interference and interference from instrumentation and external power supplies. These amplify the errors caused by modelling assumptions and can mean that theoretical predictions translate poorly into realisable circuit performance. Therefore, experimental verification is particularly important to demonstrate an increase in power output.

6.1 Simulation

In order to validate the model, time-domain simulations were performed using Cadence *OrCAD* version 16.3 with the *PSpice* circuit simulation engine. The minimum simulation step time in all cases was $0.1\ \mu\text{s}$. Simulations were run for 500 ms, with a one-second initial period during which no data was recorded to allow the circuit to enter steady-state operation. Graphs presented in the section may display a shorter time range than this for clarity. In all cases, the timing signals were pre-programmed constants to signal generator components, and therefore the power taken to detect zero crossings, charged/discharge gates etc. is not included.

6.1.1 Pre-Biasing

The first circuit simulated was that of Figure 5.1. In order to execute a realistic simulation, the following non-ideal component models were used, which were the best available at the time:

- 2N7000 MOSFETs were used for all switches.
- Schottky ZC5800 diodes were used in the main conduction and free-wheeling paths.
- Inductors were modelled as ideal but with $3\ \Omega$ series resistance.

The *OrCAD* circuit layout is shown in Figure 6.1. Note that the circuit layout includes a reference circuit (top right) with the same piezoelectric model but with an optimised resistive load.



The main circuit parameters are: $L=5$ mH, $L_R=3$ Ω , $C_P=65$ nF, $I_o=500$ μ A, $V_{cc}=25$ V, $V_{out}=28.2$ V. The additional series impedance of the MOSFETs, $R_{DS_{on}}=20$ Ω each. The frequency of excitation was 80 Hz.

Figure 6.2 shows the waveforms produced by the simulation. The top waveform shows the voltage on the piezoelectric capacitance, compared to the constant output voltage V_{out} and supply voltage V_{cc} . The middle waveform shows the two alternating charging current waveforms (the current in inductors L_2 and L_3 of Figure 6.1). The bottom waveform shows the current in the discharge inductor L_1 . Note that the phase of the extraction waveform is 0.5 ms before the beginning of either charging waveform to ensure no cross-conduction.

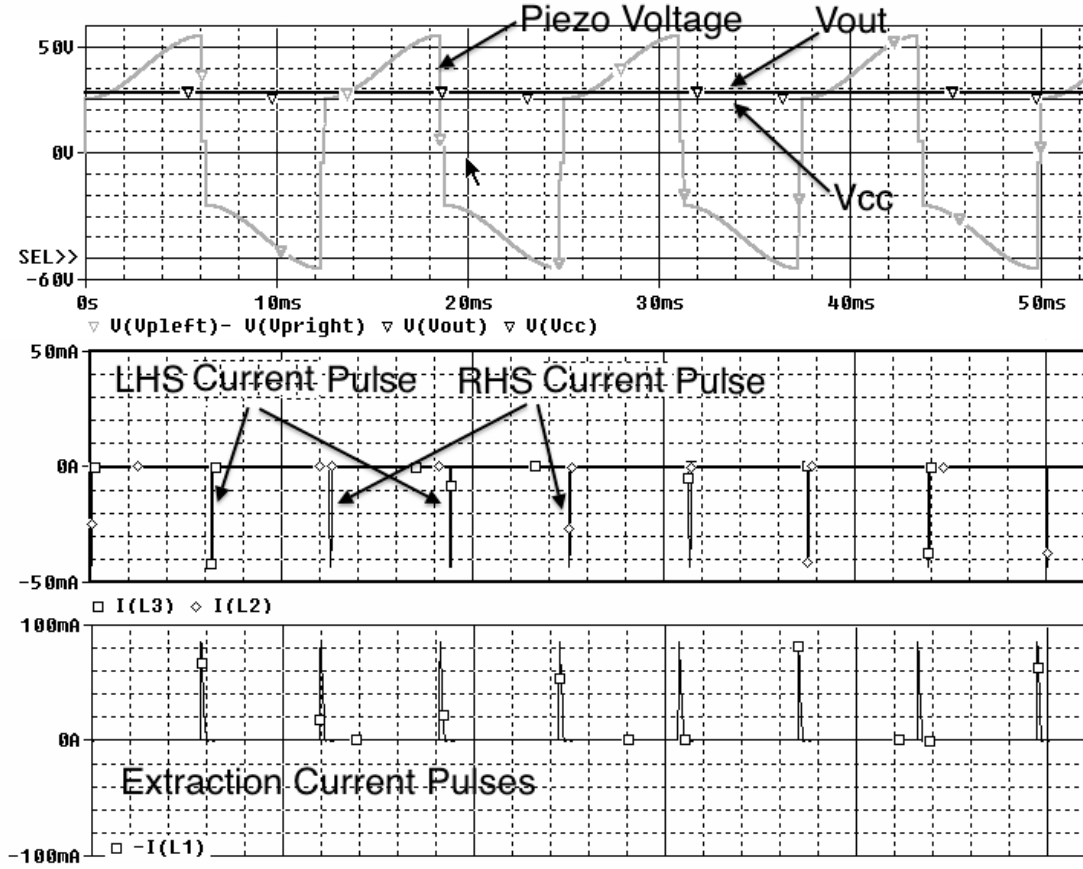


Figure 6.2: Simulation waveforms. Top: Main voltage waveforms showing voltage across piezoelectric capacitance, pre-charge supply voltage (V_{cc}) and output voltage (V_{out}); Middle: left and right charge pulses; Bottom: charge extraction pulses

Power Output

The predicted power output for this circuit configuration is calculated by multiplying (5.11) by $2f$ in this configuration, which gives a value of 10.6 mW for the circuit parameters listed above.

In the simulation, the power output of the circuit is defined as the difference between the sum of the energy in and out of the two voltage sources V_{cc} and V_{out} divided by the total simulation time. When run for 500 ms with a maximum simulation time step of 0.1 μ s, the reported power output was 10.94 mW, which is within 3% of the model prediction. The difference between the model and simulation values is therefore within the tolerance of simulation error.

For comparison, the optimal resistive load for this configuration is $1/\omega C_p = 30.61k$, with a predicted power output of 1.91 mW. The simulated optimal resistive load circuit with this configuration produced 1.92 mW, a deviation of around half of one percent.

6.1.2 Single-Supply Pre-Biasing

The second embodiment of the pre-biasing circuit, Single-Supply Pre-Biasing (Figure 5.4), was designed using *OrCAD* capture, as shown in Figure 6.3. For this simulation, due to the requirement that the switching components must conduct and block in both directions, the MOSFETs were modelled as ideal switches with a 20 Ω on-state resistance and 50 M Ω off-state leakage resistance. Section 5.2.1 describes two operating regimes (i.e. values of V_{cc}) for the circuit: zero return (where at the end of each cycle the voltage on the piezoelectric capacitance is zero), and the more general regime where the voltage at the end of each half cycle is undefined. The circuit was first simulated in the general operational regime, and then the supply voltage V_{cc} was constrained such that the voltage returned to zero each half cycle and the simulation was run again for comparison. For each simulation, the power output predicted by the analytical result in (5.29) was compared with the power output calculated from the simulation run.

Figure 6.4 shows the operation of the circuit for an operating regime where the supply voltage V_{cc} is not set for zero-return (the general regime). Figure 6.5 shows the simulation result of the same circuit, but with V_{cc} configured such that the voltage returns to zero each half cycle.

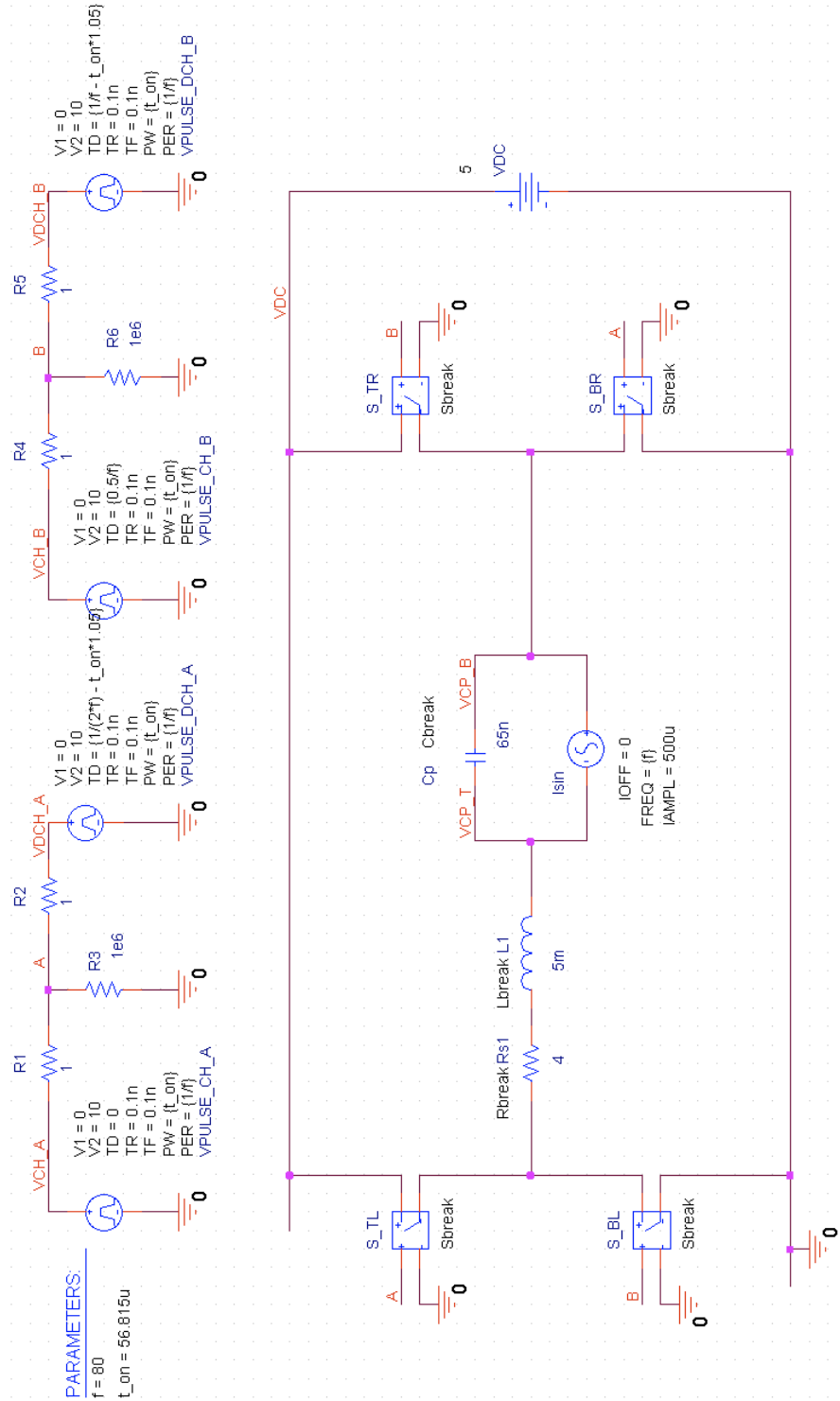


Figure 6.3: Single-Supply Pre-Biasing circuit layout represented in *OrCAD Capture* 16.3 showing main circuit components and drive signal generators. Each gate drive signal (A and B) is driven by two different signal generators, one for the discharge and one for the charge cycle.

