A FULL-CUSTOM DIGITAL-SIGNAL-PROCESSING UNIT FOR REAL-TIME CORTICAL BLOOD FLOW MONITORING

HONG ZHIQIAN
(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2009

ABSTRACT

Chairperson of the Supervisory Committee: Dr. Le Minh Thinh
Department of ECE

This thesis presents a full custom digital-signal-processing unit for real-time cortical blood flow monitoring. An evaluation of suitable algorithms using Laser Speckle Imaging statistical methods is presented from a theoretical perspective for practical implementations. All existing methods are found to be mathematically describing the same coefficient of variation but with different input samples and sample sizes. The simplest algorithm, Laser Speckle Contrast Analysis, is chosen to relax on the real-time imaging requirement.

Unlike normal imaging applications which require high speed and accuracy, biomedical imaging specifications are often relaxed to the minimum to achieve a low-power application. Consequently, CMOS sensors are evaluated and compared on their architectures that will eventually lead to the design of a low-power on-chip digital signal processing unit.

Numerous low-power digital techniques are discussed and applied on the design. These techniques include aggressive lowering of supply voltage close to or less than the sum of absolute device threshold, non pre-charged memory, clock-gating and pulse-latch clocking strategies. Performance is maintained through the use of bit-serial arithmetic units and these units include adder, multiplier, squarer, square-root and divider. This design is implemented in 0.35μm and a post-layout simulated power consumption of 887μW is achieved at a supply voltage of 1.2V while maintaining 30MHz at worst corner variation. This translates to approximately 1 million speckle contrast computations per second and a Figure of Merit of 962pW/fp.

ACKNOWLEDGMENTS

This master thesis has been carried out independently as a research programme in *National-University-of-Singapore* (NUS) and is supported by the faculty research committee grant (R-263-000-405-112 and R-263-000-405-133), Faculty of Engineering, NUS.

The author wishes to express sincere appreciation to the Department of Electrical and Computer Engineering in National University of Singapore for their financial support, supervisor Dr. Le Minh Thinh for his insights on Laser Speckle Imaging and acting co-supervisor Dr. Xu Yong Ping for his teachings in EE5507 Advanced Analog Integrated Circuit Design. Due to their previous research efforts in Laser Speckle Imaging and the approval of research grant, the existence of this thesis is promising.

The author will like to thank Dr. Heng Chun Huat for giving the most comprehensive introductory IC course in EE5507. His timely replies through forum and email to the questions posted by the author have been fruitful. Additional help is also given by Mr. Amit Bansal, graduate assistant of EE5507 and student of Dr. Heng, in his theoretical and practical advices on IC design and simulation tools. Appreciation is also given to Mr. Tan Kah Yong, student of Dr. Xu and ex-employee of STMicroelectronics, for his assistance on design methods and usage of simulation tools.

The author wouldalso like to acknowledge Mr. Teo Seow Miang for his help in computer setup, Ms. Zheng Huan Qun for enabling the Linux account and Mr. Kurt Van Genechten, ASIC MPW support Engineer from Europractice, for providing a comprehensive setup guide on Cadence and Mentor Calibre tools for the design kit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analog-to-digital converter
ALU	Arithmetic logic unit
ASIC	Application specific integrated circuit
APS	Active pixel sensor
BS	Bit-serial
CCD	Charged-coupled device
CDS	Correlated double sampling
CG	Clock gating
CM	Carry merge
CIS	CMOS Image sensor
CLA	Carry-look-ahead
CMOS	Complementary metal oxide semiconductor
CPL	Complementary pass-transistor logic
DA	Distributed arithmetic
DCT	Discrete cosine transformation
DNA	Deoxyribo nucleic acid
DPL	Double pass-transistor logic
DPS	Digital pixel sensor
DRC	Design rule check
DS	Digital serial
DSP	Digital signal processing
FFT	Fast Fourier Transform

FIR	Finite-length impulse response
FOM	Figure of merit
FPGA	Field programmable gate array
FPN	Fixed pattern noise
FPS	Frames per second
FSM	Finite state machine
HDL	Hardware description language
IC	Integrated circuit
K	Speckle contrast
LASCA	Laser speckle contrast analysis
LSB	Least significant bit
LSI	Laser speckle imaging
LTA	Loser take all
LVS	Layout versus schematic
MAC	Multiply accumulation
MCBS	Multi channel bit serial
MSB	Most significant bit
PD	Photo-diode
PG	Propagate generate
PMT	Photo multiplier tubes
PWM	Pulse width modulation
RF	Radio frequency
RTL	Register transfer level
SAFF	Sense amplifier flip-flops

SAR	Successive approximation register
SoC	System-on-chip
SRAM	Static random access memory
SRT	Square root
SS	Single slope
WTA	Winner-take-all

CHAPTER I

INTRODUCTION

Background

This individual research work is to investigate viable algorithms for visualizing and quantifying blood flows to be implemented as an I*ntegrated-Circuit* (IC) within a *System-on-Chip* (SoC) in a timeframe of two years. Outlined by the Principal Investigator, Dr. Le Minh Thinh, *Laser-Speckle-Imaging* (LSI) from [1] will only be considered for this research work. The targeted fabrication process is 0.35µm (AMIS-C035U/I3T25) [2], [3] and [4].

Motivation

System-on-chip has been widely demonstrated in the recent years to integrate various laboratory functionalities such as sensing, processing and actuation onto a single chip. A few of the recent applications include a bladder urine pressure sensor measurement system integrated with an *Application-Specific-Integrated-Circuit* (ASIC) die and a *Radio-Frequency* (RF) module [5], a droplet-based micro-fluidic biochip [6] and a charged-based capacitive sensor for *Deoxyribo-Nucleic-Acid* (DNA) detection and cells monitoring [7].

The use of *CMOS-Image-Sensor* (CIS) technology [8] has also been widely used in biomedical imaging devices, such as the bioluminescence detection lab-on-chip [9] and retina implant systems for patients suffering from vision illness [10], [11]. The main reason for these existences is that CIS technology can coexist with the same CMOS fabrication process and allows full integration of other analog/digital signal processing units and control circuits within the same chip [12]. These camera-on-a-chip miniaturizations have eventually lead to low-power, cost-effective and portable implementations and deemed very suitable to replace high-powered and expensive CCDs or *Photo-Multiplier-Tubes* (PMTs) biomedical devices [13].

A typical setup for LSI relies on the use of a *Charge-Coupled-Device* (CCD) camera to capture the reflected light from the illuminated tissue to produce the speckle pattern. The speckle pattern is then analyzed on an offline computer. A typical quasi real-time monitoring embedded implementation will include a high quantum efficiency CCD camera to acquire the speckle pattern and a high performance digital signal processor to perform image processing algorithms. Alternatively, a camera-on-chip system will provide an attractive alternative to both the embedded implementation and the present setup of using a CCD-Desktop-Matlab combination [1].

Limitations

According to the Samsung CIS Roadmap, better fabrication processes (0.18µm-0.09µm) are already in existence at the beginning of this work (2008) [14]. The use of an older technology immediately puts some limitation when approaching the design problem. For example, new design methods used in deep-submicron technology to reduce leakage current might not be appropriate when apply on older technology. Realistic specifications must be set within the performance of older technology as compared to the state-of-the-art CIS research work. However, there will be a wider knowledge of design methods existing in literature and a more mature understanding of design limitation in a 0.35µm process.

In a fast value iteration IC design work flow, the availability of the appropriate design tools and knowledge of these tools are also one of the four key contributions for project reusability [15]. There is a need for "designer reuse" where designers get to share the how's and the tricks of the tools, where one can ideally abstract away the need to drive the tools [15]. Without a collaborative team in the current environment, one has to place emphasis on allocating the time required to master the art of the tools. Although the school has a suite of Synopsys design tools for standard cell-based ASIC design, very little technical support is provided to enable the tools to work with the chosen design kit. Nonetheless, the ASIC support engineer at Europractice, Mr. Kurt Van Genechten, is able to provide instructions to setup the beta version of the design kit to work with Cadence Virtuoso Schematic Editor, Layout Editor, and Spectre Simulator for design and Mentor Calibre for physical design verification focusing on analog and custom design.

In a realistic SoC project, a valid workflow model is used as a roadmap for planning and execution [15]. A case study finds that a full SoC project covering all design aspects is completed in 20 weeks (5 months) by a group of 11 experienced IC design engineers [15]. Without such a comprehensive design team, it is thus required to narrow down the focus for an individual work. The focus of the project is now reduced to the second key point of project reusability in a fast value iteration IC design workflow [15] - the construction recipes of the *Digital-Signal-Processing* (DSP) unit implementing the processing algorithm using custom digital design methods. By focusing on the design methods of the processing algorithm, it will make a more significant contribution for future development.

In addition, CMOS sensors are already a mature product in 0.35μm process and are widely available in the market. The third key point of project reusability, floor planning [15], is thus easily available from existing literature. The fourth key point of using a standard design environment will not pose a problem in an individual work. With a good understanding of the limitations, one can then address the challenges by planning a strategy and defining a research design problem with the available and appropriate tools.

Definition

The objective of this master thesis is to evaluate the suitability of the different LSI algorithms for the design and implementation of a DSP unit to be used in conjunction with an existing CMOS image sensor for a SoC-based integration. The design and implementation of the DSP unit includes and up to the full layout using the available custom design tools that have been setup to work with the defined design kit.

Achievement

Numerous low-power digital techniques are discussed and applied on the design. These techniques include aggressive lowering of supply voltage close to or less than the sum of absolute device threshold, non pre-charged memory, clock-gating and pulse-latch clocking strategies. Performance is maintained through the use of bit-serial arithmetic units and these units include adder, multiplier, squarer, square-root and divider. This design is implemented in 0.35μm and a post-layout simulated power consumption of 887μW is achieved at a supply voltage of 1.2V while maintaining 30MHz at worst corner variation. This translates to approximately 1 million speckle contrast computations per second and a FOM of 962pW/fp.

Organization

The rest of the thesis is organised as follows. In Chapter 2, a literature review of LSI algorithms [1], existing CMOS image sensors, and low power digital design techniques are reported. In Chapter 3, the design solution and methodology are presented. This chapter also includes the formulation of the research work based on the literature review in

Chapter 2. LSBF and MSBF arithmetic blocks are discussed in Chapters 4 and 5 respectively. Chapter 6 describes the design and simulation work of the DSP unit from a top-level perspective, including the state controller, memory interface and the clock strategy. Chapter 7 then concludes with the design results that are presented in the previous chapters.

CHAPTER II

LITERATURE REVIEW

This chapter briefly discuss the four LSI algorithms outlined in [1]. A review of existing CMOS image sensors architectures and their SoC implementations are provided for a deeper understanding. This chapter then ends of with some of the existing and suitable low power digital design techniques.

Laser Speckle Imaging

Among the existing methods of blood flow monitoring, one special technique, *Laser-Speckle-Imaging* (LSI), has been used extensively in medical research to achieve the viability of real-time medical imaging [1]. This technique uses an imaging device to capture the reflected light of a low-power laser shining on an object. A typical experimental setup, consisting of a laser diode, a monochrome CCD camera and a rodent, is shown below in Figure 1.

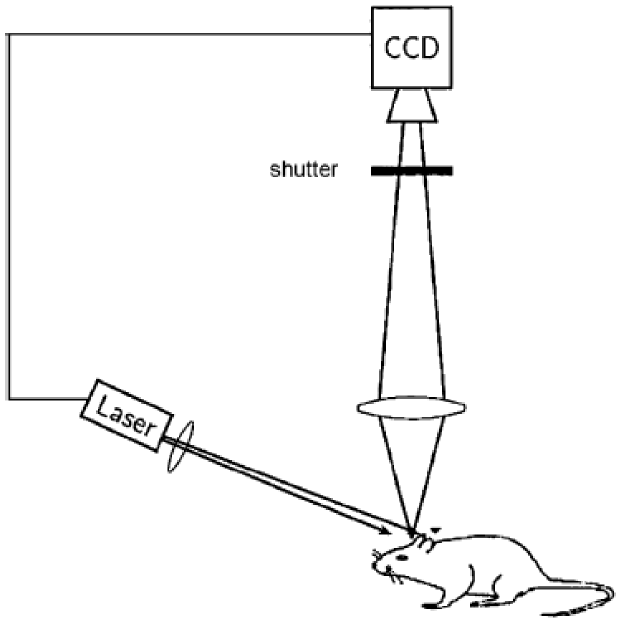


Figure 1. Experimental setup of speckle imaging of cerebral blood flow [1].

When the collimated laser light is scattered from the surface of the rodent, a random interference pattern is captured by the CCD. Although this grainy raw speckle pattern contains no useful information, it is known that when there are movements of blood cells, the speckle pattern that is produced changes. These speckles remain correlated with short movements and de-correlate with long movements [17]. By applying statistical methods, blood flow images made up of speckle contrast can then be obtained from the raw speckle pattern. The most fundamental transformation process is identified as *Laser-Speckle-Contrast-Analysis* (LASCA) where the speckle contrasts are derived from the spatial information of the raw speckle images [17]. A few other variants have also been identified and compared in [1] as sLASCA (spatially derived contrast using temporal frame averaging), modified laser speckle imaging (mLSI) and tLASCA (temporally derived contrast). Among the methods, tLASCA has out-performed the rest in terms of processing speed and contrast with better subjective and objective evaluations of images [1]. A brief summary of the different statistical methods are reviewed below for completeness and better understanding of the thesis.

LASCA

$$K_{(x,y)} = \sigma_I / \bar{I} \tag{1}$$

$$\bar{I} = \frac{1}{N}\sum_{i=1}^{N} I_i \tag{2}$$

$$\sigma_I = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(I_i - \bar{I}\right)^2} \tag{3}$$

The most fundamental transformation defines *Speckle Contrast* (1), $K_{(x,\,y)}$, as ratio of its spatial standard deviation (3), σ_I, to its spatial mean intensity (2), $\bar{I}$, of a window of pixels isolated from the raw speckle pattern, where N is the number of pixels in the window of interest (e.g. a 5×5 window has 25 pixels) [17]. In statistics, this ratio is also known as the coefficient of variation which measures the dispersion of probability distribution. $K_{(x, y)}$ then represents the pixel value at location (x, y) of the blood flow image defined by the centre of the window of interest.

sLASCA

$$\overline{K}_{(x,y)} = \frac{1}{N}\sum_{i=1}^{N} K_{(x,y)i} \tag{4}$$

An improvement over the basic technique was made in [18], where the derived speckle contrasts of an image in (1) are further averaged over a number of frames of flow images (4), where N is the number of frames and $K_{(x, y)\,i}$ is the speckle contrast located at (x, y) at the i-th frame.

mLSI

$$K_{(x,y)} = (\overline{I^2_{(x,y,t)}} - \overline{I_{(x,y,t)}}^2)/\overline{I_{(x,y,t)}}^2 \tag{5}$$

An alternative first-order temporal statistics of the time-integrated speckle pattern was also proposed in [19] where the $K_{(x, y)}$ is now defined as (5) where the intensities of the (x, y) pixels are averaged over a number of temporal frames instead of a spatial window (1).

tLASCA

$$\overline{K}_{(x,y)} = \frac{1}{W}\sum_{i=1}^{W} \sigma_{(x,y,t)} / \overline{I_{(x,y,t)}} \tag{6}$$

$$\overline{I_{(x,y,t)}} = \frac{1}{N}\sum_{t=1}^{N} I_{(x,y,t)} \tag{7}$$

$$\sigma_{(x,y,t)} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}\left(I_{(x,y,t)} - \overline{I_{(x,y,t)}}\right)^2} \tag{8}$$

The newest proposed method in [1] first calculates $K_{(x, y)}$ using (6) but with a ratio of temporal standard deviation (8), to its temporal mean intensities (7), of pixel (x, y) over a number of frames, where *N* defines the number of frames. The speckle contrasts are then further averaged over a spatial observation window (6), where *W* is the window size, to obtain a final speckle contrast. This speckle contrast now represents the pixel value at location (x, y) of the blood flow image defined by the centre of the window of interest.