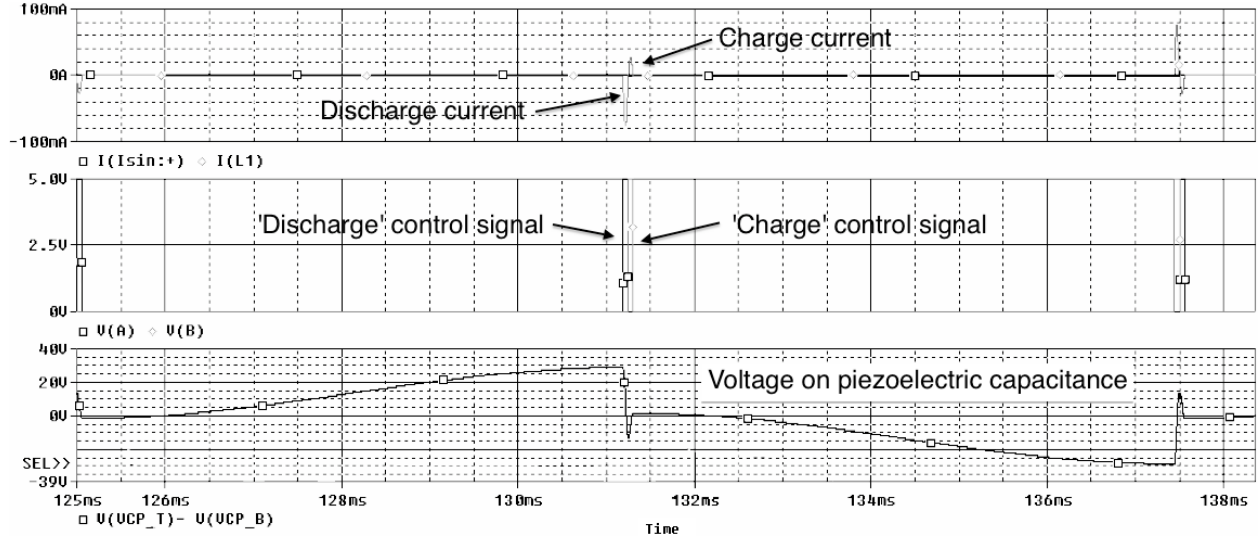


Figure 6.4: Simulation waveforms for single-supply pre-biasing circuit with $V_{cc} = 5$ (non return-to-zero regime). Top: current in the piezoelectric capacitance; Middle: left and right control signals (showing charge and discharge); Bottom: Voltage across the piezoelectric capacitance.

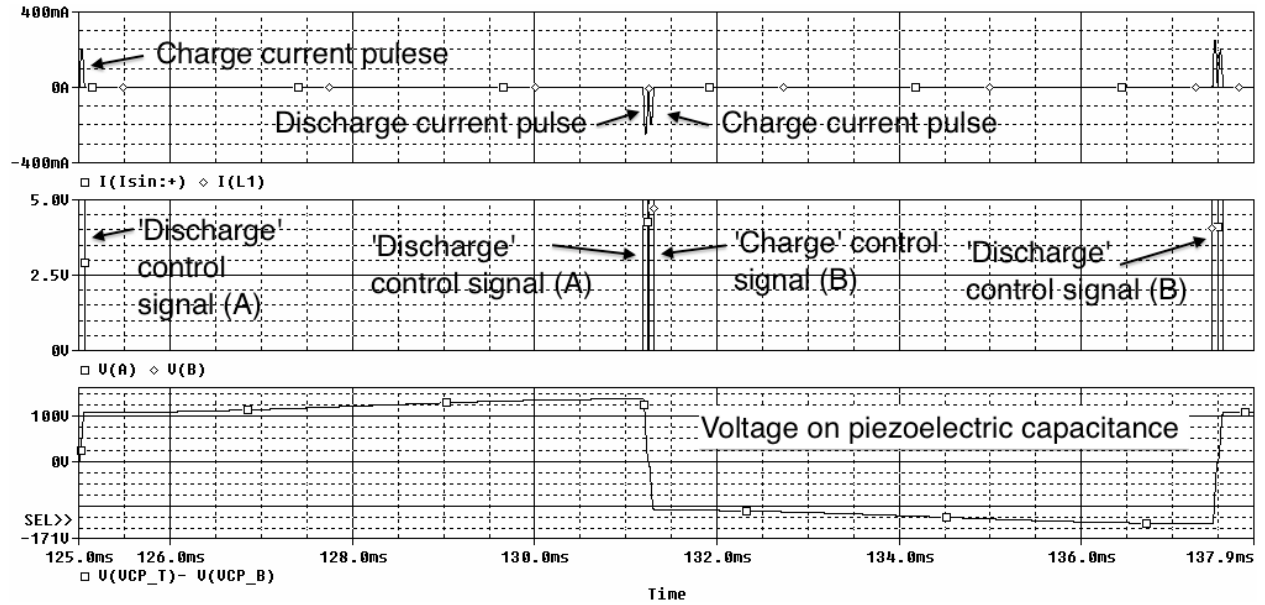


Figure 6.5: Simulation waveforms for single-supply pre-biasing circuit with $V_{cc} = 60.78$ (return-to-zero regime). Top: current in the piezoelectric capacitance; Middle: left and right control signals (showing charge and discharge); Bottom: Voltage across the piezoelectric capacitance.

Power Output

The power output was calculated by summing the energy in and out of the V_{cc} voltage source component, and dividing this by the simulation time, which was 500 ms in all cases as this is an exact multiple of the period of mechanical oscillation at 80 Hz. The simulation was allowed to run for 1 s without recording results to allow the system to reach steady-state operation.

Table 6.1 shows the results of the simulation against four voltages. First the circuit is run at 3.3 V and 5 V, which are two voltages commonly used by the low-power electronics of a sensor node. Then the optimal voltages predicted by the two operational regimes: zero-return (5.20) and non-zero-return (5.30) are used.

Table 6.1: Comparison of model prediction and *PSpice* simulation results for the Single-Supply Pre-Biasing circuit run at 3.3 and 5 V (for interfacing to sensor nodes) and at optimal voltages for the circuit.

V_{cc}	Model prediction from (5.29)	Simulation result
3.3 V	2.014 mW	2.013 mW
5 V	3.008 mW	3.006 mW
Zero-return $V_{cc_{opt}}$ from (5.20), 60.78 V	19.347 mW	19.200 mW
Generalised $V_{cc_{opt}}$ from (5.30), 61.73 V	19.351 mW	19.209 mW

6.1.3 Analysis and Comparison

There is a good agreement in the results of Table 6.1 between the power output predicted by the analysis and that obtained through time-domain simulation. Lower values of V_{cc} were particularly accurate, with negligible discrepancy found. At the higher (optimal) V_{cc} range, a small difference was noted between the analytical model and the simulated power output, around 1 % less power was recorded by the simulation than predicted by the model. The interpretation of this small discrepancy is as follows. The analysis of section 5.2 makes the assumption that the transfer of charge to and from the piezoelectric capacitance is instantaneous, and that the discharge and charge cycles occur virtually simultaneously. This implies that the entire half-period is available for

charging the piezoelectric capacitance from the source motion. In practice, MOSFETs have some non-zero gate capacitance and therefore the system must reserve a small amount of time between discharging and charging to ensure that both pairs of MOSFETs do not conduct together. In addition, with non-infinite Q-factors, then for a small proportion of the overall mechanical period one of the pairs of switches is closed, meaning the piezoelectric capacitance cannot charge from the piezoelectric current source. For $L=5$ F, $C_p=65$ μ F, $R_s=44$ Ω (as used in the simulation runs), then the time taken to discharge fully is 56.8 μ s, which at an 80 Hz is around 0.5 % of the overall period, which occurs twice per half-cycle (i.e. 1 % of the overall mechanical period).

6.2 Experimental Results

In order to validate the theoretical results, an experimental programme was conducted to build and test the two pre-biasing circuit topologies. The first Pre-Biasing circuit experiments were conducted by the author; the experiments on the Single-Supply PreBiasing circuit were principally conducted by A. Elliott [110], with assistance from the author.

The mechanical vibration source was provided by an IMV PET-05-05A 49N Shaker System with 5mm maximum peak-to-peak displacement, powered by an IMV PET-0A 0.05kVA Amplifier. The shaker system was used in closed-loop mode to eliminate any damping effect that might be caused by the action of the pre-biasing circuit, although it is not expected that such damping would make a measurable difference to the oscillation amplitude.

6.2.1 Pre-Biasing

The circuit of Figure 5.1 (Pre-Biasing) was constructed on breadboard. The components used were as follows:

- The switches were Fairchild 2N7000 FETs (in TO-92 package), which have a typical $R_{DS_{on}}$ of around $1.8\ \Omega$. The two high-side switches, and the extract switches, were controlled by FAN7361 high-side gate drivers.
- The diodes were ZC5800 Schottky diodes with typical maximum forward voltage drop of 410 mV.
- The piezoelectric element was a Kingsgate KPSG-100 piezoelectric loudspeaker, with a capacitance of 65 nF.

It was found during some initial testing that due to the low charge levels involved (the capacitance of the piezoelectric element was 65 nF), standard $100\ \text{M}\Omega$ oscilloscope probes created noticeable additional load on the circuit, as the open-circuit amplitude was noticeably lower with the probe connected. Instrumentation was therefore built in to the circuit design. Analog Electronics AD549 high-input-impedance operational amplifiers were used to provide a voltage reading for the piezoelectric capacitance.

The piezoelectric loudspeaker device was mounted onto the sinusoidal vibration platform with a M6 hex nut as a mass (1.6 g) The un-damped mechanical resonant frequency of this configuration was found by experimentation to be 80 Hz.

Control

The pre-biasing circuit requires precise gate control signals to achieve the desired pre-bias and discharge voltages. The charge and discharge pulses must be produced close to the extremes of mechanical position in order to correctly bias the piezoelectric capacitance. Due to the complexity of the control, an integrated control solution was not attempted. Instead, an external Texas Instruments DSP board was used, and a software implementation of a zero-crossing detector was written to synchronise the gate signal with the mechanical motion. Figure 6.6 shows a block diagram of the major components of the software implementation.

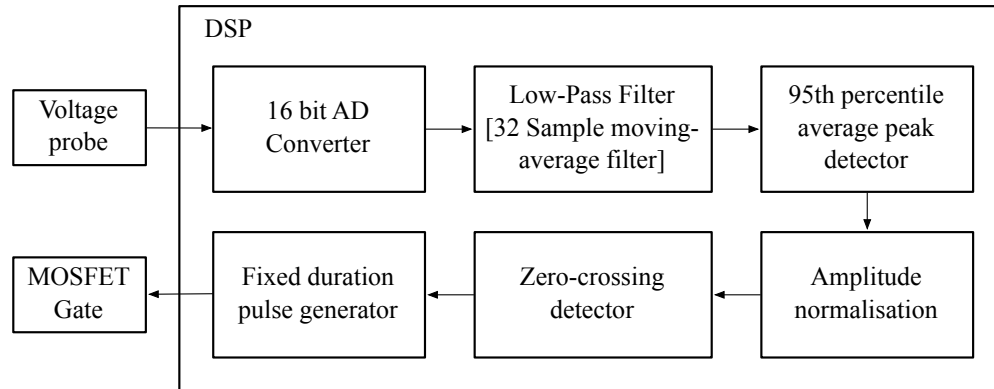


Figure 6.6: Block diagram of DSP implementation for controlling gate-drive signals.

An oscilloscope trace showing the operation of the circuit with this control system is shown in Figure 6.7, with the gate drive control signals highlighted. The bottom signal in the graph is the raw output of the voltage of the piezoelectric capacitance.

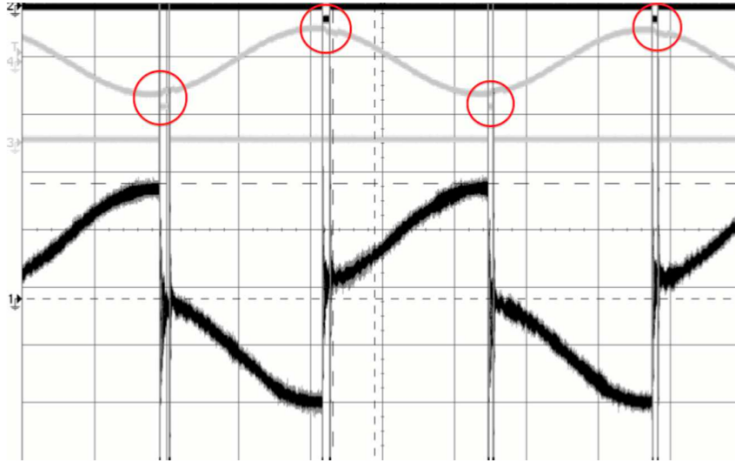


Figure 6.7: Oscilloscope trace showing gate drive signals (highlighted), the voltage on the piezo-electric capacitance (bottom) and the estimated position of the mechanical device (top).

PCB Implementation

Despite the presence of software filtering it was found that the breadboard was susceptible to noise from outside sources and self oscillation from the high-frequency gate drive signals, as can be seen in Figure 6.7. This noise caused errors in position estimation at low amplitudes, and the significant oscillation that can be seen during the gate pulses wasted energy each cycle and prevented the circuit from working with an acceptable efficiency.

To address this the circuit was constructed on a custom PCB, the layout of which can be found in Appendix A, Figures A.1 and A.2. The PCB was designed to maximise the distance between signal tracks and to minimise the capacitive coupling, and to minimise the length of control and conducting tracks to minimise inductance. The final construction is shown in Figure 6.8.

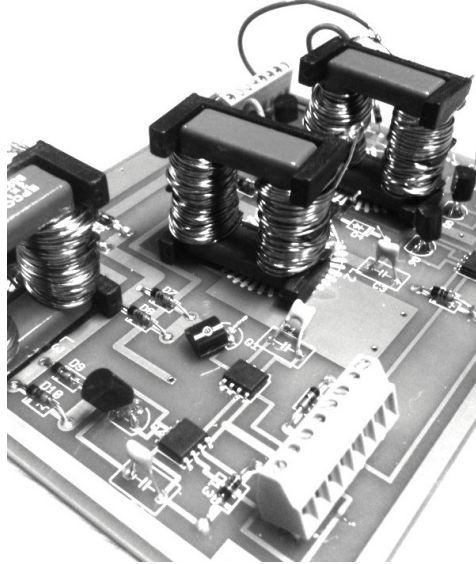


Figure 6.8: Photograph showing the prototype pre-biasing circuit constructed on the PCB of the design of Figures A.1 & A.2.

With the same operating conditions and excitation as the breadboard implementation the noise on the measurements taken from PCB implementation was significantly reduced. The system was operated at an excitation frequency of 80 Hz, with an approximate peak-to-peak amplitude of 4 mm. Figure 6.9 shows an oscilloscope trace of the operation of the circuit with the PCB. The bottom waveform shows the actual position waveform of the piezoelectric cantilever, while the top sinusoid shows the position estimated by the software on the DSP. The raw output firing angles of the two gate drives can be seen above and below the top sinusoid.

The control circuit was able to estimate the true position of the piezoelectric device far more accurately with the PCB implementation of the circuit. Figure 6.10 shows the circuit operated at 1/10 the input voltage excitation, leading to an oscillation amplitude that was too small to be measured directly with the available instrumentation. It can be seen that the position estimation (top waveform) performed by the tracking algorithm correctly recovers the position signal and is able to fire the gate pulses at the correct angles. The control system was able to operate over a wide range of input amplitudes, and lowering the input amplitude did not have a noticeable effect on the ability of the control system to track the amplitude peaks.

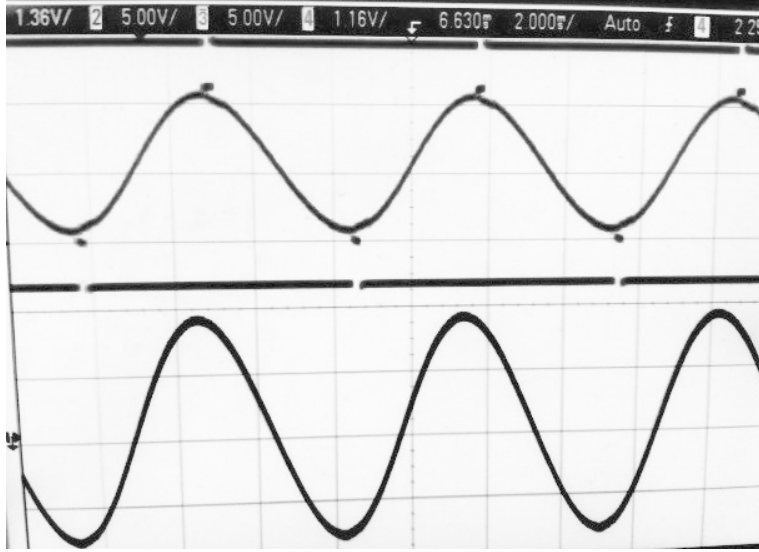


Figure 6.9: Operation of pre-biasing circuit (on PCB) showing phase-shifted excitation waveform (bottom), DSP-estimated position (top) and gate-drive firing signals (top; overlaid on position waveform). Scales are arbitrary.

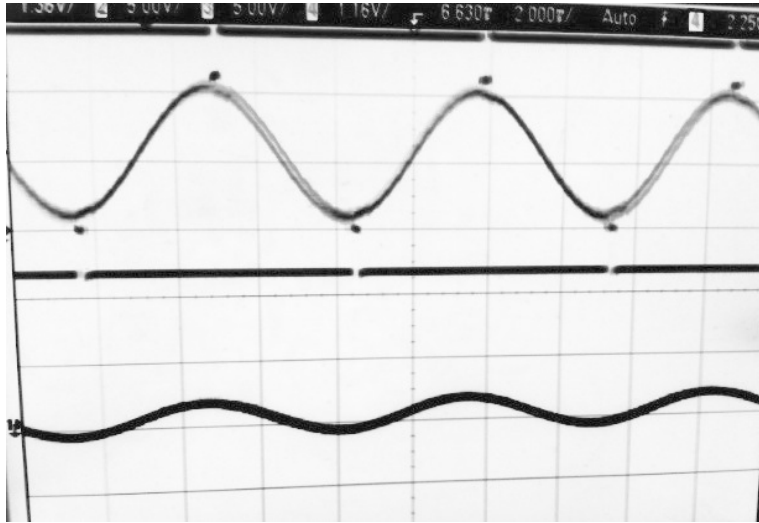


Figure 6.10: Operation of pre-biasing circuit (on PCB) under small input excitation, showing phase-shifted excitation waveform (bottom), DSP-estimated position (top) and gate-drive firing signals (top - overlaid on position waveform). Scales are arbitrary.