CMOS Image Sensor

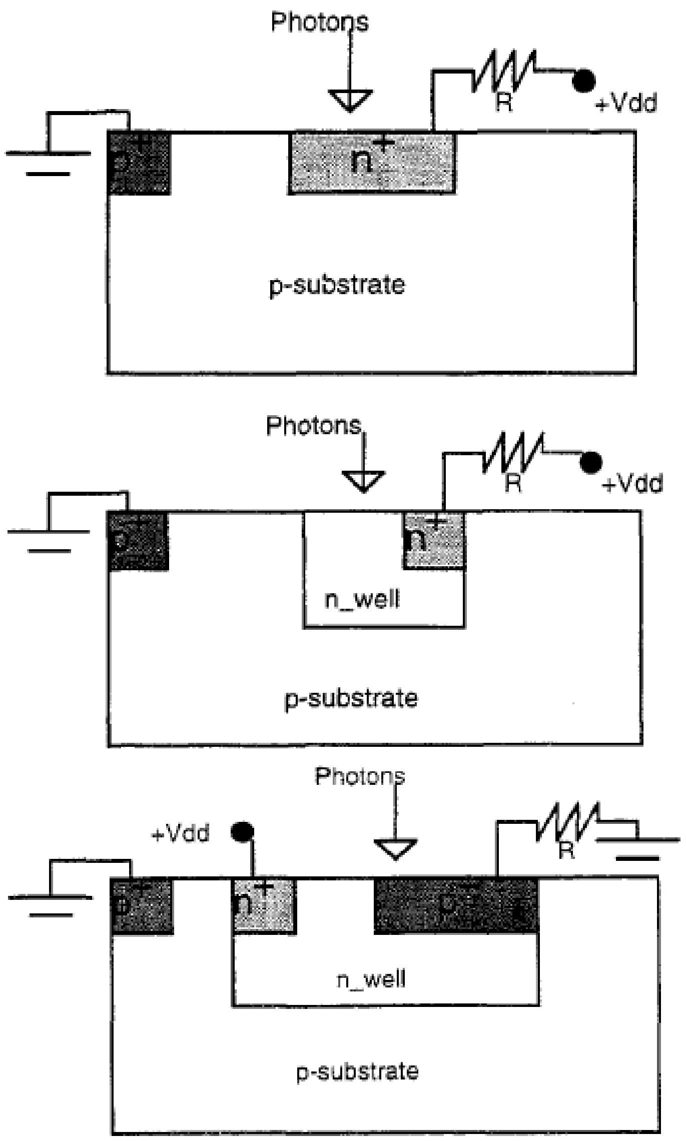


Figure 2. N+/Pwell, Nwell/Psub and P+/Nwell photodiode [20]

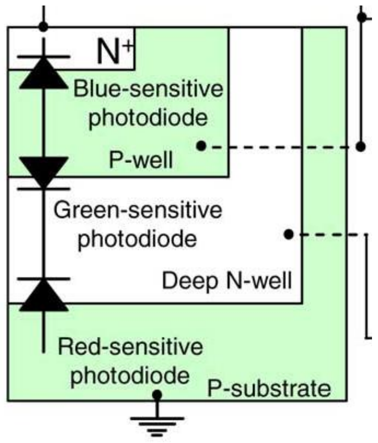


Figure 3. Triple well photodiode [21]

CMOS image sensors have been widely used in consumer products from optical mouse to high-end digital cameras [8]. With the advent of deep submicron CMOS, much more analog and digital processing are to be integrated within the same silicon die. This has brought CMOS image sensor into a new era of applications and also as a viable competitor to CCD technology. CMOS sensor uses a 2-dimension array of photodiodes to convert the input light intensity to electrical signals. In a standard CMOS process, a photodiode can be implemented as one of the following variants: N^+/P_{well}, N_{well}/P_{sub} or P^+/N_{well}, shown in Figure 2, where N_{well}/P_{sub} has a better green wavelength response, N_{well}/P_{sub} has a better infrared (longer wavelength) response and P^+/N_{well} is more useful when substrate isolated detectors are desired [20]. A more advanced triple-well CMOS process, shown in Figure 3, can even separate the colour component using vertically integrated photodiodes [22]. The principle operation of the photodiode has been left out intentionally due to its irrelevance in this work.

The electrical signals are then further read out using suitable readout architectures where the signals are converted to noise tolerant digital signals. Recent advancement in sensor technology has been focusing on the readout architectures in *Active Pixel Sensor*

(APS) [23], *Digital Pixel Sensor* (DPS) and camera-on-chip system integration applications [24] with results demonstrating ultra-high speed of 10,000 *frames per second* (fps) in DPS architecture [25], multi-mega pixel sensor with ultra-small pitch of 2μm in APS architecture [26], and highly-integrated sensor, processor and memory SoC [27].

Active Pixel Sensor

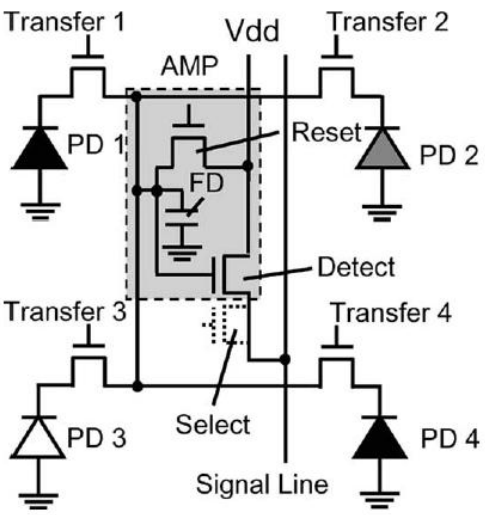


Figure 4. 1.5T/pixel voltage-mode pixel [26]

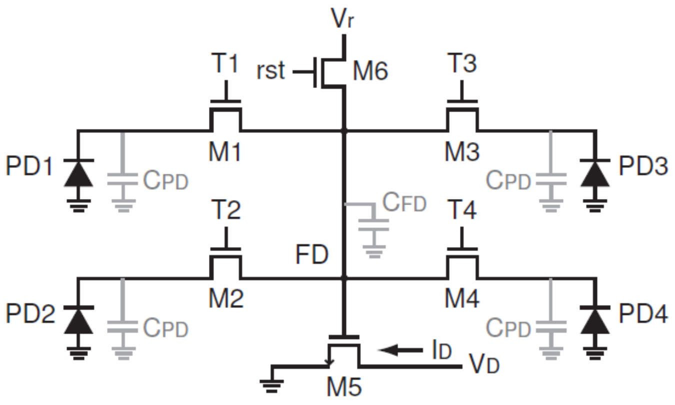


Figure 5. 1.5T/pixel current-mode pixel [28]

APS architecture exists in both voltage-mode, Figure 4, and current-mode, Figure 5. In both modes, the simplest form of APS pixel relies on a single transistor amplifier to isolate the sense node of the photodiode from the large column bus capacitance. This single transistor operates in saturation as a source follower in voltage-mode, while it operates in the linear region as a trans-conductance amplifier in current-mode. Historically, voltage-mode sensors have better noise performance, gain matching characteristics [28] and exhibit a higher linearity [29] while current-mode sensors are capable of operating at a lower voltage supply [30], more compatible to simple analog computations and able to scan at faster rates [31]. Although both modes show differences, they do exhibit similarities in the readout architectures, where [29] has integrated both modes onto the same readout architecture.

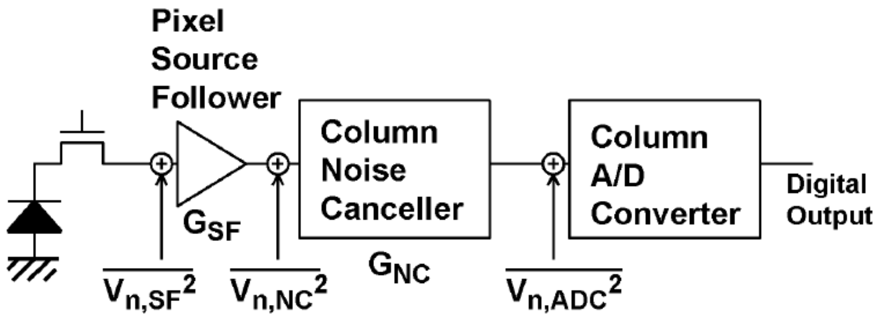


Figure 6. Signal readout chain for voltage-mode column-parallel architecture [32]

Noise cancellation is performed after amplifying the electrical signal, followed by an analog-to-digital conversion, Figure 6. This noise cancelling stage is also known as *Correlated-Double-Sampling* (CDS) and attempts to cancel the *Fixed-Pattern-Noise* (FPN) generated by threshold voltage variation [8]. The principle of operation of noise cancellation has also been left out due to its irrelevance in this work.

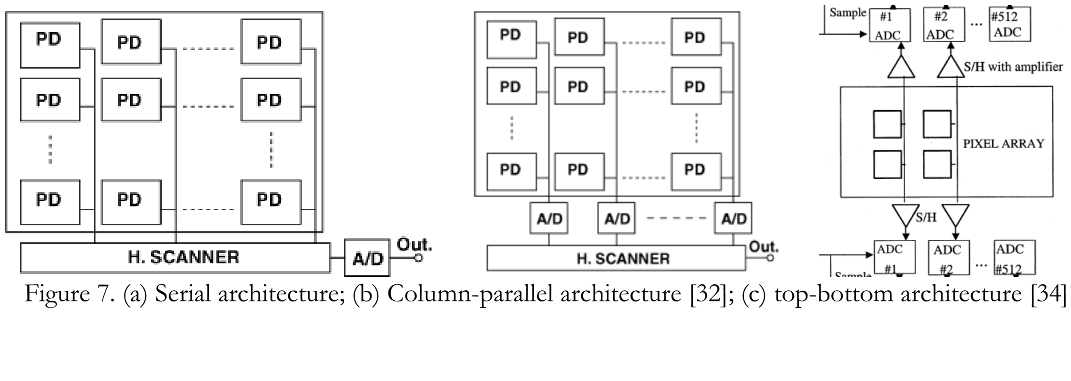


Figure 7. (a) Serial architecture; (b) Column-parallel architecture [32]; (c) top-bottom architecture [34]

The readout architectures are further classified into serial and column-parallel architecture, Figure 7. Both signal readout chains are similar with the exception that more *Analog-to-Digital Converters* (ADCs) are used in the column-parallel architecture as compared to a single global ADC in the serial architecture. In the latter, readouts are performed one row at a time in a rolling manner, i.e. top to bottom, delivering a faster rate compared to the single pixel serially read-out architecture. Common column-parallel ADC used are *Single-Slope* (SS) [33], *Successive-Approximation-Register* (SAR) [34], Cyclic [35], *Delta-Sigma* (Δ∑) [36] and *Pulse-Width-Modulation* (PWM) [37]. These ADCs are usually smaller and pitch matched to the pixel. Larger ADCs, such as [34], [35], can be split into a top-bottom architecture and sized two times bigger along the pixel pitch. They tend to operate slower compared to those ADC architectures used in a serial sensor where any high speed ADC is suitable. The principle of operation of ADC architecture is beyond the scope of this thesis, and thus is not discussed.

Digital Pixel Sensor

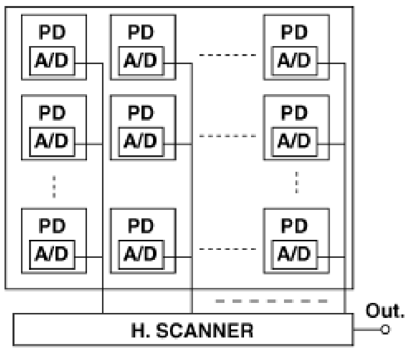


Figure 8. Digital pixel sensor architecture [32]

In DPS architecture, ADCs are integrated into individual pixel, Figure 8, enabling massive parallel analog-to-digital conversion and providing ultra-high speed digital readout. Compared to the APS, they have larger pitch size but offer other advantages such as better scaling with CMOS technology due to reduced analog circuit performance demands, and the elimination of read-related column FPN and column readout noise [8]. Although there is a very tight constraint on the area requirements, ADC architectures such as $\Delta\sum$ [38], *Multi-Channel-Bit-Serial* (MCBS) [39], SS [25], and PWM [40] are still viable. Other than ADCs, these sensors are usually packed with pixel-level memory to form a 2-dimension on-chip memory array. This memory array can be used to store the digital outputs, and also act as temporary memory buffer for other on-chip processing circuits [40].

Analog Camera-on-chip System

Camera-on-chip system is one kind of SoC combining CIS technology with ASIC and/or RF applications. These systems are commonly applied in biomedical applications, vision systems and image/video processing where there is a need to perform some algorithms. The APS and DPS architectures have been used in these systems to extract the image data with additional analog and/or digital circuits to execute the algorithms. Although there are numerous examples of analog processing algorithms, the trend is to move towards digital implementations due to significant advantages mentioned in [43]. However, a review of the more successful analog domain camera-on-chip implementations will provide great insights to the research work.

Among the analog camera-on-chip, pixel-level implementations include magnitude and gradient extraction [44], programmable pixel analog processing [45], [46] and range-position detection [47]; while sensor-level processing unit includes colour processing skin detection [21], stereo imager [48], video compression using *Discrete-Cosine-Transformation* (DCT) [49] and spatiotemporal image filtering [50]. Note that these do not represent all of the existing work in literature.

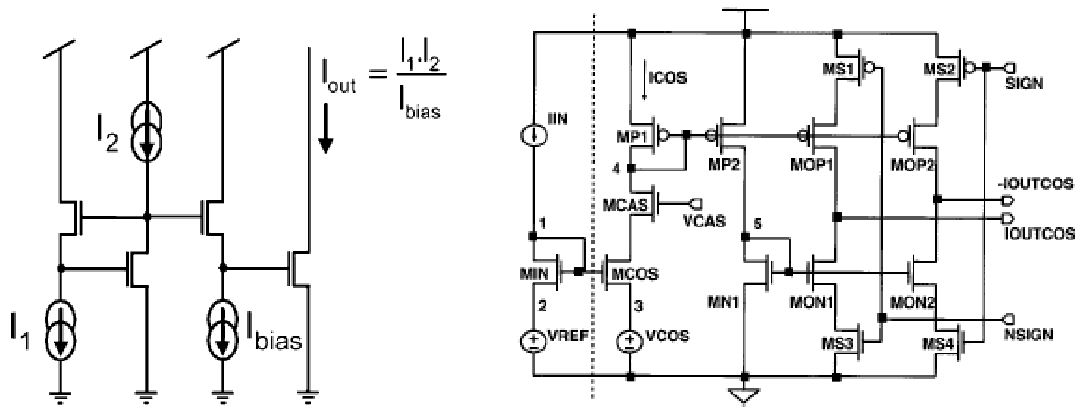


Figure 9. Sub-threshold multiplier [21], [44]

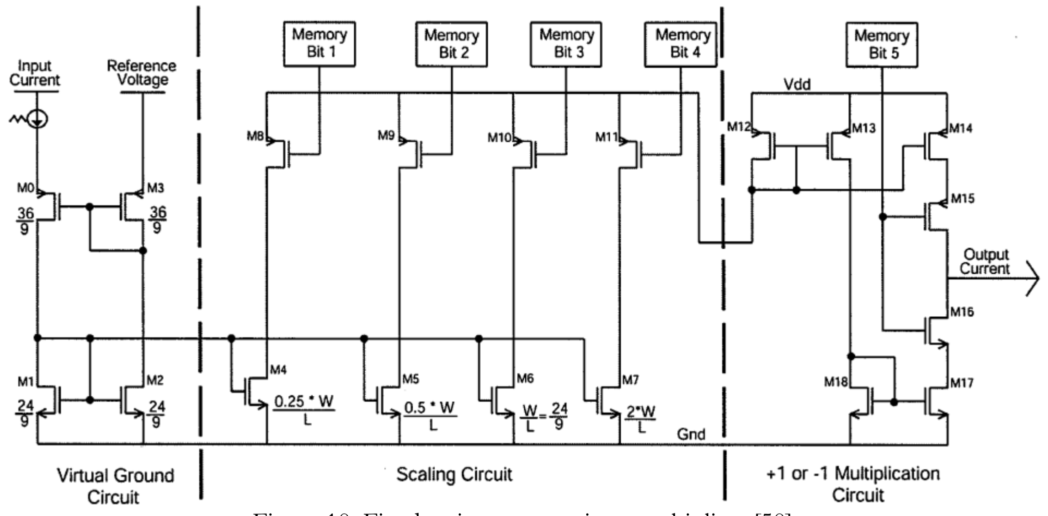


Figure 10. Fixed-ratio current mirror multipliers [50]

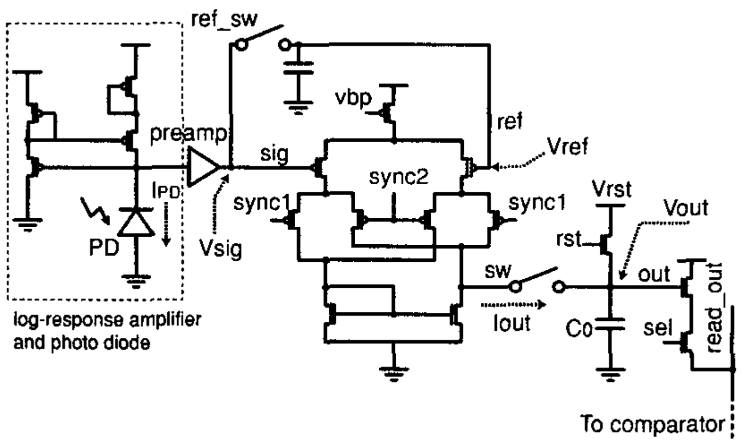


Figure 11. Gibert cell [47]

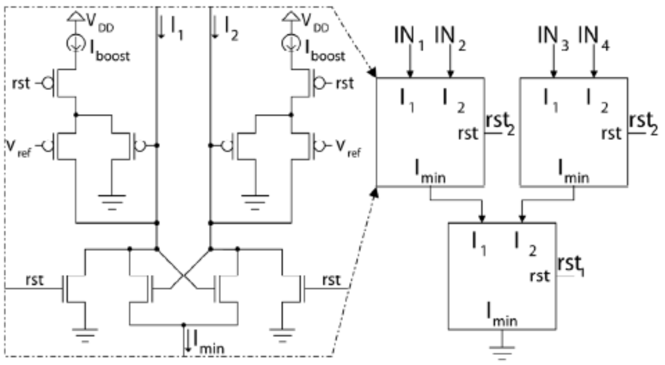


Figure 12. Loser-take-all [48]

The most popular technique in analog processing in CIS technology is to use current-mode APS architecture such as [21], [48] to implement simple arithmetic addition and subtraction, where current sums or subtracts at splitting nodes (Kirchhoff's current law). More complex operations like squaring, multiplying and dividing are implemented using sub-threshold multipliers, Figure 9, fixed-ratio current mirrors, Figure 10, and Gilbert cells, Figure 11. In addition, innovative circuits such as *Loser-Take-All* (LTA), Figure 12, and W*inner-Take-All* (WTA) algorithm also exists in [48] and [51], respectively.

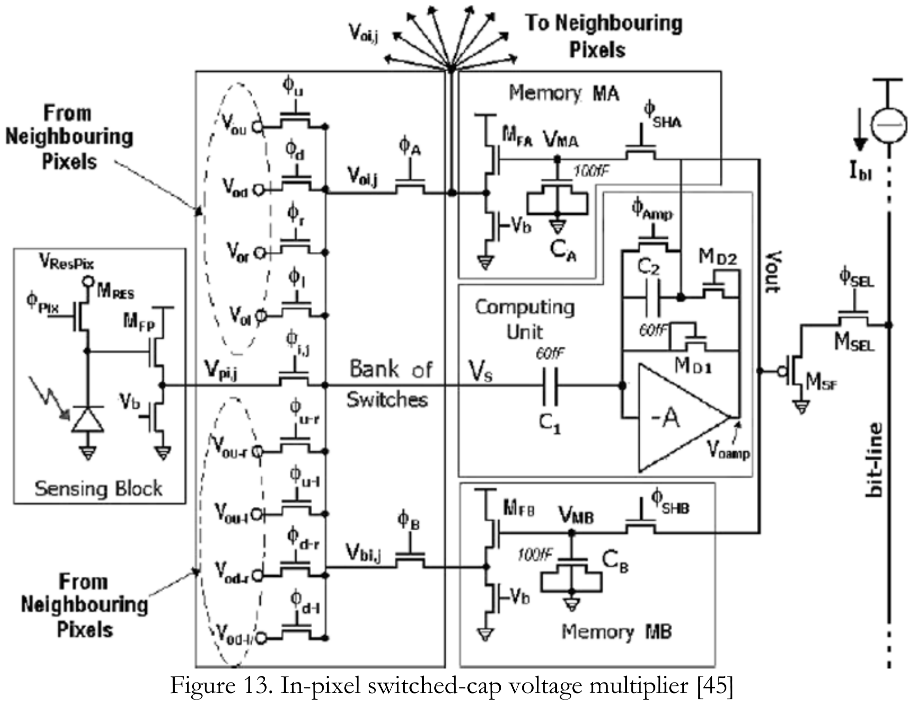


Figure 13. In-pixel switched-cap voltage multiplier [45]

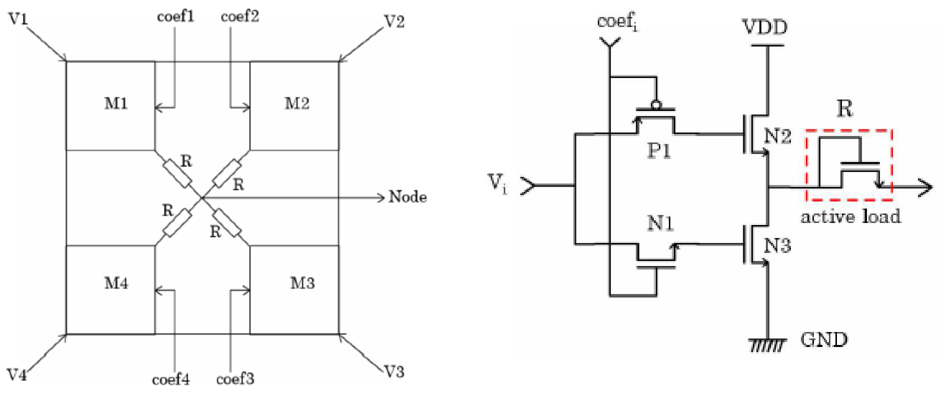


Figure 14. (a) In-pixel arithmetic unit; (b) sub-threshold multiplier [46]

Alternatively, arithmetic in voltage-mode can be realised using switched-capacitors multipliers, Figure 13, and sub-threshold multipliers, Figure 14. Although feasible, the voltage-mode multipliers are more complex to design in nature and larger than the current-mode counterparts. For example, fixed coefficient multiplication can be easily implemented as a current mirror, but exists as a differential amplifier in a switched-capacitor voltage multiplier.

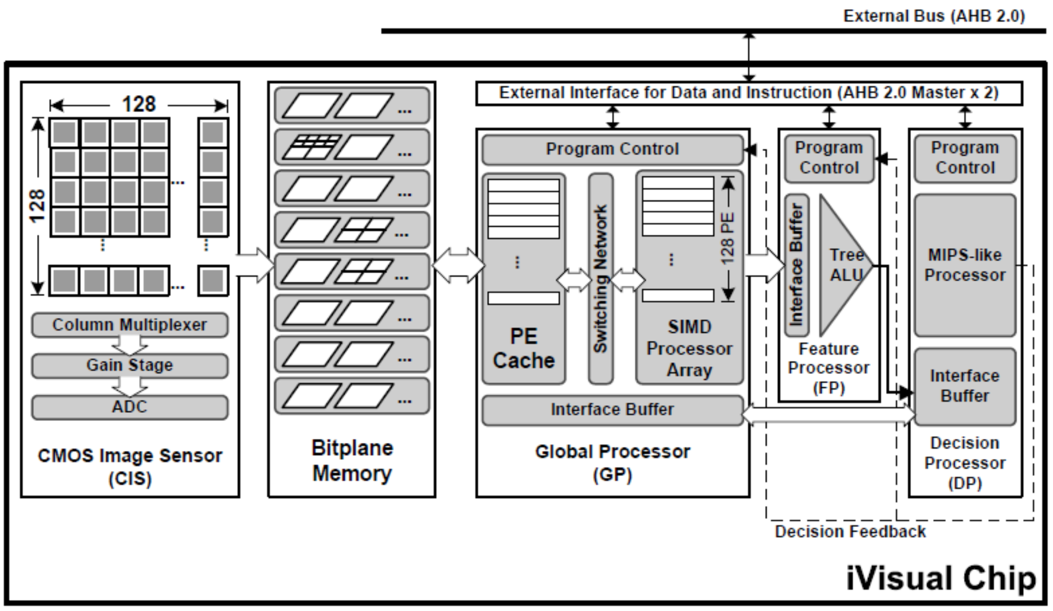


Figure 15. iVisual sensor with vision processor [27]

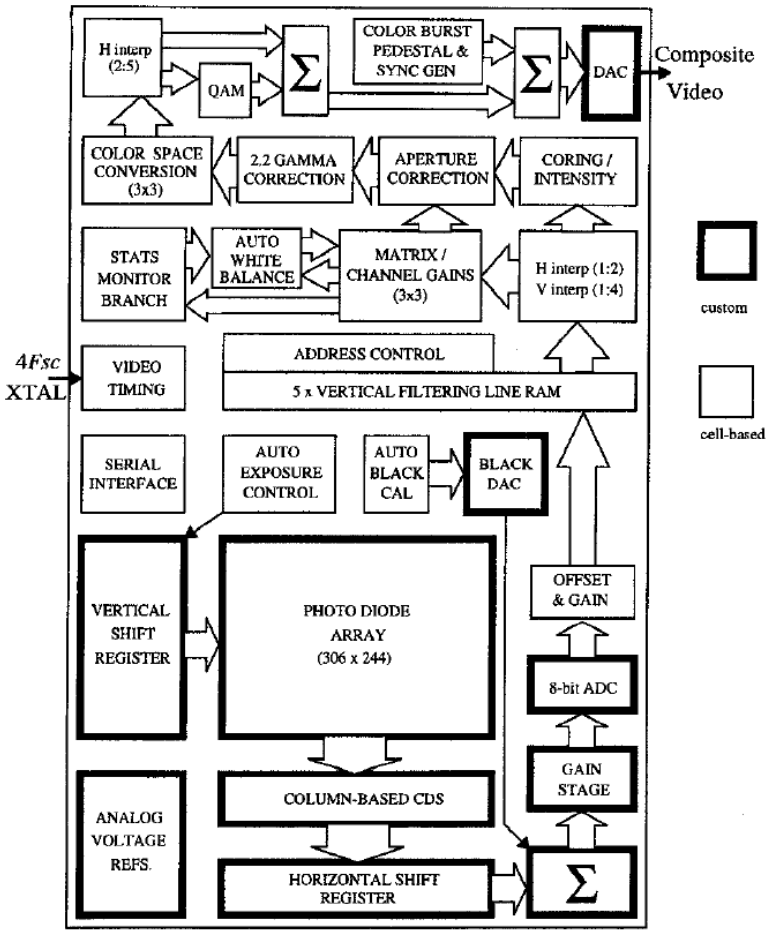


Figure 16. NTSC video camera [53]

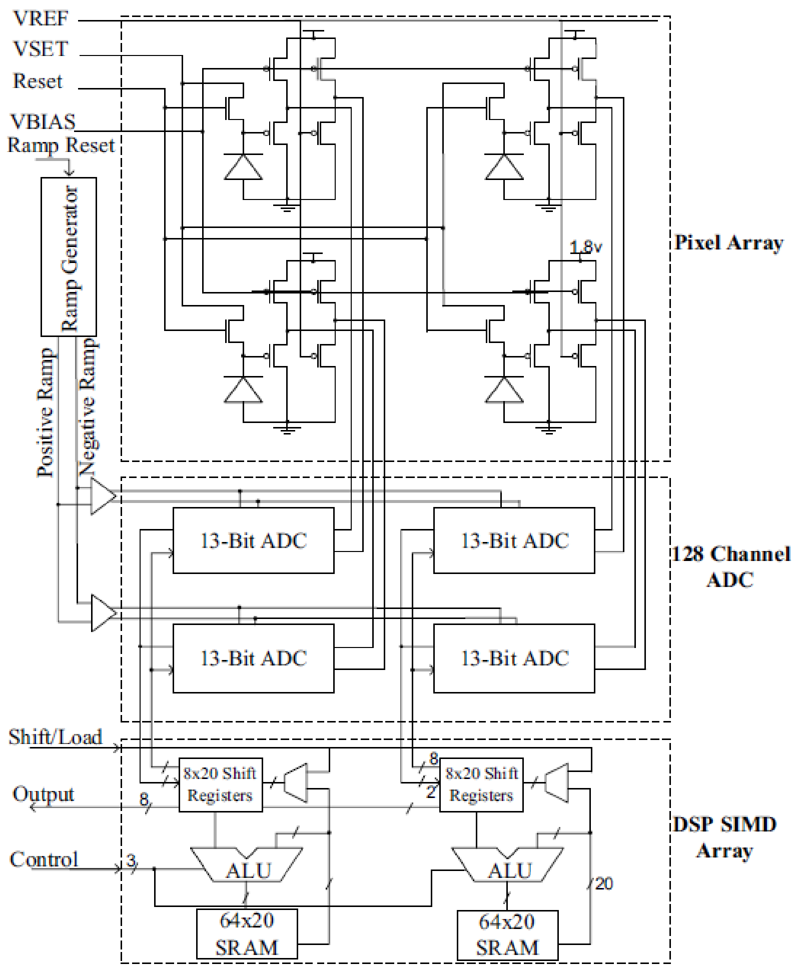


Figure 17. Bioluminescence detector [9]

More complex imaging processing algorithms are usually implemented in digital camera-on-chip systems as they are more noise tolerant and offer more precision when compared to analog processing. In addition, digital circuits offer more design reusability in nature as modern algorithms are designed on desktop applications such as Matlab. Compared to the analog-camera-on-chip systems, these systems are more integrated. For example: iVisual vision processor, Figure 15, NTSC video camera, Figure 16, and bioluminescence detector, Figure 17, have successfully embedded tons of processing elements into the image sensor.