The optimal resistive load for this mechanical excitation and piezoelectric capacitance of 65 nF was 30.3 k Ω . This delivered an output power of 0.103 mW. At the optimal pre-bias voltage of 12.6 V, the pre-biasing circuit delivered an output power of 1.11 mW, which is an improvement of 10.8 times over the resistive load. This result was reported in [106]. The power output predicted by the model for this configuration was 1.34 mW, so the circuit was only operating at 80% of the predicted output power. The difference was accounted for by resistance and inductance of the wires connecting the piezoelectric element to the circuit, which were around 70 cm long in each direction.

6.2.2 Single-Supply Pre-Biasing

In conjunction with A. Elliott (in [107] and [110]), the second revision of the circuit, Single-Supply Pre-Biasing, was constructed. To obtain more accurate power measurements a Yokogawa WT210 power analyser was used to sum all energy leaving and entering the power supply. The timing system was similar to the one used for the previous pre-biasing experiments, and consisted of zero-crossing detection, a time delay and a monostable to time the gate pulses correctly.

The P-type MOSFETs were model BSH201, the N-type were model BSS138. A Kingsgate KPSG-100 loudspeaker was used again, this time with a capacitance of 52.9 nF. The combined series resistance of the loudspeaker and 7.5 mH inductor was 65 Ω , giving a combined Q factor of 5.8.

Switch Implementation

There is additional complexity in the implementation of the SSPB circuit over the Pre-Biasing circuit due to the requirement that the switches bi-directionally block and conduct during different parts of the cycle. The solution was to use P-type MOSFETS in place of both switches, with the addition of a series N-type MOSFET to each of the low-sides as shown in Figure 6.11.

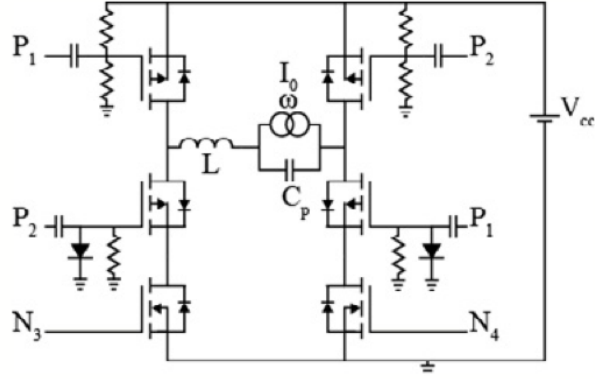


Figure 6.11: Single-Supply Pre-Biasing circuit bi-directionally blocking and conducting switch implementation.

Although the number of logical control signals for this circuit remains the same at two, this MOSFET configuration requires six gate drive signals. The control arrangement shown in Figure 6.11 negated the need for high-side gate drives.

Power Output

The mechanical resonance of the configuration used for this circuit was found to be around 200 Hz. The circuit was operated at this frequency, and the piezoelectric voltage and current waveforms are shown in Fig. 6.12. The top part shows the piezoelectric waveform for just over one cycle, and the bottom part shows a detailed view of the piezoelectric current and voltage for one discharge/charge cycle.

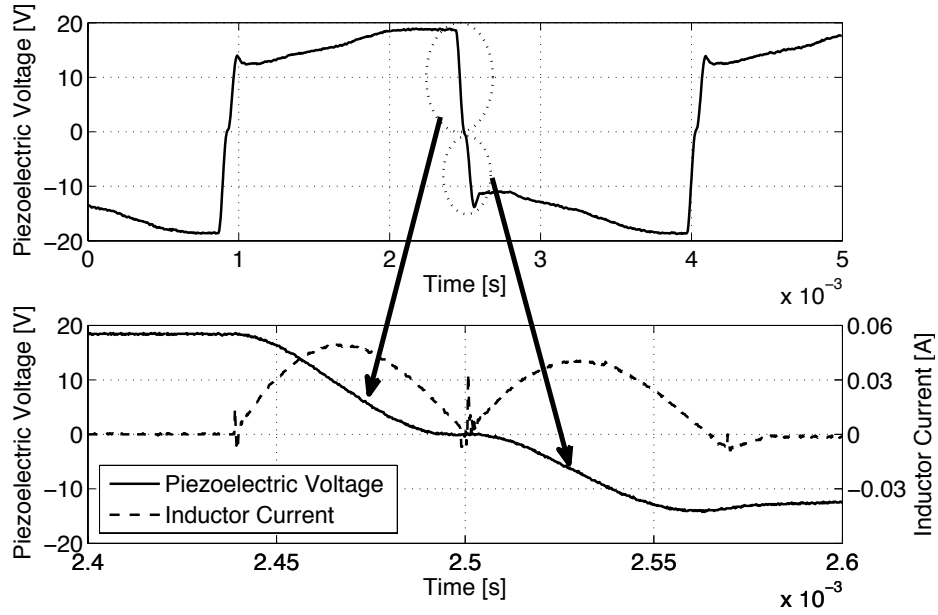


Figure 6.12: Experimentally measured waveforms showing piezoelectric voltage over 2 cycles (top), and an expanded view of this voltage and of the inductor current during a transition (bottom).

As can be seen, the resonant discharge pulse causes the piezoelectric voltage to return to near zero. Once this has occurred, the charging pulse then pre-biases the piezo capacitance for the next half-cycle of motion. It can be seen that the discharge pulse has a greater current magnitude than the charge pulse due to the piezoelectric capacitance having been charged by the mechanical motion during the half cycle.

The useful power output of the circuit as measured by the power meter was 2.6 mW. This is somewhat short of the power output predicted by the model of 6.5 mW. The discrepancy between the two values is accounted for by the following:

- Inaccuracies in timing signals causing gates to be opened and closed at sub-optimal instants,
- 0.4 mW were used for the control circuitry, therefore by optimising this the useful power could be increased,
- Measuring the Q-factor of the inductor accurately proved to be difficult, and the complex impedance of the custom-wound inductor had other terms than just resistance and inductance.

These were not accounted for in the model, but at the high-frequency of the switching, stray capacitance may have negatively affected the charge transfer.

6.3 Summary

The simulation results demonstrate a very good agreement with the power output predictions of the analytical expressions of the previous chapter, to within a few percent. The modelling assumptions taken do not introduce significant error when the simulations are run with realistic *PSpice* component models.

The experimental programme showed that both pre-biasing circuits are able to generate more power than the resistive load. However, the amount of harvested power was still below the amount predicted by the analytical expressions. The difference was accounted for by the losses in the components used, and difficulty estimating and measuring the losses, particularly the Q factor of the inductors used. Lower-loss experimental implementations are required in order to accurately validate the model with a high degree of accuracy.

Chapter 7

Conclusions

This thesis investigates two fundamental technological challenges which are critical to enable water utilities to deploy infrastructure monitoring apparatus with high spatial and temporal resolution: increasing the power output from a vibration-driven energy harvester based on piezoelectric transduction by increasing the available damping force, and the processing, storage and querying of large volumes of data resulting from the increased level of pressure and flow rate monitoring.

7.1 Summary and Contributions

The undertaken research work has lead to the following contributions:

Data Management

- The application-specific data storage requirements were reviewed, which highlighted that there exists a set of conceptually simple operators, which are mostly simply aggregates, that are highly effective in predicting failure modes and operating condition of water infrastructure. Making scientific use of this data at a high enough resolution presents technical challenges in terms of processing and storage.
- The practical implications of storing and processing such large datasets were examined, and

it was shown that single-server architectures are not suitable due to limitations on bandwidth and processing capabilities. It was concluded that a scalable, multi-server architecture is essential to deliver the required level of performance and reliability.

- Existing database technologies were reviewed and compared and it was concluded that a specific database system for time-series data is needed that enables researchers to reliably store the large volumes of data that would be captured during a typical monitoring programme, and to execute a set of simple predefined operators and custom queries on the data without having to learn about the fundamentals of distributed systems or distributed query processing.
- A simplified query structure was designed, based on highly-optimised aggregates, that aims to enable queries that can be used to eliminate as much data as possible from consideration, before that data is passed to other domain-specific tools.
- A durable, distributed time-series data storage layer, with a optimised in-memory index layer, was designed to enable a query execution layer to have rapid access to raw data and index records.
- A *MapReduce*-like query execution framework was designed, to allow researchers to implement custom algorithms over large datasets without having to understand the complexities of the distribution of tasks amongst a cluster of systems.
- An experimental programme was designed, which integrated the developed integrated sensor technologies and data processing architecture. The system was successfully tested over a period of 18 months in operational water supply systems.

Energy Harvesting

- The potential sources of energy in water distribution mains that are able to support the continuous operation of a real-time sensor device were reviewed, and it was concluded that there are few candidates. A harvester based on fluid-induced vibrations was outlined as the most suitable approach, with piezoelectric transduction.
- The dynamic damping force of a typical water utility application was estimated, and it was

concluded that the limiting factor of the power output of a practical device will be the damping force that can be achieved with piezoelectric transduction.

- An analytical framework to objectively compare the existing circuits that aim to increase the power output of a piezoelectric energy harvester was designed, and used to review the state-of-art circuits.
- A new piezoelectric interface circuit was designed by the author, called Pre-Biasing. This circuit increases the power output by a factor of two compared with the state-of-art circuits.
- The new piezoelectric interface circuit topology (Pre-Biasing) was verified experimentally, and it was shown that the circuit produces more power than the optimal resistive load, and a broad agreement with the model and simulation was demonstrated.