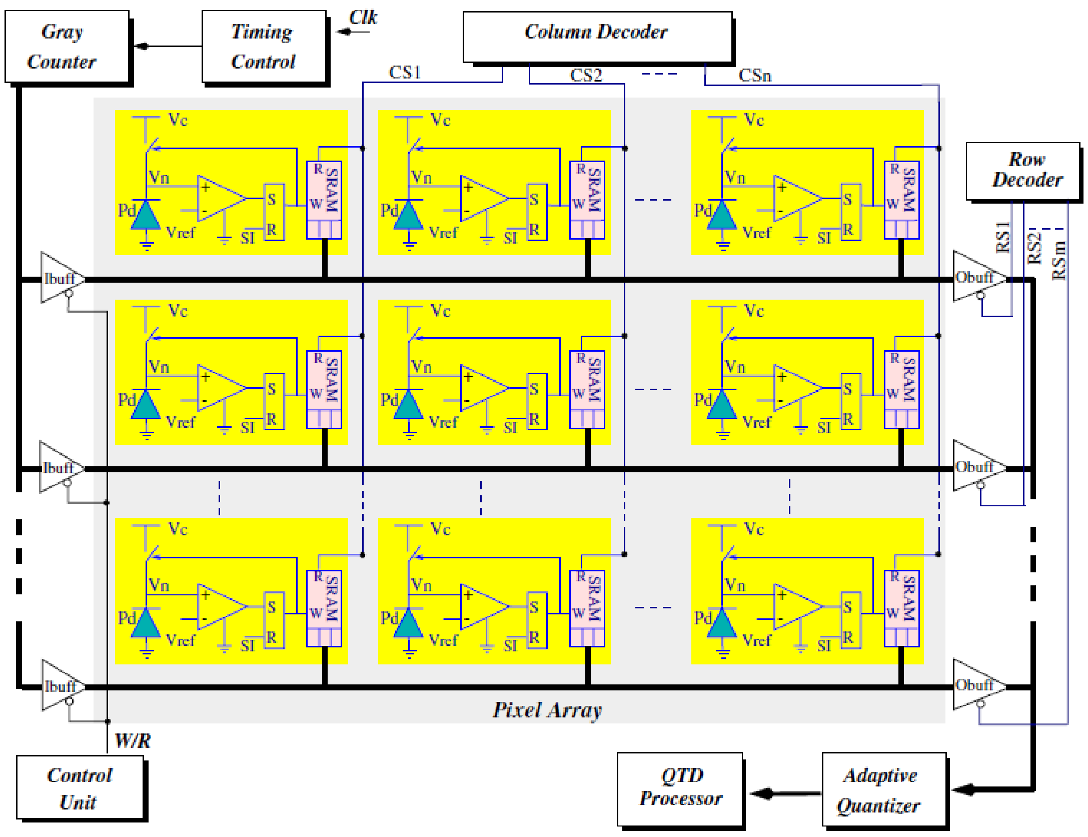


Figure 18. On-chip image compression [54]

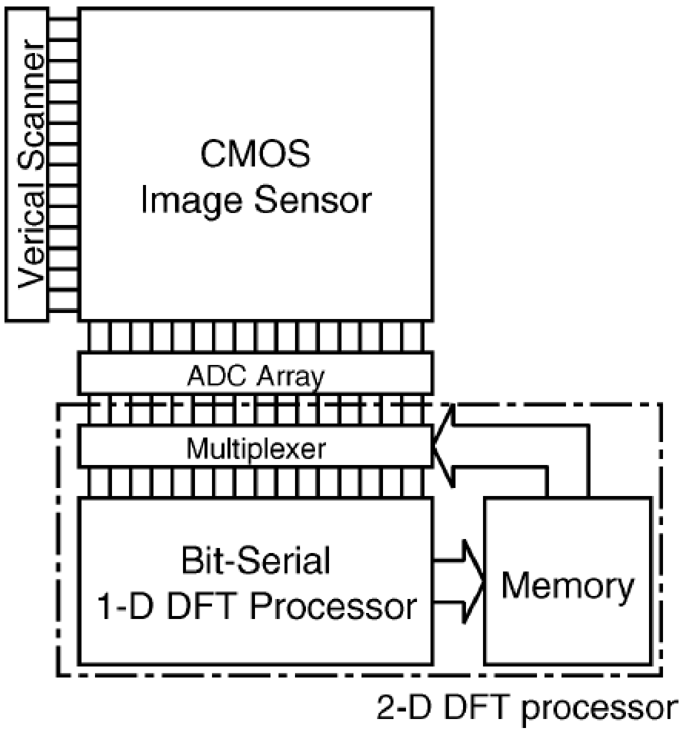


Figure 19. On-chip bit-serial DFT [55]

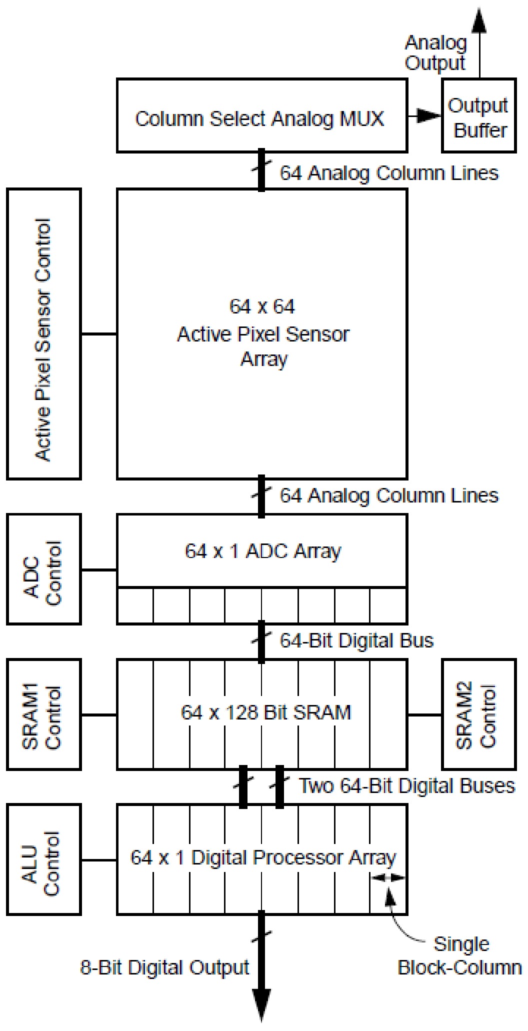


Figure 20. Column-based processor array [56]

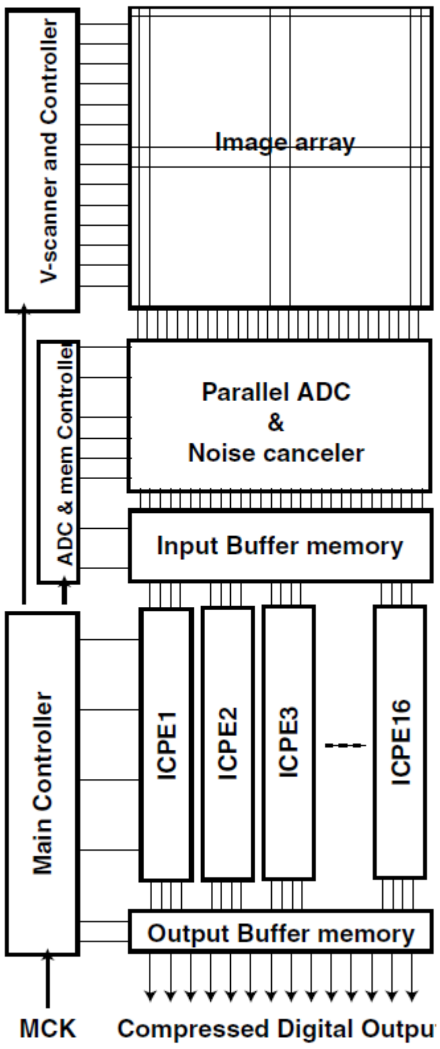


Figure 21. Parallel image compression [57]

Simpler architectures integrating single unit on-chip image compression, Figure 18, and DFT, Figure 19, are also found in literature. The former treats the image sensor as a distributed static memory array and the latter relies on digital bit-serial readout to relax on the hardware interface requirement. To achieve a higher throughput in a large resolution image sensor, the column-parallel APS architecture is exploited by embedding more column-based processing element. In Figure 20, a column-based processing array architecture using generic processor is proposed and similar architecture is also found recently in an on-chip image compression sensor, Figure 21, using dedicated discrete cosine transformation processor with a distributed arithmetic architecture from [58].

Low Power Digital Design

$$\begin{aligned} P_{total} &= P_{dynamic} + P_{static} \\ &= \left(P_{switch} + P_{short}\right) + P_{leakage} \end{aligned} \tag{9}$$

In an SoC design, the total power consumption is made up of dynamic power and static power, where dynamic power is the active power consumed when signals are changing values and static power is the power consumed when the signals are not changing [59]. In CMOS devices, the dynamic power is dominated by the switching and short circuit power while the static power is dominated by the leakage power and is defined by (9).

$$P_{dynamic} = C_{effective} \times V_{dd}^{2} \times f_{clock} \tag{10}$$

$$I_{DS} = \mu C_{ox} \times \tfrac{W}{L} \times \tfrac{(V_{GS} - V_T)^2}{2} \tag{11}$$

At 0.35μm where the threshold voltage is high, the dynamic power is much more dominant than the leakage power and is represented by the simplified switching power formula (10). Ignoring leakage power consumption at the current process, power can be effectively reduced by lowering the three components in (10). Ideally, the effective capacitance ($C_{\mathrm{effective}}$) and supply voltage (V_{dd}) should be reduced while maintaining the operating frequency (f_{clock}) for effective throughput requirement of the application. However, reducing transistors sizing ($C_{\mathrm{effective}}$) or lowering the V_{dd} reduces the drive current (11) which reduce the f_{clock} and might result in an inefficient application.

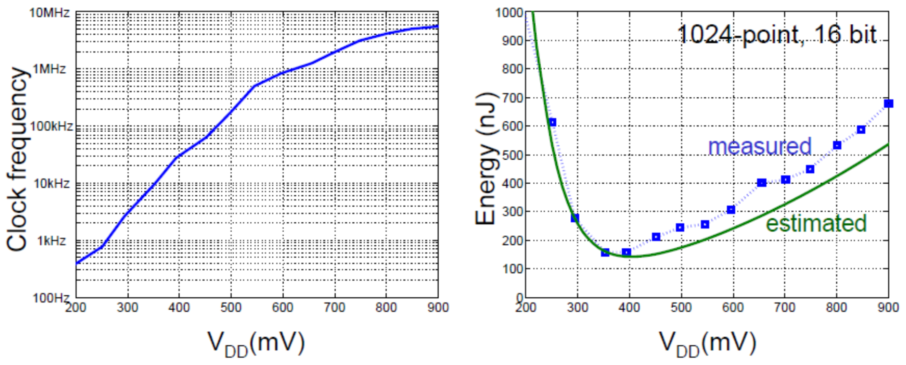


Figure 22. Energy efficient at different supply voltage [60]

In [60], a sub-threshold *Fast-Fourier-Transform* (FFT) processor has a minimum energy dissipation per 1024-FFT at V_{dd} =350mV but operates at only 9.6kHz compared to its 6MHz operation at V_{dd} =900mV, Figure 22. On many occasions where slow operations are not acceptable, the limit of low supply voltage is still placed on the application requirement and a balance is required for an efficient power driven solution. A review of relevant low power techniques associated to dynamic power consumption is provided for references in this work. However, this review do not account for all existing low power techniques in literature.

Supply Voltage

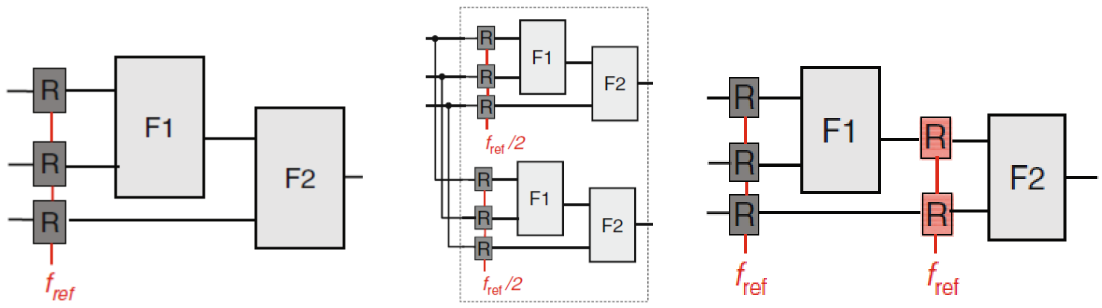


Figure 23. (a) Single-reference; (b) parallel; (c) pipelined implementation [61]

$$\begin{aligned} P_{dynamic,parallel} &= 2 \times C_{effective} \times V_{dd,reduced}^2 \times \tfrac{1}{2} \times f_{clock} \\ &= C_{effective} \times V_{dd,reduced}^2 \times f_{clock} \\ &\le P_{dynamic} \end{aligned} \tag{12}$$

The most effective method to reduce dynamic power consumption is by decreasing V_{dd} in (10) while maintaining performance of the system due to the squaring factor. To maintain the performance of the system, parallel implementation of the same design can be exploited [62]. For simplicity, consider the single reference in Figure 23 has a dynamic power (10) and duplicated implementations of the same design ($2\times C_{effective}$) with each running at half the frequency ($\frac{1}{2}\times f_{clock}$) but is able to operate at a much lower supply voltage, $V_{dd,reduced}$. The throughput performance of the system is maintained but yielding a reduction in power consumption (12) excluding any overhead power consumption. This methodology has a high power savings if massive parallel implementation is applied at the cost of larger estate. Alternatively, one can consider maintaining f_{clock} at lower V_{dd} by using a higher pipelined version [62]. While pipelined reduced the critical path delay to maintain the f_{clock}, it does increase the latency of the system but at a lower area cost. In scenarios, where the output is fed back into the input, pipelining is not applicable as the latency has increased.

Clocking Strategies

In a synchronous digital system, the clock acts as a synchronizing signal for data transfer and ALU operations. Traditional *Register-Transfer-Level* (RTL) designs assume the use of two level-triggered D-type flip-flops and configure itself as a master-slave clocking element. Designers then express their combinational logics using *Hardware Description Language* (HDL). Such flip-flops exist as standard cells in design kits of modern fabrication process and can be found in the present design kit. Although designs can be simplified with the used of HDL, clock strategies and clocking elements will then be difficult to alter if such design methods are employed.

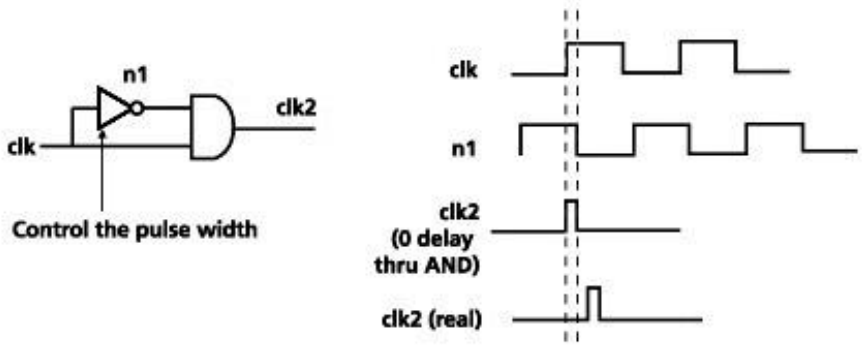


Figure 24. Pulse-latch generator [64]

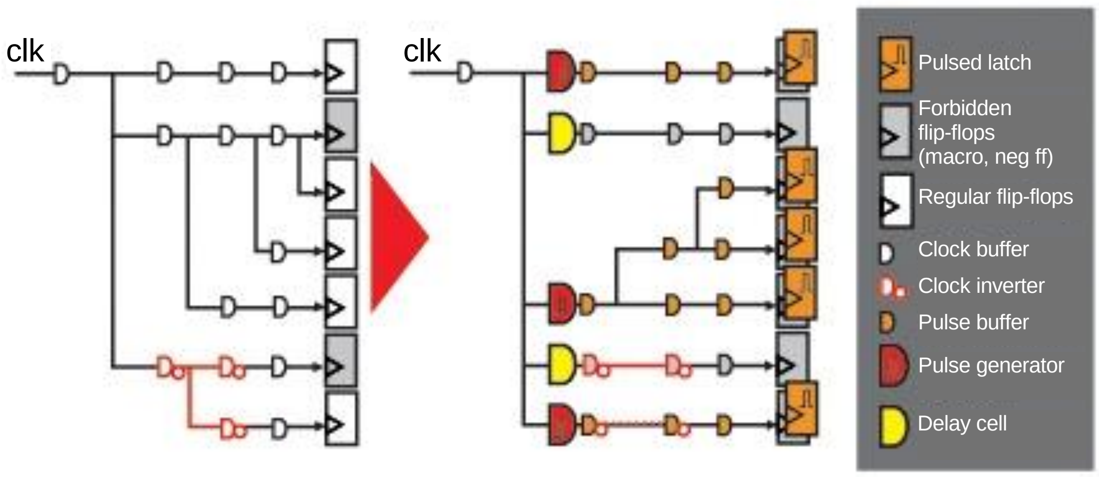
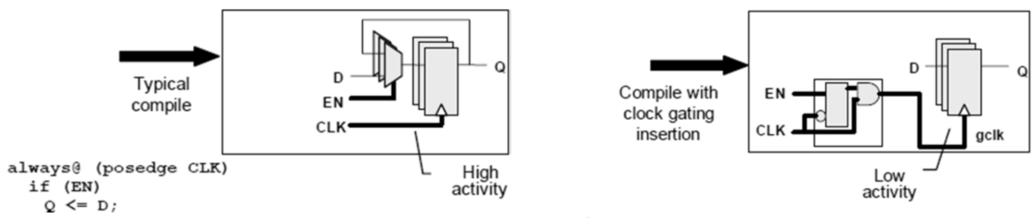


Figure 25. Pulse-latch replacement methodology [64]

Typically, clock paths are usually made up of long global interconnected lines coupled to a large number of clocking elements. As such, they often contribute to a significant fraction of the power consumption, accounting for half of dissipated dynamic power in a recent IBM study [63]. One of the modern design techniques uses a pulse-latch clock strategy, Figure 24, where real designs exhibit an approximate reduction of 20% in dynamic power [64] and these power savings come from the replacement of flip-flops with simpler and lower powered latches [63]. To enable such replacements, Figure 25, pulse generators are inserted into the clock network such that a level-triggered latch can operate similarly to an edge-triggered flip-flop. Although these pulse generators increase the overall power consumption, the incremental power can be significantly reduced by sharing a single pulse generator to more latches [65]. As such, additional pulse generators and compatible latches are required to be designed. Although the complexity increases, the combinational logic design using HDL can still be reused before the replacement of latches over flip-flops. However, additional timing analysis is required to ensure the functionality of the overall design.

In addition, there is also a hidden power reduction methodology that is applicable to a pulse-latch approach. In [66], flow-through latches have shown an improvement of 10% cycle time and 30% reduction of overall clock load. While reduction of clock load contributes directly to the overall power reduction, an improvement of cycle time also permits the lowering of voltage supply to meet the original speed requirements which in turn reduce the overall power consumption.

Clock Gating



```
always@ (posedge CLK)
  if (EN)
    Q <= D;
```

Figure 26. Clock gating replacement for memorizing registers [59]

Clock-Gating (CG) is a common method to turn off clocks when they are not required, Figure 26. This is done by inserting a gated-clock along the clock path to control the switching activity of the clock path. While a gated-clock introduce glitches, an opposite-edged-triggered latch is normally inserted to remove the glitches and is grouped together to form a clock-gating cell in standard library. (Note that clock-gating cells are not available in the standard library of the targeted design kit.) When the clocks are turned off, the state of the registers are preserved. This will disable unnecessary signal propagation, effectively reducing dynamic power. Additional power is saved from the lower switching activity of the gated-clock and additional area is saved by reducing the feedback multiplexer used in memorizing cells, Figure 26. To effectively reduce dynamic power, a single clock-gating cell can be shared by a group of registers and it is found that clock-gating of one unit is not power and area efficient [59]. An example of clock gating is demonstrated by a MPEG-4 decoder in [67].

Memory

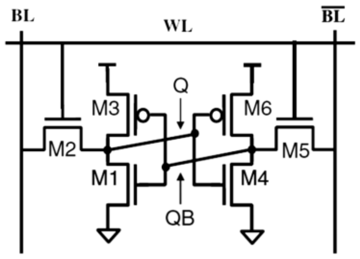


Figure 27. Traditional 6-transistor SRAM cell [61]

Almost all SoC design requires embedded memory blocks, particularly *Static-Random-Access-Memory* (SRAM), Figure 27, and accounts for a large portion of area and power [61]. Dynamic power is consumed when a read or write is performed on the SRAM cells and static power is consumed when SRAM is holding the value. During a read or write, both complementary bit-lines are charged/discharged and swings between 0 to V_{dd}. Particularly during reading, bit-lines are pre-charged to V_{dd} and are costly in terms of power consumptions as these bit-lines are densely connected by SRAMs and highly capacitive. As such, power savings can be improved by pre-charging using NMOS devices to lower the charge along the bit-lines [61].

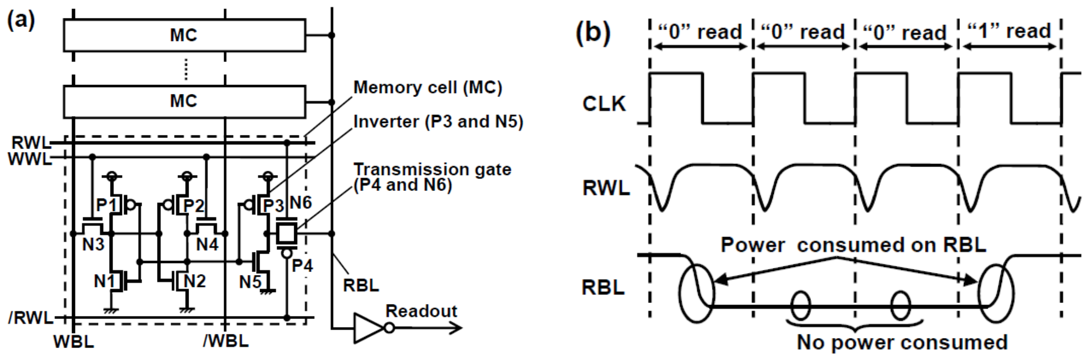


Figure 28. 10T Non pre-charge single-ended SRAM [68]

Recent studies in [68] and [69] have shown that there a high correlation of data across adjacent pixels in video/image processing. The *Most-Significant-Bits* (MSBs) are found to be lopsided to logic '0'/'1' while the *Least-Significant-Bits* (LSBs) are found to be random [68]. In such examples, there is a strong correlation in the MSB and a logic transition ('0' → '1'/ '0' → '1') occurs lesser in the MSBs as compared to LSBs. When data are read across the video/image, a 74% power reduction was found when applied on a H.264 reconstructed-image using a non pre-charge single-ended 10T SRAM [68]. These power savings can be seen from Figure 28. The downside is that this architecture does not operate as fast as a pre-charge differential-ended SRAM [69].

Types of Logic

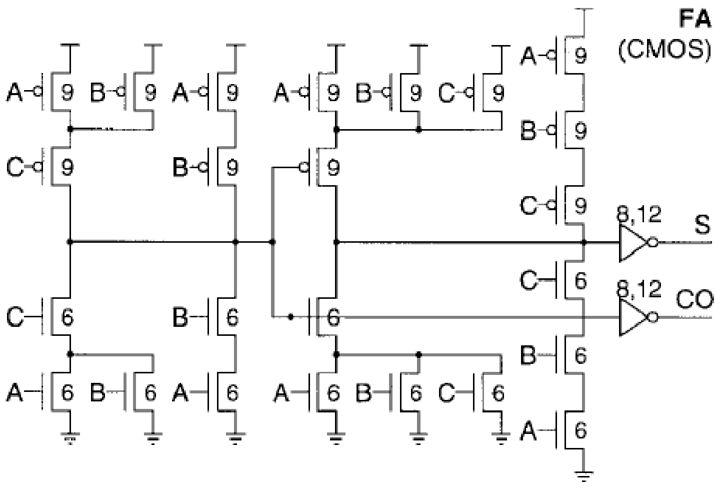


Figure 29. Static full adder [75]

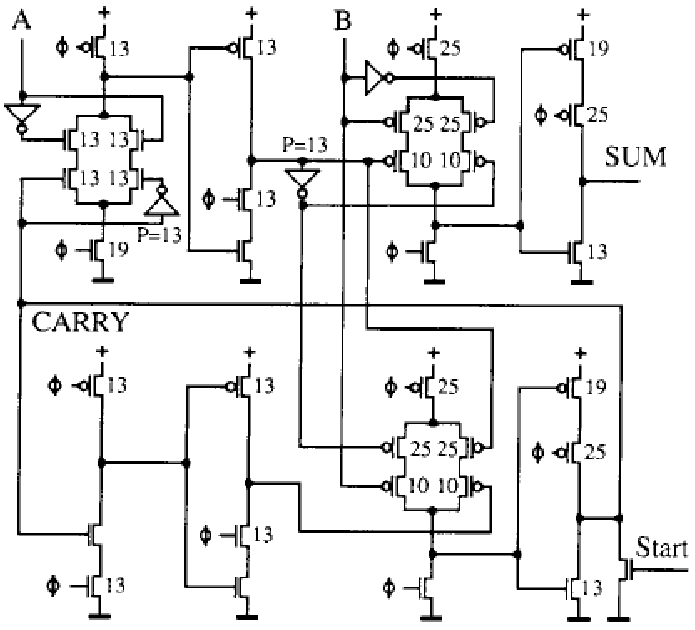


Figure 30. dynamic TSPC full adder [71]

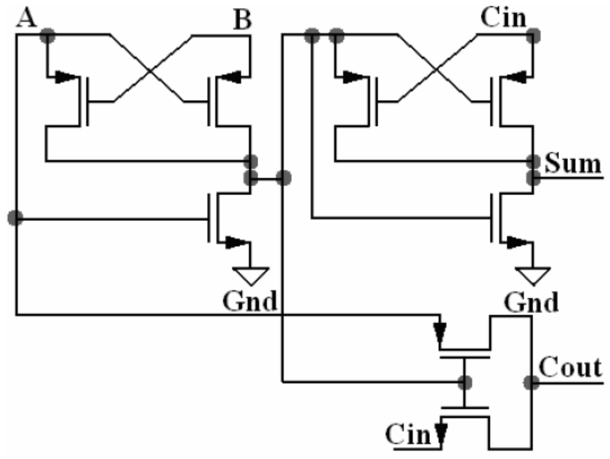


Figure 31. 8-T full adder [76]

Digital logic styles include static, dynamic and pass-transistor logic and are widely reported in literature. Static is the most commonly found and preferred style in existing literature as it is the most robust form of implementation with respect to voltage and transistor scaling [70]. Conversely, dynamic style is popular for high speed design [71] and commonly found in modern microprocessor design in the form of domino logic to relax on the critical path requirement such as [72], [73] and [74]. The final pass-transistor style makes use of single transistor logic to perform logic operation, making it the most attractive in terms of minimizing transistor count [75], driving the transistor count for a full adder cell such as Figure 29 and Figure 30 to as low as 8 transistor, Figure 31.

A comparison between static and transistor-pass logics was discussed in [62] and compared in [70]. Although the transistor-pass logic appears attractive in terms of lowering $C_{effective}$ through reducing transistor count, these circuits suffer from threshold voltage drop across single pass transistor, resulting in lower current drive and operate slower or inoperable under low supply voltage [62]. From a system perspective, the inconvenience to reduce operating supply voltage hinders power efficiency. Even in the case of *Complementary-Pass-transistor-Logic* (CPL), *Double-Pass-transistor-Logic* (DPL) where full supply voltage swing is achieved, the increase in transistor count in CPL and DPL still dissipates more power as compared to static CMOS logic style [70].

Dynamic logics evaluate a function through the use of PMOS pull-up and NMOS pull-down logic in two phases, a pre-charge and an evaluation phase. The pre-charge is done more commonly using PMOS pull-up and evaluate using only NMOS transistors. This will reduce the total input output capacitance and operates faster as compared to other logic styles. However, the constant pre-charge requirement, increase in clock load and additional self-time clocks are power hungry and unattractive for low power design [77], [78]. Dynamic logic style also suffers from charge leakage of floating nodes' and lower noise margin which limits the operating frequency and supply voltage [79]. Noise immunity circuits are often employed by using weak or inverter feedback or by inserting high skew static gates in between dynamic logics [74]. Consequently, additional overhead results in higher power consumption.

Glitch Control

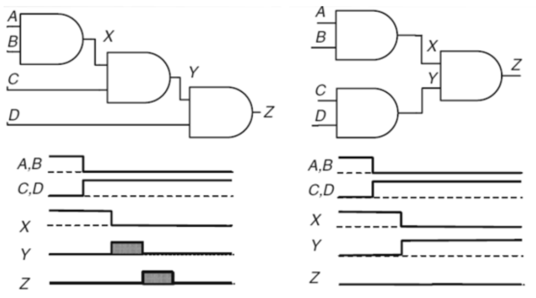


Figure 32. Path balancing [61]

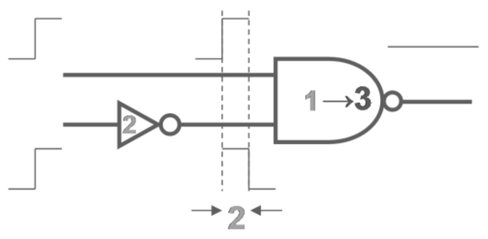


Figure 33. Hazard filtering [80]

Glitches occur when a single input change causes multiple transitions at the output and result in unnecessary dynamic power consumption [61]. The most widely mentioned method in literature is to balance the delay paths to gate Z in Figure 32, such that the inputs at gate Z switch simultaneously. Alternatively, hazard filtering can be altering the gate delay through transistor sizing, Figure 33. However, such methods have become ineffective with large variation of process parameters [81]. An alternative way to reduce glitches is to increase the number of pipeline stages as glitches are stopped at registers edge [82]. Although this method is more useful in *Field-Programmable-Gate-Array* (FPGA) as plenty of unused flip-flops are available, it does complement the use of pipelining to lower supply voltage and eventually lower power consumption.