7.2 Energy Harvesting Interface Circuits

The problem of energy harvesting in operational water supply systems has received significant attention, but with little progress. The limitations of remote access points, accessibility, underground conditions, etc, have significantly limited the potential energy sources. Recent technologies, as of Summer 2012, using the installation of miniature turbines on a bypass of a pressure-reducing valve, has demonstrated one potential solution. The main limitation of such a device is the requirement of a substantial local pressure drop, which is only available at discrete points in the network. The approach to local power generation that was discussed in this thesis is based on flow-induced vibrations. The energy harvester can be placed in any diameter pipe and location, and has no impact on the hydraulic conditions.

The maximum extraction of energy requires optimal coupling between the mechanical flow-induced vibration source and the electrical interface. State-of-art circuits for interfacing to piezoelectric energy harvesters have been analysed and compared in a unified analytical framework. The circuits have different levels of functionality, in that some are only able to dissipate energy in a resistive load and some are able to store the generated energy in a battery or capacitor. Under the considered operating conditions for energy harvesting devices, the damping force achievable from the

piezoelectric element due to extraction of electrical energy is below that required for the harvester to operate at maximum power density. The main challenge was to extract the maximum power possible with from a device with a shunt capacitance, which is inherent in the piezoelectric element. The circuits analysed that can extract more energy from the piezoelectric element than can be extracted using a simple bridge rectifier do so because they actively modify the voltage on the piezoelectric capacitance. This means the charge from the current source is forced into a higher voltage, corresponding to increased work being done and correspondingly, an increase in the electrical damping and output power.

A theoretical model for the power output of each circuit was obtained, which includes non-idealities in the components, particularly diode on-state voltage drops and series resistance in inductors. It was shown that, if diode drops are neglected, the output power from each circuit can be normalised to $I_o^2/\omega C_p$. This allows the performance of each circuit to be compared purely as a function of the Q-factor of any paths containing an inductor. The circuits that flip the polarity of the charge on the piezoelectric element at the extremities of device motion, termed SSH, behave with approximately the same performance - the power gain over a simple passive bridge rectifier arrangement tends to $4Q/\pi$ for moderate values of Q (around 10 or more).

A new circuit technique termed pre-biasing was developed. The technique separately pre-charges and discharges the piezoelectric capacitance and has a power gain over the simple bridge rectifier case of $8Q/\pi$. While the pre-biasing and single-supply pre-biasing variants have the same theoretical power, the single-supply pre-biasing circuit requires 2 fewer switches and 2 fewer inductors (Fig. 5.4) meaning a higher practical Q-factor is achievable in a given volume. In theory any of the diodes in these circuits could be implemented with synchronous, bi-directionally blocking switches, although as seen in Table 5.2, the pre-biasing circuit retains advantages in terms of higher performance and lower complexity.

7.3 Data Management

The prospect of self-powering sensor nodes will drastically increase the volumes data generated for the near-real time operational management of water supply systems. This requires a paradigm shift

in data management to be adopted by the water utilities. Optimal tools for combining statistical and analytical models would be required. These will be better served by a framework which allows searching large data streams and designing custom queries.

Rather than processing the entire dataset for features of interest, a strategy was proposed whereby a considerable amount of the data is eliminated from processing beforehand using simple filter clauses. The resulting reduced dataset can then be passed to bespoke, application specific tools for further processing. It was noted that disk access is often the slowest part to query execution. The key finding was that most of the common operational questions can be satisfied with a relatively small amount of disk access, provided that both the query language and the indexing system are designed and optimised in such a way as to minimise this.

An operational programme was designed and completed, which gathered a number of data streams of high-frequency pressure from pumping stations in London, and remote distribution mains. The prototype distributed database was tested against representative large datasets to validate the execution performance for a set of representative queries. Tests conducted with simulated failures showed that the distributed database was able to recover and continue serving requests after a short period. In addition, the system was successfully deployed over a period of 18 months.

Chapter 8

Further Work

This chapter outlines the recommendations for further work.

8.1 Limitations of the Analytical Model

The analytical framework used to compare the power extraction circuits in Chapter 4 attempts to compare all of the circuits in terms of the losses caused by the Q-factor of the inductors, and the on-state voltage drop of the diodes present in the main conduction paths. The intention of the analysis was to demonstrate relative performance figures in order to compare circuits operated under ideal mechanical conditions. The analysis does not consider any effect upon the mechanical source vibration caused by the extraction of energy, i.e. the coupling between the electrical side and the mechanical side is assumed to be low. Since the aim of the work is to increase this electromechanical coupling, there is an inherent contradiction whereby the more successful a power-extraction circuit can be made, the less valid the analytical framework will be in estimating its effectiveness due to the increased damping this would represent. Any damping upon the mechanical motion would decrease the magnitude of the vibration, and hence decrease the power output, meaning that the quoted figures will all tend to over-estimate the power output. The relative performance terms may stay consistent over a wider operating range, but the non-linear effects of diodes means that the Q-factor alone is insufficient to rank the devices precisely.

Since the mechanical vibration source of a bluff body experiencing fluid-induced vibration in a fluid flow would likely oscillate in the 0-10 Hz frequency range [40], its lack of stiffness may mean that for this application the effect of the increased damping is higher than for some other vibration sources, such as combustion engines, where the materials have a higher resonant frequency and hence tend to be stiffer.

The work presented work focused primarily on the electronic interface circuits, further work is required to maximise the flow-induced vibration source with an optimal bluff body design for a wide range of flow velocities that occur in operational systems. Optimal coupling between the bluff body design and the piezoelectric material also requires careful consideration.

8.2 Pre-Biasing Experimental Programme

The experimental programme showed that both pre-biasing circuits are able to generate more power than the resistive load. However, the amount of harvested power was still below the amount predicted by the analytical expressions. The difference was accounted for by the losses in the components used, and difficulty estimating and measuring the losses, particularly the Q factor of the inductors used. Lower-loss experimental implementations are required in order to accurately validate the model with a high degree of accuracy.

8.3 Data Management

The query language designed for this work was limited in the range of operators and functions that were provided, and is not intended to be a complete toolkit of functions of use to a fluid dynamicist as that was outside the scope of this work. Many of the commercial analysis packages contain an extensive standard library of such functions. Rather than providing a complete set of functions and operators, representative examples were used of each of the broad classes of functions that a system might encounter. For example, the aggregators `min`, `max`, `sum` and `count` were provided as these can express a great number of query concepts and this was deemed sufficient for representative testing, although in practice other operations would be needed in a production

system. The prototyped research database presented in this thesis is able to execute user-defined queries using a distributable, *MapReduce*-style framework. The implementation of domain-specific functions and toolkits using this mechanism is left as future development work.

8.3.1 Scalability

In Chapter 3 it was outlined that in order to be considered truly scalable a distributed system should have no single point of failure, and the failure of a single node should never cause data loss. The prototype database system described in this thesis achieves those aims, but the implementation could be improved. Currently, each distinct sample stream is assigned a different node as its master, and that master node coordinates all reads and writes to that stream amongst the other nodes in the cluster. If a request arrives to a node that is not the master, then that node looks up the master and transparently redirects the request to the proper node. In most distributed database architectures, such as *Apache HBase*, it is possible to read and write any key (row) from any node in the cluster transparently, directly to one of the replicas where the data is stored, without having to involve the master. The implication of this is that when a node fails, no reads or writes are possible for the streams for which it was the master until the cluster has healed itself, and elected new masters for the affected streams, and this behaviour was evident in the simulated failure test of section 3.12.5. The distributed system is presently partition-intolerant, i.e., in the case of a partition and a failed master for a particular data stream, the system may incorrectly decide to make another node the master. Although many distributed database systems have this flaw, the failure mode should be better defined.

8.3.2 Query Optimisation

Relational databases have highly-optimised query executors, that are able to perform a wide range of query structure transformations in order to execute queries faster. Because the SQL language is declarative but supports a broad class of operations and relational queries, and little is known by the implementation about the semantics of the data and queries, query optimisation is an important factor in the performance of a relational database. The query optimisation strategy

used in the prototype database presented in this thesis relies primarily on elimination of disk access by the use of indexes, which are computed based on the functions and operators that are known to the system. There is a broad class of operations for which the performance of the prototype database system could be increased even further by advanced optimisation techniques. A way to express the additional semantics of certain operators would likely yield a significant improvement in performance for a broad range of queries.