DSP Architecture

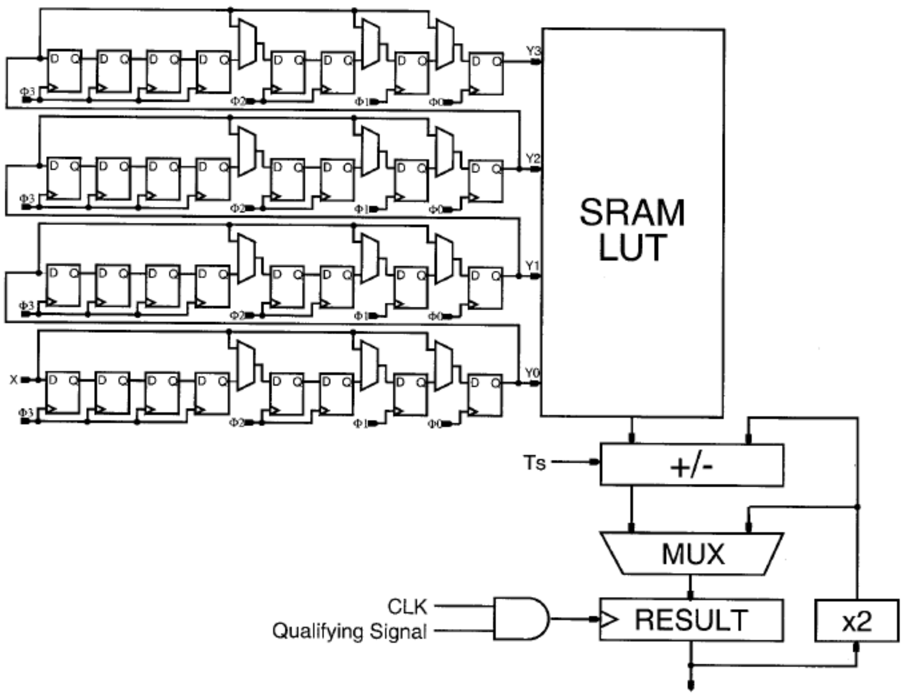


Figure 34. Distributed arithmetic architecture of μ-powered DSP [83]

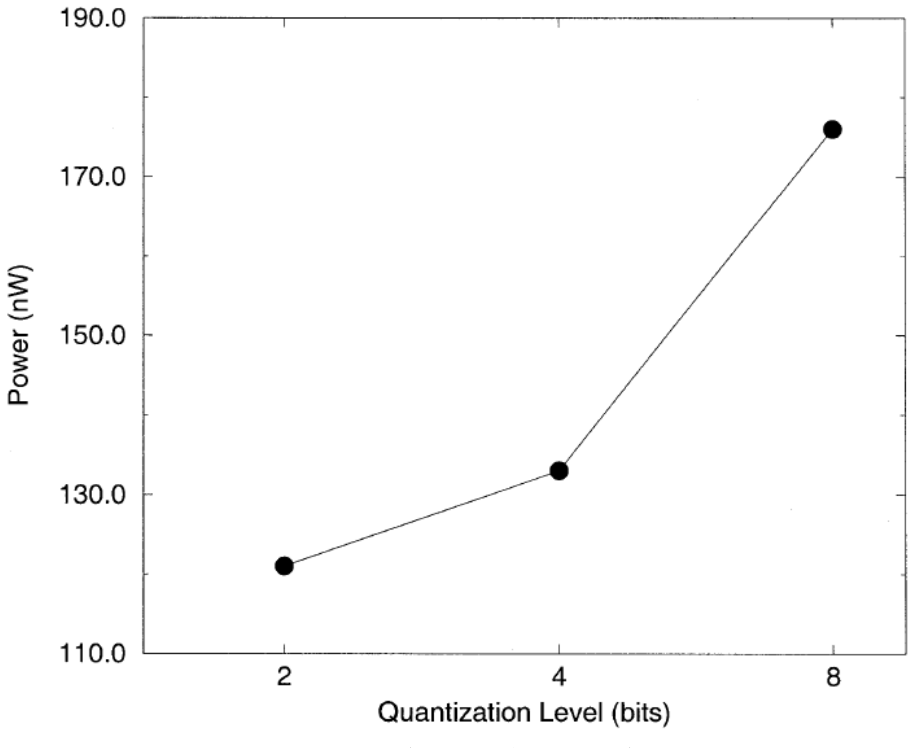


Figure 35. Measured power of μ-powered DSP [83]

While there are many low power techniques available in literature, a good DSP architecture design is more effective to produce a better efficient power system [61]. One technique used in energy harvesting architecture is the use of *Bit-Serial* (BS) and *Distributed-Arithmetic* (DA) design to reduce power consumption [84]. Such application has demonstrated a micro-powered programmable DSP, Figure 34, running at 1.2 kHz (99.8%)/250 kHz (0.2%) [83]. As DA is able to compute vector dot products without multipliers through BS arithmetic and *Look-Up-Table* (LUT), it is one of the most efficient implementation to perform *Multiply-Accumulation* (MAC) operation found in *Finite-length-Impulse-Response* (FIR) filters. In Figure 35, the work also demonstrates that if accuracy is not important, the input word-length quantization can be further relaxed to reduce power consumption.

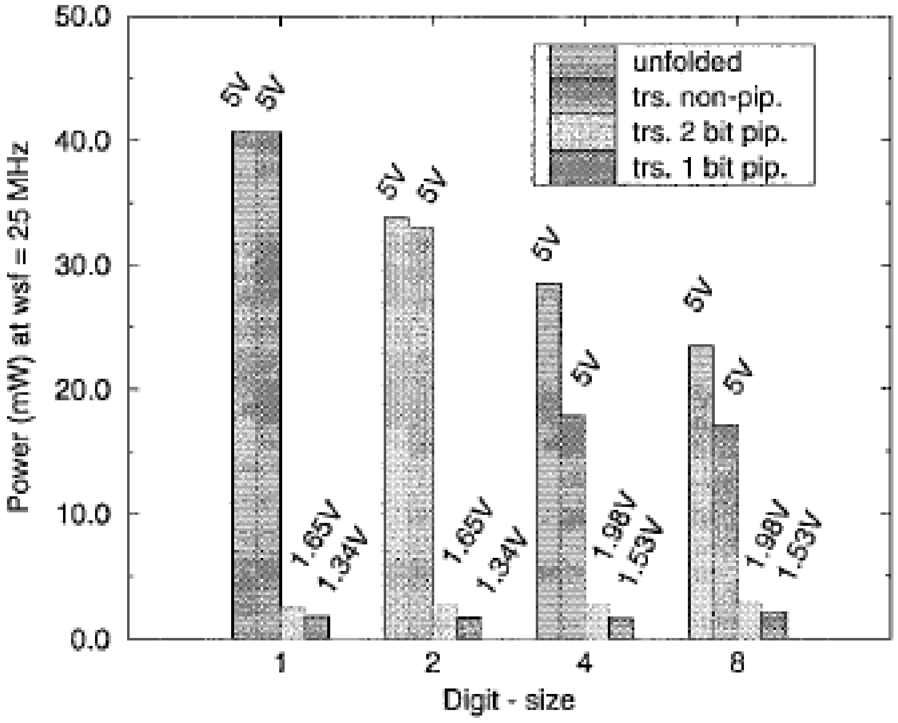


Figure 36. Comparison of 16-bit digit-serial multipliers [85]

BS and *Digit-Serial* (DS) multipliers have also demonstrated that the use of pipelining resulted in a much lower operating supply while maintaining the same word sampling frequency when compared to its unfolded multiplier, Figure 36 [85]. While significant power is saved by reducing supply voltage, a larger size digit-serial implementation is more favourable as it has a lower clock speed requirement and in turn, its voltage requirement can be lowered.

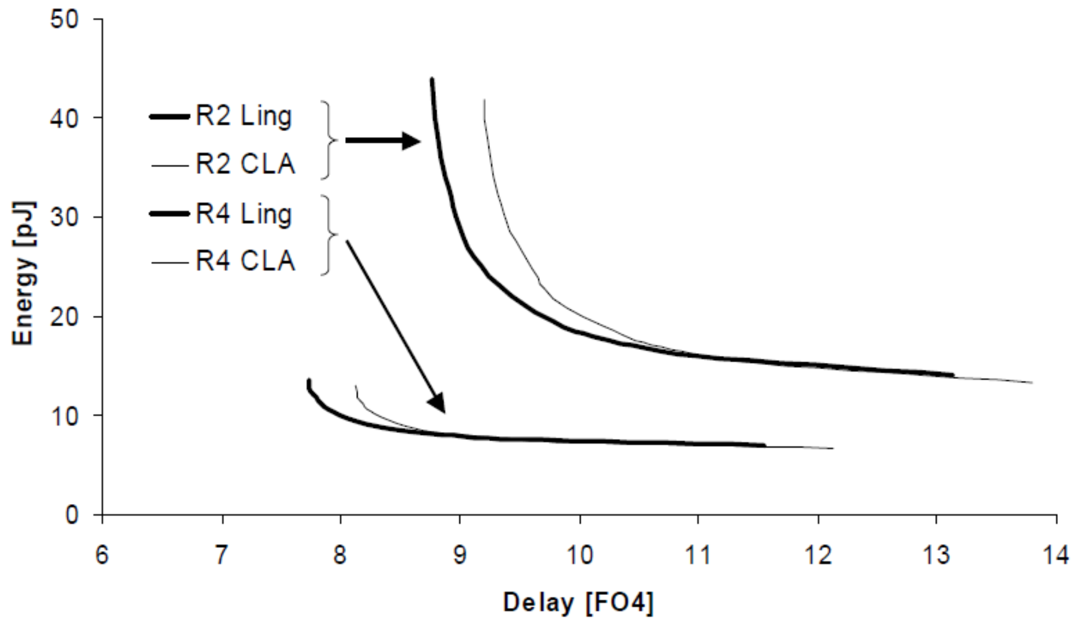


Figure 37. Ling vs CLA adder [86]

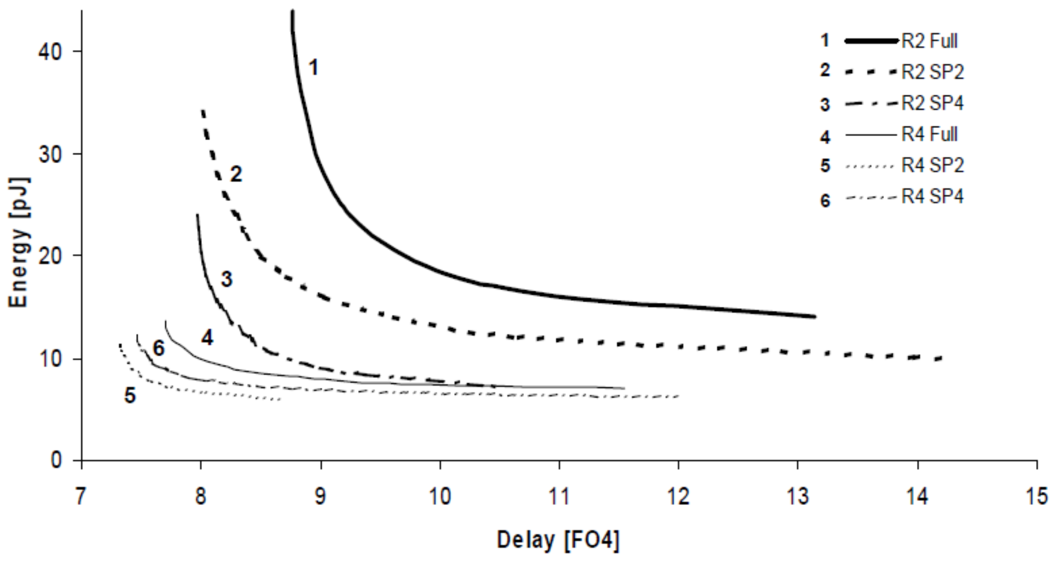


Figure 38. Sparse-tree domino ling adders [86]

The adder being implemented in all digital systems is one of the most significant cores which is subjected to optimization. Fast and energy-efficient single cycle core is often required in high performance microprocessor design and a combination implementation of radix-2 or radix-4 with full or sparse tree of conventional or Ling's *Carry-Look-Ahead* (CLA) adder is the primary choice [86]. Shown in Figure 37, the radix-4 adder proves to be more energy efficient than the radix-2 adder for the particular technology and design constraints used in [86]. In addition, the sparseness of the adders also reduces the energy consumption of the adders as shown in Figure 38.

CHAPTER III

FORMULATION OF SPECIFICATION

In this chapter, a suitable equation from the existing algorithm for hardware realization is derived. Existing CMOS sensors' architectures are then compared for suitability. Advantages and disadvantages of possible design are also discussed and the final specifications are formulated and derived for design implementation.

Algorithm

While many techniques derived from LASCA allow the estimation of blood flow based on its statistics, a suitable method for real-time imaging has to be identified for the application. Unlike normal imaging applications which require high speed, biomedical imaging specifications are often relaxed to the minimum to achieve a low-power application. Therefore, the simpler spatial derived contrast (1) is actually more suitable for implementation for real-time bio-medical application, even though the temporal statistical method has been proven to be a better technique for analysis in [1]. For instance, a temporal algorithm which requires 10 frames of image data will require more power and memory to capture, store and process more data as compared to the single frame spatial derived algorithm.

$$\begin{aligned} \sigma_I &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(I_i - \bar{I}\right)^2} \\ &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(I_i^2 - 2I_i\bar{I} + \bar{I}^2\right)} \\ &= \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} I_i^2 - 2\bar{I}\sum_{i=1}^{N} I_i + N\bar{I}^2\right)} \\ &= \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} I_i^2 - N\bar{I}^2\right)} \end{aligned} \tag{13}$$

The biggest advantage of a hardware implemented algorithm is the possibility of implementing parallel processing as compared to a sequentially process programming language. An observation on LASCA algorithm (1), page 7, is that it is parallel computation unfriendly as (3), page 7, is dependent on (2), page 7. However, (3) can be simplified to an alternative mathematical equivalence (13) , where (2) is no longer in an iterative summation loop.

$$\begin{aligned} K_{(x,y)} &= \sigma_I / \bar{I} \\ &= \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N} I_i^2 - N\bar{I}^2\right)} \Bigg/ \left(\frac{1}{N}\sum_{i=1}^{N} I_i\right) \\ &= \sqrt{\frac{N^2}{N-1}\left(\sum_{i=1}^{N} I_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} I_i\right)^2\right)} \Bigg/ \sum_{i=1}^{N} I_i \\ &= \sqrt{\frac{N}{N-1}\left(N\sum_{i=1}^{N} I_i^2 - \left(\sum_{i=1}^{N} I_i\right)^2\right)} \Bigg/ \sum_{i=1}^{N} I_i \end{aligned} \tag{14}$$

A further substitution of (2) and (13) into (1) simplifies to (14) which only differs to (5) by a constant factor of √(*N*/*N*-1). However, this constant factor approaches to 1 when the sample size is large and $K_{(x,\ y)}$ approximates to (15) which is actually mathematically equivalent to (5). From a statistical point of view, the four different methods in [1] are mathematically describing the same coefficient of variation and only differ in terms of the input samples and sample size. Typically, a larger sample size using statistical methods will lead to an increased precision in estimating the evaluated properties. Using the observation results from [1], a size of 5×5 window of 8-bit data processed with LASCA should be sufficient for monitoring purpose and this will heavily relax on the timing and memory requirements.

$$K_{(x,y)} \approx \sqrt{N\sum_{i=1}^{N} I_i^2 - \left(\sum_{i=1}^{N} I_i\right)^2} \Bigg/ \sum_{i=1}^{N} I_i \tag{15}$$

Symbol	*Bit width*
I	8
I^2	16
ΣI	13
ΣI^2	21
$N\Sigma I^2$	26
$(\Sigma I)^2$	26
$N\Sigma I^2-(\Sigma I)^2$	26
$\sqrt{(N\Sigma I^2-(\Sigma I)^2)}$	13

Table 1. Bit width requirement

The reason of using (15) is its simplicity, defined only with N, ΣI and ΣI^2. In addition, the equation is parallel computation friendly as ΣI and ΣI^2 are independent of one another. Further simplification by squaring both the numerator and denominator will only incur a large bit width divider and is undesirable. Using inputs of 5×5×8-bit (N=25), the bit width for the various symbols in (15) has been tabulated in Table 1. Subsequently, the required bit width for each arithmetic operator can be identified.

CMOS Image Sensor

Although the current work does not involve the design of the sensor, the choice of sensor architecture does affect the design of the processing unit. The choice of the sensor architecture should ideally be a low-power and power-efficient solution while maintaining a decent throughput.

Description	*Process*	*Supply*	*Resolution*	*Pitch*	*Fps*	*Power*	*FOM*	*Operator*	*Output*
Skin detection [21]	0.18μm	1.8v	-	9T	-	-	-	×,÷,+,-	1 bit
Neuromorphic vision [48]	0.35μm	3.3v	2×128×128	10μm	40	33.6mW	25.6nW/fp	+,-,LTA	7 bit
→	-	-	-	-	30	33.2mW	30.7nW/fp	-	-
Gradient extraction [44]	0.5μm	-	100×100	80μm	1k	50mW	5.0nW/fp	×,÷,+,-	3.5 bit
Spatiotemporal image filters [50]	1.2μm	5v	16×16	30μm	20k	1mW	195pW/fp	×,÷,+,-	5 bit
Range position [47]	0.5μm	3.3v	64×64	40μm	2k	400mW	400nW/fp	×,+,-	1 bit
Pixel analog processor [45]	0.35μm	3.3v	32×32	35μm	30	6mW	195nW/fp	×,+,-	-
Video compression (sensor) [49]	0.35μm	3v	128×128	16μm	1k	7.2mW	439pW/fp	-	-
Video compression (DCT) [49]	-	-	2×8×8	-	~24k	-	7.1nW/fp	×,+	8 bit
Color correction sensor [52]	0.35μm	3.3v	352×288	4T	30	20mW	6.6nW/fp	×,÷,+,-	8 bit
Pixel analog processor [46]	0.35μm	3.3v	64×64	35μm	10k	250mW	6.1nW/fp	×,+	4.5 bit

Table 2. Performance of analog domain camera-on-chip

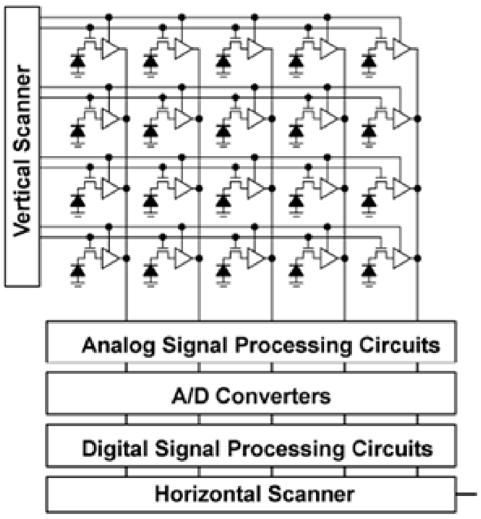
In Table 2, a list of analog domain camera-on-chip is tabulated to discuss on the disadvantages of using analog processing for the current work. The major limitation of analog processing is the incredible low accuracy of 1-bit to 8-bits of processed data where the accuracy requirement of this work is 26-bits, Table 1. Even in the application of more precise computation [48], [49] and [52], only simple addition/subtraction and fixed-multiplication using switched-capped voltage multiplier is feasible. More importantly, the accuracy of squaring operation in (15) will be difficult to achieve in analog implementation. Even if achievable, there is still a lot of downside. The fundamental of analog processing lies in matching and to achieve matching between transistors, large transistors are required. This will result in even larger pixels when compared to [44], [47] and [46] if in-pixel analog processing is required. Thus a digital approach is more suitable for the current work.

Description	*Process*	*Supply*	*Resolution*	*Pitch*	*Fps*	*Power*	*FOM*	*Output*
APS-SAR [41]	0.35μm	1.2v	176×144	5.0μm	20	48μW(no pad)	95pW/fp	8 bits
APS-SAR [42]	0.35μm	1.5v	176×144	5.0μm	30	550μW	723pW/fp	8 bits
APS-COL-SS [33]	0.25μm	2.5v	400×330	7.4μm	142	52mW	2.77nW/fp	10 bits
APS-COL-SAR [34]	0.35μm	-	512×512	16μm	2.5k	500mW	763pW/fp	9 bits
APS-COL-CYCLIC [35]	0.25μm	2.5/3.3v	514×530	20μm	3.5k	1W	1nW/fp	12 bits
APS-COL-Δ∑ [36]	0.5μm	5v	1k×1k	10μm	1	115mW	115nW/fp	12 bits
APS-COL-PWM [37]	0.35μm	1.35v	128×96	10μm	9.6	55.2μW	468pW/fp	9 bits
DPS-Δ∑ [38]	1.2μm	5v	64×64	60μm	30	1mW	8.1nW/fp	8 bits
DPS-MCBS [39]	0.35μm	3.3v	320×256	10.5μm	30	20mW	8.1nW/fp	8 bits
DPS-SS [25]	0.18μm	1.8v	352×288	9.4μm	10k	50mW	49pW/fp	8 bits
DPS-PWM [40]	0.18μm	1.8v	5128	~25.1μm	2	1.4mW	146nW/fp	-

Table 3. Performance of APS and DPS sensors

$$FOM = \frac{Power}{fps \times pixels} \tag{16}$$

While many CMOS sensor design claims low power operation in few milli-watts, many have failed to address the power efficiency of the sensor. In Table 3, a comparison of existing APS and DPS architectures that include sufficient information for calculating *Figure-Of-Merit* (*FOM*) is given to evaluate on suitable low-power and power-efficient sensor design. *FOM* is defined in [48] by normalizing the power dissipation by the frame rate and number of pixels (16) with units of milli-watt per frame-pixel (mW/fp). This implicitly measures the amount of energy required to extract information from a single pixel. Note that the characteristics of the photodiode are excluded. The most power-efficient sensor has a DPS architecture using SS ADC [25], while the lowest power-consumption sensor is a APS sensor with column-parallel PWM ADC [37] and ranks second in terms of power-efficient. The use of SAR ADC also came close in both low-power and power-efficient sensor.



Vertical Scanner
Analog Signal Processing Circuits
A/D Converters
Digital Signal Processing Circuits
Horizontal Scanner

Figure 39. CMOS sensor with column parallel analog and digital circuits [32]

Particularly, the use of column-parallel DSP units with the sensor in [56] coincides with exploiting parallelism for low power applications [62]. A recap of such architecture is shown in Figure 39. This architecture also corresponds with a low-power and power-efficient image sensor found in [37]. In order to maximise the number of column-parallel DSP units, the area of the DSP unit is critically important. This in turn minimises the operating supply of the digital circuits. Therefore, employing such architecture gives more design room space for the low-power digital circuits.

DSP Architecture

The computational complexity of the algorithm (15) lies in implementing square-root and division operation. Unlike MAC operations, square-root and division are usually implemented using cycle-operated algorithms such as restoring, non-restoring, SRT and Newton-Raphson [87]. Particular of interest is a non-restoring square-root unit (using only one adder/subtractor and shift registers) in [88], coupled with a conventional subtractive algorithmic division (one subtractor and shift-load registers) fulfils the advantage of small area implementation. The smallest radix-2 unit generates a digit per cycle from the MSB and can be chained together in the same pipeline stage.

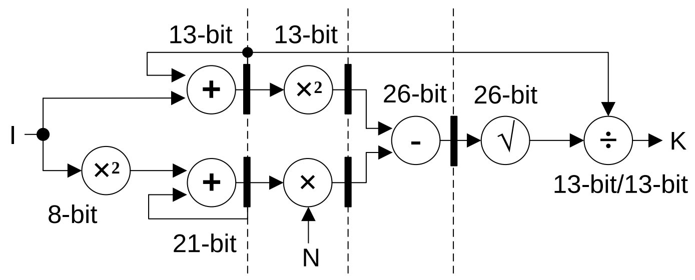


Figure 40. Bit-parallel iterative with maximum pipelining

If the cycle-operated divider and square-rooter are iterated in a bit-parallel fashion, Figure 40, each pipeline stages will operate within a number of clock cycles. In order to calculate ΣI and ΣI^2 in (15), the first pipeline stage is limited to one-cycle arithmetic operations to align its input to the second stage. Subsequent stages can be implemented using cycle-operated or multi-cycled units. Although this architecture can be easily implementable, large area and long critical path delay is associated with the bit-parallel arithmetic units. Even if the squarer, fixed-multiplier and subtractor in the second and third stage are implemented as cycle-operated or multi-cycled units, the overall path delay is still restricted to the 21-bit adder in the first stage.

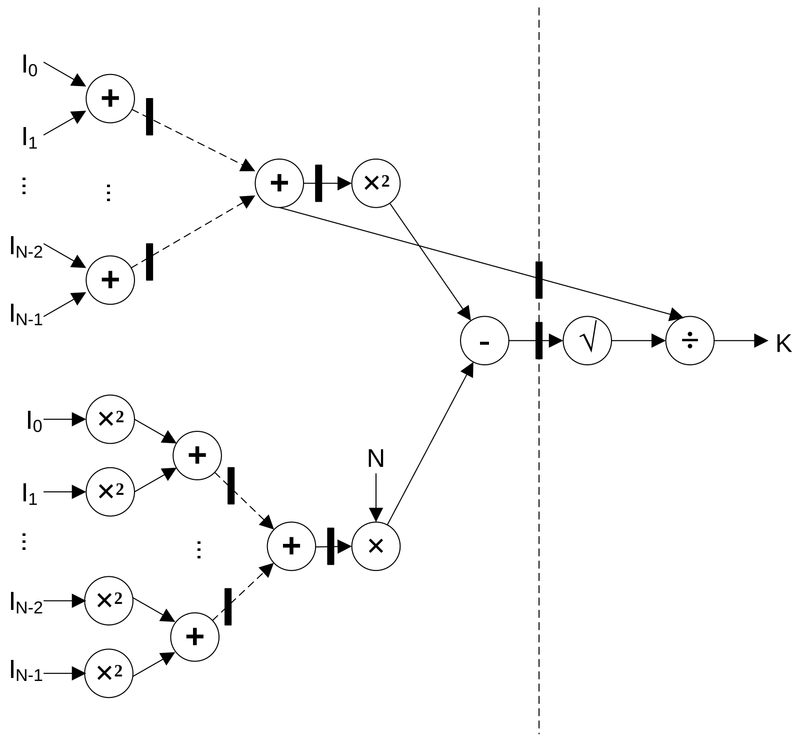


Figure 41. Bit-serial architecture

Since the cycle-operated division and square-root unit stays in the architecture, the use of BS arithmetic is an alternative to the bit-parallel cycle-operated arithmetic units. The architecture can be divided into two pipeline stages, Figure 41, where the first stage consists of the adding, squaring, multiplying and subtracting, and the second stage is made up of the square-root and divisor unit. In the first stage, a continuous chain of LSB-*First* (LSBF) BS processing unit can be realised as described in [89] and this eliminates any path delay in the first stage. In addition, a single BS unit occupies much lesser area as compared to its bit-parallel implementation at the expense of more cycle operations. For example: a 26-bit adder/subtractor can be realised with a 1-bit adder and registers. However, there is a blown up of 25 BS-squarer required at the input compare to a single 8-bit bit-parallel squarer. It is unlikely that this design will realise a more compact design than the prior version.

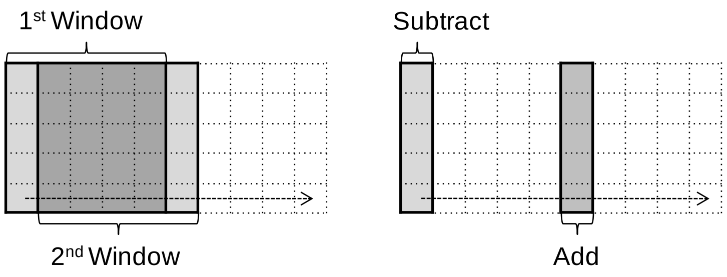


Figure 42. 5×5 window selection of pixels and difference in window

Consider a window scanning sequence from left to right defined by the dotted arrow in Figure 42, there is an overlap of data between the first window and second window for processing. For that reason, re-calculating ΣI and ΣI^2 in (15) from scratch is redundant and all that is required is to add and subtract the difference from the previous window. If the intermediate results of ΣI and ΣI^2 can be stored within the DSP, the amount of memory reads required per speckle contrast will be reduced from 25 pixels to 10 pixels. Consequently, the amount of addition and multiplication is also reduced to 10. Not only does this methodology reduce the power consumption of memory reads by 60%, it also relaxed the number of BS-squarer from 25 to 10.

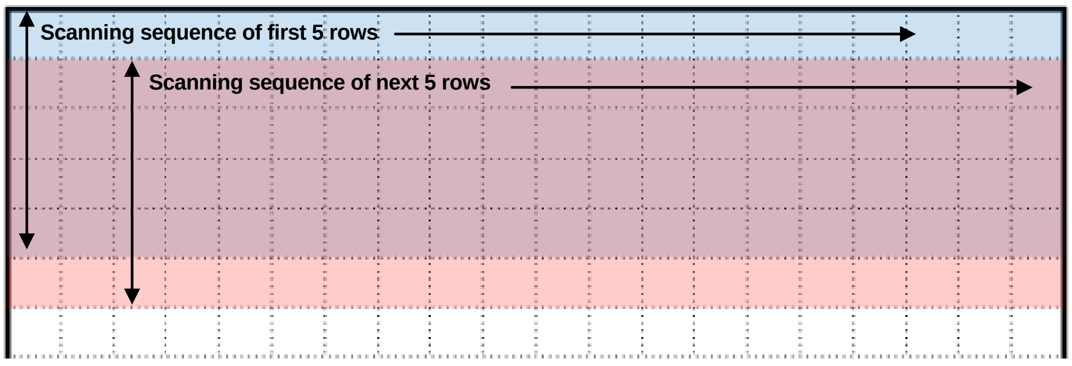

Scanning sequence of first 5 rows
Scanning sequence of next 5 rows

Figure 43, Scanning sequences of different rows

The scanning sequences can then be repeated for different rows as shown in Figure 43. One might think that storing ΣI and ΣI^2 per column might further reduce pixel memory access to compute the total ΣI and ΣI^2 per window, but in actual fact it is not. This is because scanning sequences of different rows have overlapped region and individual pixel's data are required to be known to differentiate the scanning sequences. As such additional memories and memory transfers are required to store ΣI and ΣI^2 from a block of pixel's data to a new block of memory. This could amount to a large number of memory blocks if the numbers of columns are large and is undesirable.

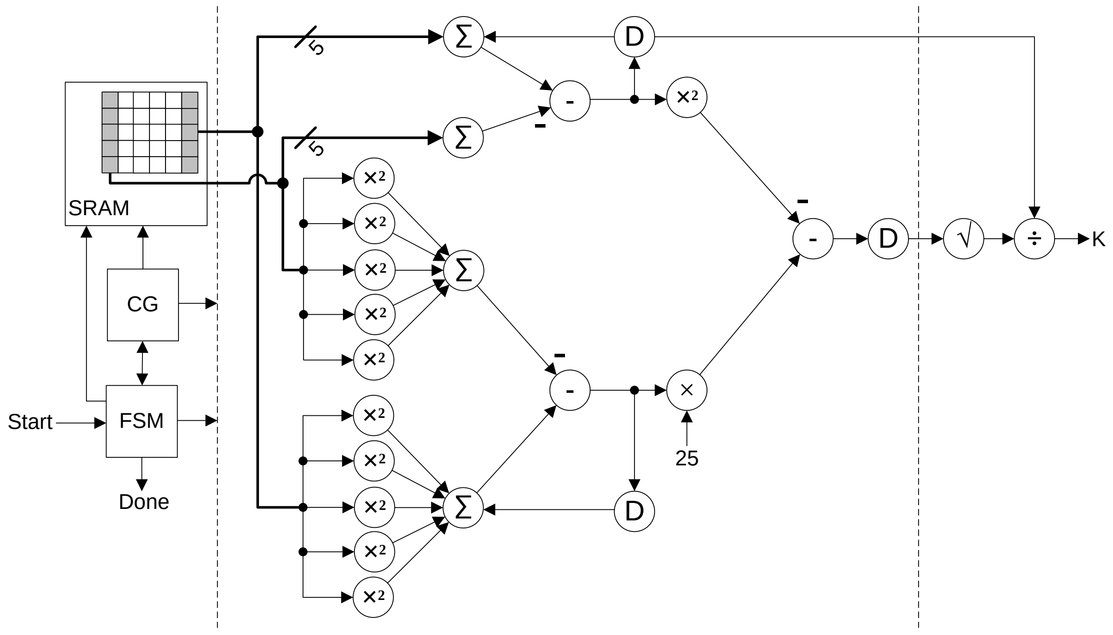


Figure 44. Reduced bit-serial architecture (D - delay elements)

Description	*Gate*	*Frequency*
Bit-parallel, Figure 40	9.7k	118MHz
Bit-serial, Figure 44	9.4k	211MHz

Table 4. Estimate gate count and operating frequency of arithmetic nodes at 3.3v

A reduced BS architecture with its *Finite-State-Machine* (FSM) controller, *Clock-Gating* (CG) generator and a required window scanning SRAM is shown in Figure 44. To differentiate both the design, estimates of gate-count using un-optimised arithmetic units were performed on both implementation and are shown in Table 4. The bit-parallel implementations are estimated with slower but area-efficient ripple-carry adders while the rest are compiled using the standard library. The square-root and division units remain same in both designs. In both cases, the gate counts are approximately the same, with the BS estimates having more non-combinatorial logic. The ripple-carry adders are found to limit the operating frequency in the bit-parallel estimates, while the latter is limited by the square-root unit. Note that optimizing the bit-parallel adders is only possible at the expense of occupying more area. The main purpose of this estimate is to ensure that 10 BS-squarer do not occupy too much area so that power reduction can be performed by reducing the supply voltage aggressively.