Appendices

Appendix A

Pre-biasing experimentation PCB Layout

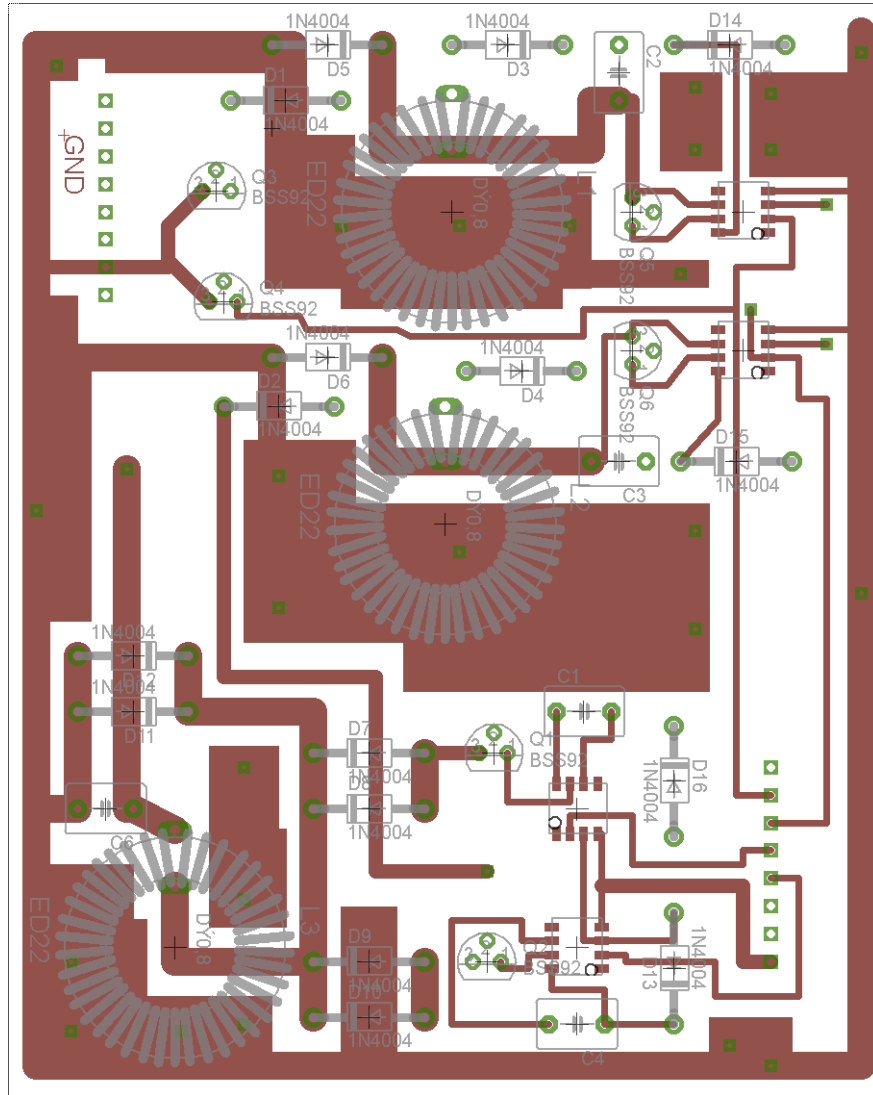


Figure A.1: Top of PCB layout for first embodiment of Pre-Biasing Circuit, designed using CadSoft EAGLE Version 5.6.0.

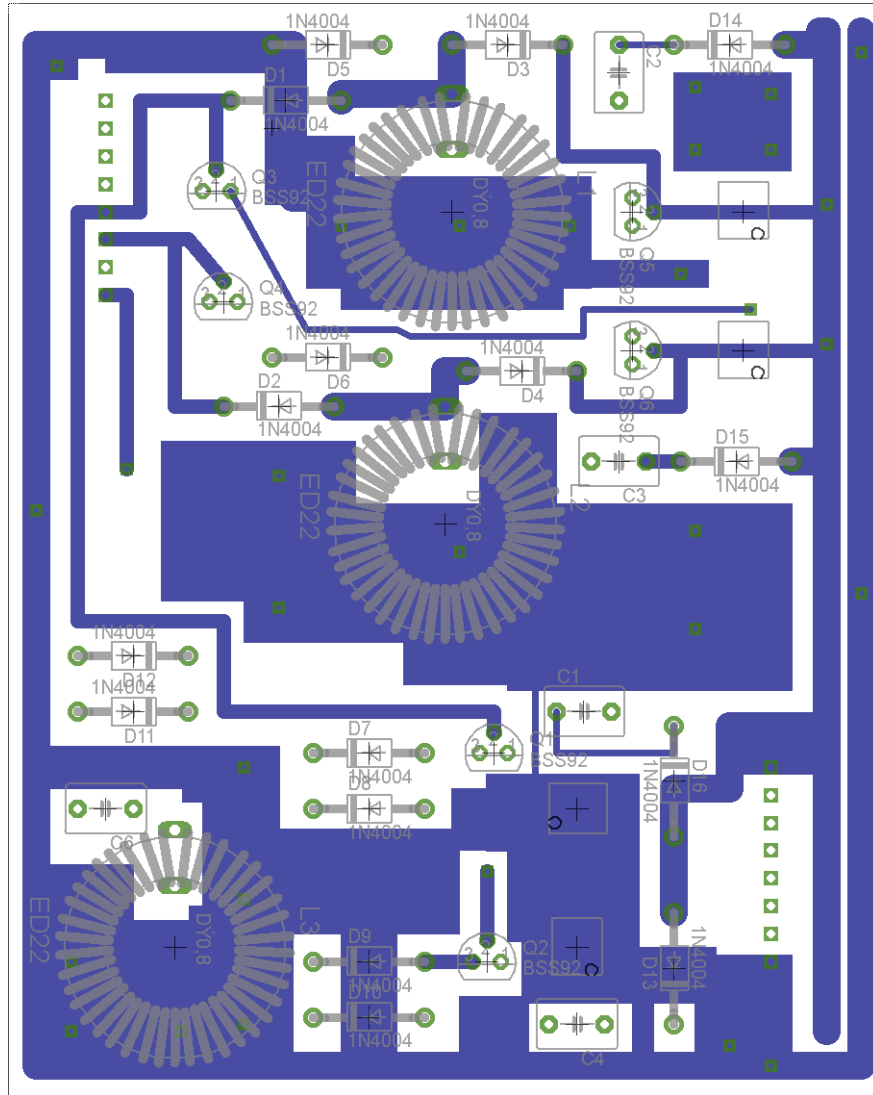


Figure A.2: Reverse of PCB layout for first embodiment of Pre-Biasing Circuit, designed using CadSoft EAGLE Version 5.6.0.

Appendix B

Protocol Definitions

All messages exchanged between nodes, and between clients and a given node, are HTTP messages. JSON is used for responses where a response is specified, otherwise a blank response with a 200 status code is an acknowledgement. Status code 500 is used for processing error, and a code in the range 400-499 is used for an error with data supplied by the client.

B.1 Client to Node

Persist new samples

POST /source/{sourceId}/append?syncMode={none|one|full}

Param 'sync': whether or not to ensure data is fully replicated before returning. Value of **none** means data is only buffered in memory on the recipient, value of **one** means it is written to a single disk, and **full** means it is written to all node that form the replication set. Response: 200 on success (empty reply), plaintext on non-200.

Note that in all cases where the **expectedVersion** parameter is required, this number is checked against the local version. If a mismatch is found, then an 4XX error is returned.

Client Query Execute

GET /query?q={queryString}

Response: JSON as follows:

```
{
  "timeMs": (time taken in ms)
  "type": BOOLEAN | DOUBLE | TIME_RANGES | SAMPLES
  "value": (query result)
}
```

B.2 Node to Node

Update frame

POST /source/{sourceId}/frame/{frameId}/{newVersionId}
?numSamples={numSamples}&expectedVersion

Payload: binary object containing samples to be appended as 64-bit long, followed by 64-bit double.

Read frame

GET /source/{sourceId}/frame/{frameId}/data?version={expectedVersion}

Response: binary object containing raw samples

Delete frame

DELETE /source/{sourceId}/frame/{frameId}

Response: 200 if OK, non-200 with plaintext payload on error.

Node to master: Update frame result (callback)

`/source/{sourceId}/frame/{frameId}?newVersion={newVersion}`

Appendix C

Random Dataset Generator

It is useful to have a dataset that can be generated quickly, and which has the distribution of values that might be expected in a real physical system. Purely random values can be used, but these typically do not resemble real data. Data points in nature typically do not differ largely from their neighbours, so a generator was developed to emulate real-world data generation schemes. Each iteration, a randomly-generated value with gaussian distribution is generated and added to the previous value. This is then multiplied by a value which is very close to, but less than 1, to keep the summation close to 0. A Java implementation of the meandering gaussian is as follows:

```
java.util.Random random = new java.util.Random();
random.setSeed(...);
double value = 0;
double next() {
    value += random.nextDouble() - 0.5;
    value /= 1.0000000001;
    return value;
}
```

To generate the dataset used in tests, the seed was 2854145, and a sample was output every 0.25 seconds between the years 2000 and 2001 for a total of 1261440000 values.

Bibliography

- [1] M. Heitzer, “Plastic limit loads of defective pipes under combined internal pressure and axial tension,” *International Journal of Mechanical Sciences*, vol. 44, no. 6, pp. 1219 – 1224, 2002, Seventh International Symposium on Structural Failure and Plasticity.
- [2] B. Jung and B. Karney, “Systematic surge protection for worst-case transient loadings in water distribution systems,” *Journal of Hydraulic Engineering*, vol. 135, no. 3, pp. 218–223, 2009.
- [3] R. Truss and A. Vale, “Understanding brittle failure of uPVC (unplasticised polyvinyl chloride) pipe,” *Pure and Applied Chemistry*, vol. 57, no. 7, pp. 993–1000, 1985.
- [4] M. B. Barker, J. Bowman, and M. Bevis, “The performance and causes of failure of polyethylene pipes subjected to constant and fluctuating internal pressure loadings,” *Journal of Materials Science*, vol. 18, pp. 1095–1118, 1983, 10.1007/BF00551979.
- [5] G. McKenna and R. Penn, “Time-dependent failure in poly(methyl methacrylate) and polyethylene,” *Polymer*, vol. 21, no. 2, pp. 213 – 220, 1980.
- [6] H. Ramos and A. Borga, “Surge effects in pressure systems for different pipe materials,” in *Advances in Water Resources and Hydraulic Engineering*. Springer Berlin Heidelberg, 2009, pp. 2152–2156.
- [7] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, and M. Csail, “Pipenet: A wireless sensor network for pipeline monitoring,” in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. IEEE, 2007, pp. 264–273.

- [8] P. Storli and T. Nielsen, “Transient Friction in Pressurized Pipes. I: Investigation of Zielke’s Model,” *Journal of Hydraulic Engineering*, vol. 137, no. 5, pp. 577–584, 2011.
- [9] D. Misiunas, M. Lambert, A. Simpson, and G. Olsson, “Burst detection and location in water distribution networks,” *Water Science & Technology: Water Supply*, vol. 5, no. 3, pp. 71–80, 2005.
- [10] D. Covas, I. Stoianov, H. Ramos, N. Graham, F. Maksimovic, and D. Butler, “Water hammer in pressurized polyethylene pipes: conceptual model and experimental analysis,” *Urban Water Journal*, vol. 1, no. 2, pp. 177–197, 2004.
- [11] B. Sharp and D. Sharp, *Water hammer: practical solutions*. Butterworth-Heinemann, 1995.
- [12] C. C. Bonin, “Water-hammer damage to oigawa power station,” *Journal of Engineering for Power*, vol. 82, no. 2, pp. 111–116, 1960.
- [13] C. SCHMITT, G. PLUVINAGE, E. HADJ-TAIEB, and R. AKID, “Water pipeline failure due to water hammer effects,” *Fatigue & Fracture of Engineering Materials & Structures*, vol. 29, no. 12, pp. 1075–1082, 2006.
- [14] S. Cook, “Erosion by water-hammer,” *Proceedings of the Royal Society of London. Series A*, vol. 119, no. 783, pp. 481–488, 1928.
- [15] EDF Energy. (2009) Factsheet: A guide to reactive power. [Online]. Available: <http://www.edfenergy.com/products-services/large-business/PDF/MBC-FS-GRP-002-0709.pdf>
- [16] B. Sweeney, “Pressure reducing valve,” Jan. 3 1933, uS Patent 1,893,254.
- [17] S. Buchberger and G. Nadimpalli, “Leak estimation in water distribution systems by statistical analysis of flow readings,” *Journal of Water Resources Planning and Management*, vol. 130, no. 4, pp. 321–329, 2004.
- [18] S. Srirangarajan, M. Iqbal, H. B. Lim, M. Allen, A. Preis, and A. J. Whittle, “Water main burst event detection and localization,” *12th Annual Conference on Water Distribution Systems Analysis (WDSA)*, Tucson, Arizona, United States, September 12-15, 2010, 2010.
- [19] O. H. W. T. Chu, “Acoustical characteristics of leak signals in plastic water distribution pipes,” *Applied Acoustics*, vol. 58, pp. 235–254, 1999.

- [20] H. Zhou, I. Stoianov, C. Maksimovic, and N. Graham, "Power consumption in gprs data logger for water distribution network monitoring," *Imperial College London*.
- [21] M. Al Ahmad and H. N. Alshareef, "Modeling the power output of piezoelectric energy harvesters," *Journal of Electronic Materials*, vol. 40, no. 7, pp. 1477–1484, Jul 2011.
- [22] N. S. Shenck, "A demonstration of useful electric energy generation from piezoceramics in a shoe," 1999.
- [23] D. Guyomar, A. Badel, E. Lefeuvre, and C. Richard, "Toward energy harvesting using active materials and conversion improvement by nonlinear processing," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 52, no. 4, pp. 584–595, Apr 2005.
- [24] F. M. Discenzo, D. Chung, and K. A. Loparo, "Pump condition monitoring using self-powered wireless sensors," *Sound and Vibration*, vol. 40, no. 5, pp. 12–15, 2006.
- [25] Storage Review. (2010) Western Digital Caviar Black Review (2TB). [Online]. Available: http://www.storagereview.com/western_digital_caviar_black_review_2tb
- [26] Y. Deng, "What is the future of disk drives, death or rebirth?" *ACM Computer Survey*, vol. 43, no. 3, pp. 23:1–23:27, Apr. 2011.
- [27] C. Knight, J. Davidson, and S. Behrens, "Energy options for wireless sensor nodes," *Sensors*, vol. 8, no. 12, pp. 8037–8066, 2008. [Online]. Available: <http://www.mdpi.com/1424-8220/8/12/8037>
- [28] C. Belhadj-Yahya, "Energy options for wireless sensors," in *Energy Conference and Exhibition (EnergyCon), 2010 IEEE International*, December 2010, pp. 564–569.
- [29] The ABB Group. (2011) ABB AquaProbe FEA100/FEA200 Product Manual. [Online]. Available: [http://www05.abb.com/global/scot/scot203.nsf/veritydisplay/0d29397c9d793bacc125790b00465599/\\$file/OI_FEA100_200-EN.pdf](http://www05.abb.com/global/scot/scot203.nsf/veritydisplay/0d29397c9d793bacc125790b00465599/$file/OI_FEA100_200-EN.pdf)
- [30] J. BAINES and V. NEWMAN, "Dinorwig pumped storage scheme. part 1: Design. part 2: Construction." *ICE Proceedings*, vol. 74, pp. 637–718(81), 1983. [Online]. Available: <http://www.icevirtuallibrary.com/content/article/10.1680/iicep.1983.1360>

- [31] G. Kokossalakis, “Acoustic data communication system for in-pipe wireless sensor networks,” *Massachusetts Institute of Technology. Dept. of Civil and Environmental Engineering.*, 2006.
- [32] S. Pobering and N. Schwesinger, “Power supply for wireless sensor systems,” *Sensors*, 2008 *IEEE*, pp. 685–688, oct. 2008.
- [33] A. M. Gorlov and V. M. Silantyev, “Limits of the turbine efficiency for free fluid flow,” *Journal of Energy Resources Technology*, vol. 123, pp. 311–317, 2001.
- [34] CLA-VAL. (2012) CLA-VAL e-Power MP CLA-VAL Turbine. [Online]. Available: <http://www.cla-val.ch/Portals/0/News/LIN043E.pdf>
- [35] G. Ye and K. Soga, “Power harvesting for water distribution systems,” *Project NEPTUNE Technical Report - Non-public domain*, 2010.
- [36] P. D. Mitcheson, T. C. Green, E. M. Yeatman, and A. S. Holmes, “Architectures for vibration-driven micropower generators,” *Microelectromechanical Systems, Journal of*, vol. 13, no. 3, pp. 429–440, 2004.
- [37] G. R. S. Assi, “Mechanisms for ow-induced vibration of interfering bluff bodies,” Ph.D. dissertation, Department of Aeronautics Imperial College London, 2009.
- [38] M. V. Dyke, *Album of Fluid Motion*, 1st ed. Milton Van Dyke, 1982.
- [39] J. H. Gerrard, “The mechanics of the formation region of vortices behind bluff bodies,” *Journal of Fluid Mechanics*, vol. 25, pp. 401–413, 5 1966. [Online]. Available: http://journals.cambridge.org/article_S0022112066001721
- [40] P. W. Bearman, “Vortex shedding from oscillating bluff bodies,” *Annual Review of Fluid Mechanics*, vol. 16, no. 1, pp. 195–222, 1984.
- [41] T. Ghaoud and D. Clarke, “Modelling and tracking a vortex flow meter signal,” *Flow Measurement and Instrumentation*, vol. 13, no. 9, pp. 103–117, 2002.
- [42] L. Tang, “Piezoelectric energy harvesting devices for low frequency vibration applications,” Ph.D. dissertation, Department of Mechanical Engineering, McGill University Montral, November 2007.

- [43] J. J. Allen, “Bluff body energy converter,” US Patent US 7 224 077 B2, May 29, 2007.
- [44] C. B. Carroll, “Energy harvesting eel us patent 6424079,” US Patent US 6 424 079, July 23, 2002.
- [45] D.-A. Wang and H.-H. Ko, “Piezoelectric energy harvesting from flow-induced vibration,” *Journal of Micromechanics and Microengineering*, vol. 20, no. 2, p. 025019, 2010. [Online]. Available: <http://stacks.iop.org/0960-1317/20/i=2/a=025019>
- [46] J. R. Morison, J. Johnson, and S. Schaaf, “The force exerted by surface waves on piles,” *Journal of Petroleum Technology*, vol. 2, no. 5, pp. 149–154, 1950.
- [47] M. Hayakawa, “Electric wristwatch with generator,” US Patent 5 001 685, March, 1991.
- [48] L. Solymar and D. Walsh, *Lectures on the Electrical Properties of Materials 5th edition*. Walton Street, Oxford, UK: Oxford University Press, 1993.
- [49] D. Shen, “Piezoelectric energy harvesting devices for low frequency vibration applications,” Ph.D. dissertation, Graduate Faculty of Auburn University, 2009.
- [50] S. Pobering and N. Schwesinger, “A novel hydropower harvesting device,” in *ICMENS '04: Proceedings of the 2004 International Conference on MEMS, NANO and Smart Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 480–485.
- [51] J. Liang and W. Liao, “Energy flow in piezoelectric energy harvesting systems,” *Smart Materials & Structures*, vol. 20, no. 1, Jan 2011.
- [52] Piezo Systems Inc. (2013) High-Performance Piezoelectric Bending Actuators. [Online]. Available: <http://www.piezo.com/prodbm2highperf.html>
- [53] International Organization for Standardization. (1999) Information technology - Database languages, SQL Part 2: Foundation (SQL/Foundation). [Online]. Available: http://www.iso.org/iso/catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26197
- [54] Forrester Research. (2008) Market Update on Open Source Databases. [Online]. Available: <http://www.mysql.com/why-mysql/marketshare>

- [55] D. Nagy, A. M. Yassin, and A. Bhattacharjee, “Organizational adoption of open source software: barriers and remedies,” *Commun. ACM*, vol. 53, no. 3, pp. 148–151, Mar. 2010.
- [56] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 4:1–4:26, Jun. 2008.
- [57] J. Dean, “Designs, lessons and advice from building large distributed systems,” in *presentation at Third ACM SIGOPS International Workshop, Big Sky, Montana*, 2009.
- [58] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
- [59] K. Hall, S. Gilpin, and G. Mann, “MapReduce/Bigtable for distributed optimization,” in *NIPS LCCC Workshop*, 2010.
- [60] P. Bhatotia, A. Wieder, I. E. Akkuş, R. Rodrigues, and U. A. Acar, “Large-scale incremental data processing with change propagation,” in *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, ser. HotCloud’11. Berkeley, CA, USA: USENIX Association, 2011, pp. 18–18.
- [61] D. Peng and F. Dabek, “Large-scale incremental processing using distributed transactions and notifications,” in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, 2010, pp. 1–15.
- [62] Amazon Web Services, Inc. (2013) Amazon EC2 Pricing - Small Instance. [Online]. Available: <http://aws.amazon.com/ec2/pricing/>
- [63] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, “Aurora: a new model and architecture for data stream management,” *The VLDB Journal*, vol. 12, pp. 120–139, 2003, 10.1007/s00778-003-0095-z.
- [64] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina *et al.*, “The design of the borealis stream processing engine.” CIDR, 2005.

- [65] M. Balazinska, H. Balakrishnan, and M. Stonebraker, “Load management and high availability in the medusa distributed stream processing system,” in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* ACM, 2004, pp. 929–930.
- [66] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, “Query processing, resource management, and approximation in a data stream management system.” CIDR, 2003.
- [67] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “Tinydb: An acquisitional query processing system for sensor networks,” *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, 2005.
- [68] W. Cody, J. Coonen, D. Gay, K. Hanson, D. Hough, W. Kahan, R. Karpinski, J. Palmer, F. Ris, and D. Stevenson, “A proposed radix- and word-length-independent standard for floating-point arithmetic,” *Micro, IEEE*, vol. 4, no. 4, pp. 86–100, August 1984.
- [69] N. Pansare, V. Borkar, C. Jermaine, and T. Condie, “Online aggregation for large mapreduce jobs,” *PVLDB*, vol. 4, no. 11, pp. 1135–1145, 2011.
- [70] K. Thompson, “Unix implementation,” *The Bell System Technical Journal*, vol. 57, no. 6, pp. 1931–1946, 1978.
- [71] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
- [72] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.
- [73] D. Skeen and M. Stonebraker, “A formal model of crash recovery in a distributed system,” *Software Engineering, IEEE Transactions on*, vol. SE-9, no. 3, pp. 219 – 228, may 1983.
- [74] L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998.
- [75] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency control and recovery in database systems.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1987.

- [76] T. Haerder and A. Reuter, “Principles of transaction-oriented database recovery,” *ACM Computing Surveys*, vol. 15, pp. 287–317, 1983.
- [77] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, “Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, ser. STOC ’97. New York, NY, USA: ACM, 1997, pp. 654–663.
- [78] R. Gallager, P. Humblet, and P. Spira, “A distributed algorithm for minimum-weight spanning trees,” *ACM Transactions on Programming Languages and systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.
- [79] The Open Group Base Specifications Issue 6. (2004) IEEE Std 1003.1, 2004 Edition. [Online]. Available: <http://pubs.opengroup.org/onlinepubs/009695399/functions/rename.html>
- [80] J. C. Reynolds, “The discoveries of continuations,” *LISP and Symbolic Computation*, vol. 6, pp. 233–247, 1993, 10.1007/BF01019459.
- [81] D. P. Mehta and S. Sahni, *Handbook of Datastructures and Applications*. Chapman & Hall/CRC, 2004, ch. Analysis of Algorithms, pp. 9–15.
- [82] M. Bhadkamkar, F. Farfan, V. Hristidis, and R. Rangaswami, “Storing semi-structured data on disk drives,” *Trans. Storage*, vol. 5, no. 2, pp. 6:1–6:35, Jun. 2009.
- [83] R. A. Finkel and J. L. Bentley, “Quad trees a data structure for retrieval on composite keys,” *Acta Informatica*, vol. 4, pp. 1–9, 1974, 10.1007/BF00288933.
- [84] E. W. Dijkstra, “Algol 60 Translation : An Algol 60 Translator For The X1 And Making A Translator For Algol 60,” CWI Technical Report Stichting Mathematisch Centrum. Rekenafdeling-MR 34/61, 1961, (ALGOL Bulletin, 1).
- [85] S. P. Beeby, R. N. Torah, M. J. Tudor, P. Glynne-Jones, T. O’Donnell, C. R. Saha, and S. Roy, “A micro electromagnetic generator for vibration energy harvesting,” *Journal of Micromechanics and Microengineering*, vol. 17, no. 7, p. 1257, 2007.
- [86] H. A. Sodano, D. J. Inman, and G. Park, “A review of power harvesting from vibration using piezoelectric materials,” *The Shock and Vibration Digest*, vol. 36, no. 3, pp. 197–205, 2004.

- [87] P. D. Mitcheson, T. Sterken, C. He, M. Kiziroglou, E. M. Yeatman, and R. Puers, “Electrostatic microgenerators,” *Measurement and Control UK*, vol. 41, no. 4, pp. 114–119, 2008.
- [88] Y. Suzuki, D. Miki, M. Edamoto, and M. Honzumi, “A mems electret generator with electrostatic levitation for vibration-driven energy-harvesting applications,” *Journal of Micromechanics and Microengineering*, vol. 20, no. 10, p. 104002, 2010.
- [89] T. Sterken, P. Fiorini, K. Baert, G. Borghs, and R. Puers, “Novel design and fabrication of a mems electrostatic vibration scavenger,” *The 4th Int. Workshop on Micro and Nanotechnology for Power Generation and Energy Conversion Applications*, 2004.
- [90] E. S. Leland and P. K. Wright, “Resonance tuning of piezoelectric vibration energy scavenging generators using compressive axial preload,” *Smart Materials and Structures*, vol. 15, no. 5, p. 1413, 2006.
- [91] L. Blystad and E. Halvorsen, “A piezoelectric energy harvester with a mechanical end stop on one side,” *Design Test Integration and Packaging of MEMS/MOEMS (DTIP), 2010 Symposium on*, vol. 17, no. 4, pp. 505–511, Apr 2011.
- [92] P. D. Mitcheson and T. T. Toh, *Energy Harvesting for Autonomous Systems*. Artech House Publishing, 2010, ch. Power Management Electronics, pp. 159–209.
- [93] S. D. Senturia, *Microsystem Design*, 2nd ed. Springer Science+Business Media, 2001.
- [94] G. Ottman, H. Hofmann, A. Bhatt, and G. Lesieutre, “Adaptive piezoelectric energy harvesting circuit for wireless remote power supply,” *Power Electronics, IEEE Transactions on*, vol. 17, no. 5, pp. 669 – 676, Sep 2002.
- [95] E. Lefeuvre, A. Badel, L. Petit, C. Richard, and D. Guyomar, “Semi-passive piezoelectric structural damping by synchronized switching on voltage sources,” *Journal of Intelligent Material Systems and Structures*, vol. 17, no. 8-9, pp. 653–660, 2006.
- [96] D. Guyomar, C. Magnet, E. Lefeuvre, and C. Richard, “Nonlinear processing of the output voltage of a piezoelectric transformer,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 53, no. 7, pp. 1362–1375, 2006.

- [97] N. Krihely and S. Ben-Yaakov, "Self-contained resonant rectifier for piezoelectric sources under variable mechanical excitation," *Power Electronics, IEEE Transactions on*, vol. 26, no. 2, pp. 612–621, Feb 2011.
- [98] Y. C. Shu, I. C. Lien, and W. J. Wu, "An improved analysis of the SSHI interface in piezoelectric energy harvesting," *Smart Materials and Structures*, vol. 16, no. 6, p. 2253, 2007.
- [99] D. Niederberger and M. Morari, "An autonomous shunt circuit for vibration damping," *Smart Materials and Structures*, vol. 15, no. 2, p. 359, 2006.
- [100] E. Lefeuvre, A. Badel, C. Richard, L. Petit, and D. Guyomar, "A comparison between several vibration-powered piezoelectric generators for standalone systems," *Sensors and Actuators A: Physical*, vol. 126, no. 2, pp. 405–416, 2006.
- [101] L. Garbuio, M. Lallart, D. Guyomar, C. Richard, and D. Audigier, "Mechanical energy harvester with ultralow threshold rectification based on sshi nonlinear technique," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 4, pp. 1048–1056, april 2009.
- [102] M. Lallart, C. Richard, L. Garbuio, L. Petit, and D. Guyomar, "High efficiency, wide load bandwidth piezoelectric energy scavenging by a hybrid nonlinear approach," *Sensors and Actuators A: Physical*, vol. 165, no. 2, pp. 294–302, 2011.
- [103] M. Lallart, L. Garbuio, L. Petit, C. Richard, and D. Guyomar, "Double synchronized switch harvesting (dssh): a new energy harvesting scheme for efficient energy extraction," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 55, no. 10, pp. 2119–2130, october 2008.
- [104] H. Shen, J. Qiu, H. Ji, K. Zhu, and M. Balsi, "Enhanced synchronized switch harvesting: a new energy harvesting scheme for efficient energy extraction," *Smart Materials and Structures*, vol. 19, no. 11, p. 115017, 2010.
- [105] K. Makihara, J. Onoda, and T. Miyakawa, "Low energy dissipation electric circuit for energy harvesting," *Smart Materials and Structures*, vol. 15, no. 5, p. 1493, 2006.

- [106] J. Dicken, P. D. Mitcheson, I. Stoianov, and E. M. Yeatman, "Increased power output from piezoelectric energy harvesters by pre-biasing," *PowerMEMS, Washington DC, USA, December 1-4*, pp. 75–78, 2009.
- [107] J. Dicken, P. D. Mitcheson, A. Elliott, and E. M. Yeatman, "Single-supply pre-biasing circuit for low-amplitude energy harvesting applications," *PowerMEMS, Seoul, Republic of Korea, November 15-18*, pp. 46–49, 2011.
- [108] H. Shen, J. Qiu, H. Ji, K. Zhu, M. Balsi, I. Giorgio, and F. dell'Isola, "A low-power circuit for piezoelectric vibration control by synchronized switching on voltage sources," *ArXiv e-prints*, Jul. 2010.
- [109] M. Lallart and D. Guyomar, "An optimized self-powered switching circuit for non-linear energy harvesting with low voltage output," *Smart Materials and Structures*, vol. 17, no. 3, p. 035030, 2008.
- [110] A. D. Elliott and P. D. Mitcheson, "Implementation of a single supply pre-biasing circuit for piezoelectric energy harvesters," *Procedia Engineering*, vol. 47, no. 0, pp. 1311 – 1314, 2012, <http://www.sciencedirect.com/science/article/pii/S1877705812044591>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877705812044591>