Specification

The number of cycles required to operate will now be determined by the depth of BS-chain and the output bit width. A depth of 4 and 26-bit required by ΣI^2 in (15) will limit the operating cycles to be 30 per pipeline stage. Since the output precision is determined by the divisor at the last stage, a highly precise output of 13 integer bits and 15 fractional bits (Q13.15) will be produced in 30 cycles (one digit per cycle for radix-2 divisor). The last two cycles is used on loading/resetting and propagating the output from the square-root unit to the divisor. Although the DSP unit requires multiple cycles to generate a single output, this can be compensated by duplicating them in the column-parallel architecture, Figure 39.

Due to complexity of the algorithm, a similar reference of column-parallel image compression sensor in [57] is used to determine the pixel pitch of the sensor. In [57], the pixel pitch is 15μm in a 0.25μm technology and the pixel pitch is relaxed to 20μm for the current 0.35μm technology. Although this pixel pitch is slightly bigger than previously mentioned architecture in Table 3, most of the column-parallel implementations are achieved in top-bottom architecture, Figure 7 on page 12, which is not feasible in this case.

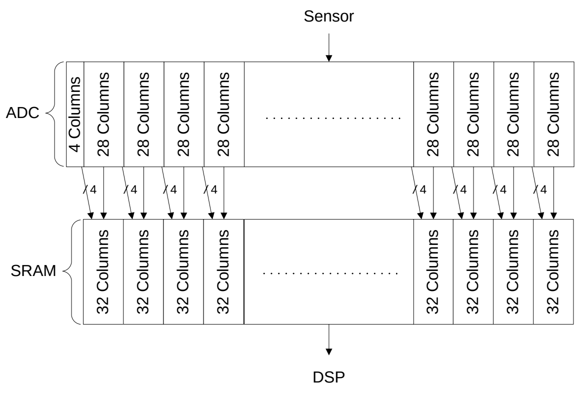


Figure 45. Packed SRAM arrangement

Unlike block-based image processing algorithm such as DCT where neighbouring data is not required between blocks, a window-based image processing algorithm requires all neighbouring data to be present for every pixel. In [56], a global metal bus is used to pass information between neighbours as it operates as a generic processor. A simpler method is to pack more columns of SRAM within each separable block and duplicate the data at edge of the blocks. For example, using a window size of 5×5 will require a packed SRAM arrangement of 32 columns to every 28 columns as shown in Figure 45. Since the data is digital, splitting the node at the edge should not be problematic if a BS-interface such as [55] is employed.

$$Column_{image} = 4 + n \times 28 \tag{17}$$

Choosing a convenient 32-to-28 column limits the pitch width of the DSP unit to 28×20μm=560μm. Note that the pixel width can be adjusted for future implementation by packing more SRAM into the DSP block. The column of the image can only take values of (17), where n is an integer, while the row is not limited. For example, a large resolution image, *n* can take 32 and thus $Column_{image}$=4+32×28=900 while number of rows can be 1024.

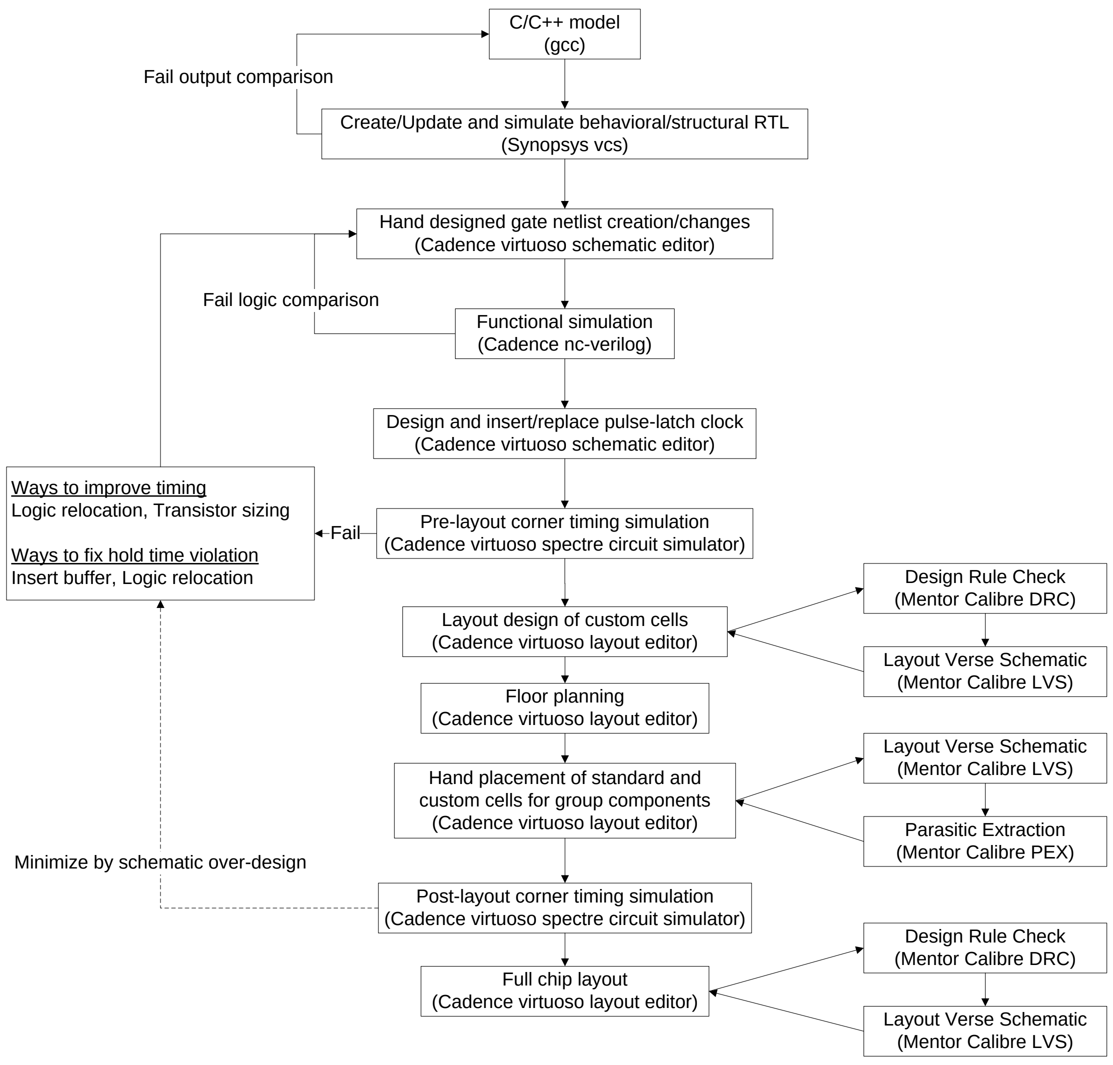
$$f_{\min} \approx pixels \times fps \times 30(cycle/speckle)/units \tag{18}$$

The operating frequency of the DSP unit can be estimated with (18) and can be estimated to be 26MHz using resolution of 1024×900 at a frame-rate of 30fps with 32 DSP units. It is rounded up to 30MHz to provide an extra overhead for data transfer between ADC and the SRAM. The unit can be operated slower at 20fps or lesser to reduce power consumption since video rate is not required in biomedical imaging. Nevertheless, the DSP's operating frequency is set as a target of 30MHz. A summary of the specification is shown below in Table 5.

Description	*Specification*
Resolution	1024×900 (arbitrary chosen)
DSP units	32 (arbitrary chosen)
Frame-rate	30 fps (arbitrary chosen)
Pixel-pitch	20μm (arbitrary chosen)
DSP-pitch	560μm
Frequency	30MHz
Window size	5×5 (*N*=25)
Pipeline	2 stages
Latency	60 cycles
Throughput	1/30 speckle/cycle
Input precision	8-bit
Output precision	Q13.15

Table 5. Summary of specification

Design Flow



C/C++ model (gcc)
Fail output comparison
Create/Update and simulate behavioral/structural RTL (Synopsys vcs)
Hand designed gate netlist creation/changes (Cadence virtuoso schematic editor)
Fail logic comparison
Functional simulation (Cadence nc-verilog)
Design and insert/replace pulse-latch clock (Cadence virtuoso schematic editor)
Ways to improve timing
Logic relocation, Transistor sizing
Ways to fix hold time violation
Insert buffer, Logic relocation
Fail
Pre-layout corner timing simulation (Cadence virtuoso spectre circuit simulator)
Design Rule Check (Mentor Calibre DRC)
Layout design of custom cells (Cadence virtuoso layout editor)
Layout Verse Schematic (Mentor Calibre LVS)
Floor planning (Cadence virtuoso layout editor)
Layout Verse Schematic (Mentor Calibre LVS)
Hand placement of standard and custom cells for group components (Cadence virtuoso layout editor)
Parasitic Extraction (Mentor Calibre PEX)
Minimize by schematic over-design
Post-layout corner timing simulation (Cadence virtuoso spectre circuit simulator)
Design Rule Check (Mentor Calibre DRC)
Full chip layout (Cadence virtuoso layout editor)
Layout Verse Schematic (Mentor Calibre LVS)

Figure 46. Top level design flow

The top level design flow, Figure 46, first starts off by constructing a C/C++ model of the algorithm for verification. An equivalent behavioural/structural RTL is then designed and simulated in verilog, with its output compared with the C/C++ model for functional correctness. Following, a gate-level netlist is created manually, functionally simulated and logically checked for accuracy.

When all digital simulations are correct, the schematics of the pulse-latch custom cells are designed and inserted into the clock-tree of the gate-level netlist. A more accurate and conventional method is to test the digital design using more accurate transistor level simulation. This is only possible if the design is broken up into smaller component blocks so that simulations at sub-component level can then be performed time-efficiently. To reduce the time taken for transistor level simulation, the schematics are over-designed to reduce the expected failures at the post-layout stage.

Placement and routing is performed manually due to a lack of automatic placement and routing technology for mixed custom and standard cells supporting the targeted design kit. In addition, the lack of good timing analysis software for digital circuits supporting this design kit and dedicated/distributed simulation machines make it impossible to perform a top level parasitic extraction and timing simulation. The design flow then ends off with a top level *Design-Rule-Check* (DRC) and *Layout-Versus-Schematic* (LVS) to ensure the sub-component are connected correctly. Similar top level design flow can also be found in [72].

Design Principles

The general design approach is to minimise the power consumption while maintaining the specifications requirement. A summary of general design principles are written to ease the design methods.

1. While this is a custom design development, individual transistor sizing approach is not used. A more generic rule of thumb is used instead to infer the sizing from the standard library. Along the critical path, the standard sizing from the standard library is used and along the non-critical path, a 50% sizing of the smallest cell from the standard library is used. This will ease the layout design on the digital logic cells.
2. Dynamic logic is minimised due to higher power consumption while static logic is preferred as it is available from the standard library. However, it can be used minimally

to relax the critical path. Due to the high power consumption in self-timing clock for some dynamic logic (domino), it is only used when the clocked logic is aligned with the clock-edge.

3. It is estimated that the critical path in Figure 44 lies in the square-root unit. Therefore smaller transistors are used on the first pipeline stage and normal sized transistors are used in the second stage. This approach will minimise the operating supply voltage as the second stage is required to operate faster. There is a minimum operating supply voltage in this system and is explained in the memory section.
4. Due to the heavy reliance of clocking elements in BS architecture, a pulse-latch clocking strategy approach is employed.
5. Leakage power is not considered due to high V_{th} of the process parameters.
6. Other design methodologies such as clock-gating, glitch control and memory optimization are applied when necessary.
7. Simulations are performed at the sub-component level to ensure functional operation of the design. Post-layout simulations are performed with supply voltages of 1.2V and 1.8V at 27 ^{o} across corner variation.
8. Power consumptions are then measured at 1.2V, 30MHz, 27^o in typical process corners at the post-layout simulation using a 50% switching activity input approach.

CHAPTER IV

LSBF ARITHMETIC UNITS

This chapter describes the operations and simulations of all the *Least-Significant-Bit-First* (LSBF) arithmetic blocks in the DSP unit. The LSBF arithmetic blocks include an adder, a subtractor, a tree adder, a multiplier and a squarer. The following symbols shown below in Figure 47 will be reused in this chapter for illustration.

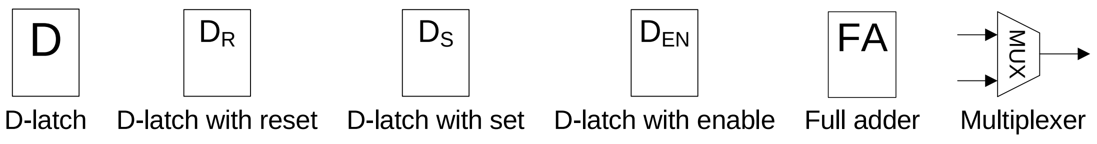
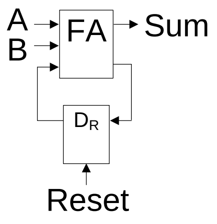


Figure 47. LSBF symbols

Bit-Serial Adder/Subtractor



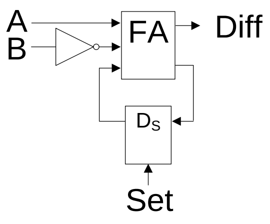


Figure 48. Bit-serial adder (Sum=A+B) [89]   Figure 49. Bit-serial subtractor (Diff=A-B) [89]

Bit-Serial (BS) adder/subtractor is also known as carry-save adder/subtractor as carry is saved from one bit position to the next. Figure 48 and Figure 49 shows the logical equivalence of a single BS adder and subtractor where numbers are input in two's complement representation from the least significant bit [89]. These adders can be used to perform addition and subtraction for any bit-width number sequence and effectively trade more operational cycles for smaller area implementations. A BS adder/subtractor has an operating cycle equal to the number of bits at the output. For example, two 8-bits addition will need 9 clock cycles as it has a 9-bit output. As the full adders are not cascaded, glitches are minimised as there is no ripple through effect. Another advantage of using BS adder/subtractor is efficient implementation of multi-input adders.

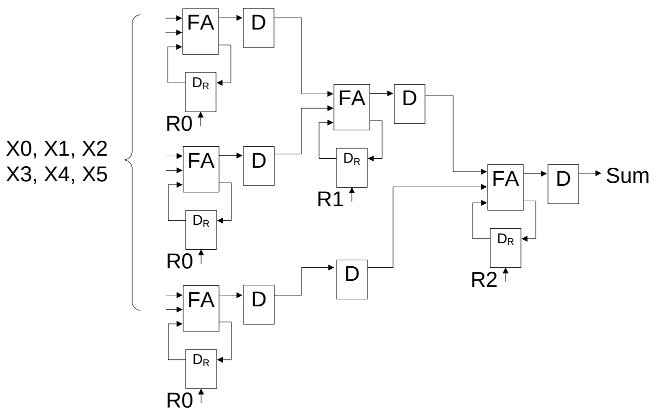


Figure 50. 6-input tree adder (Σ = X0+X1+X2+X3+X4+X5)

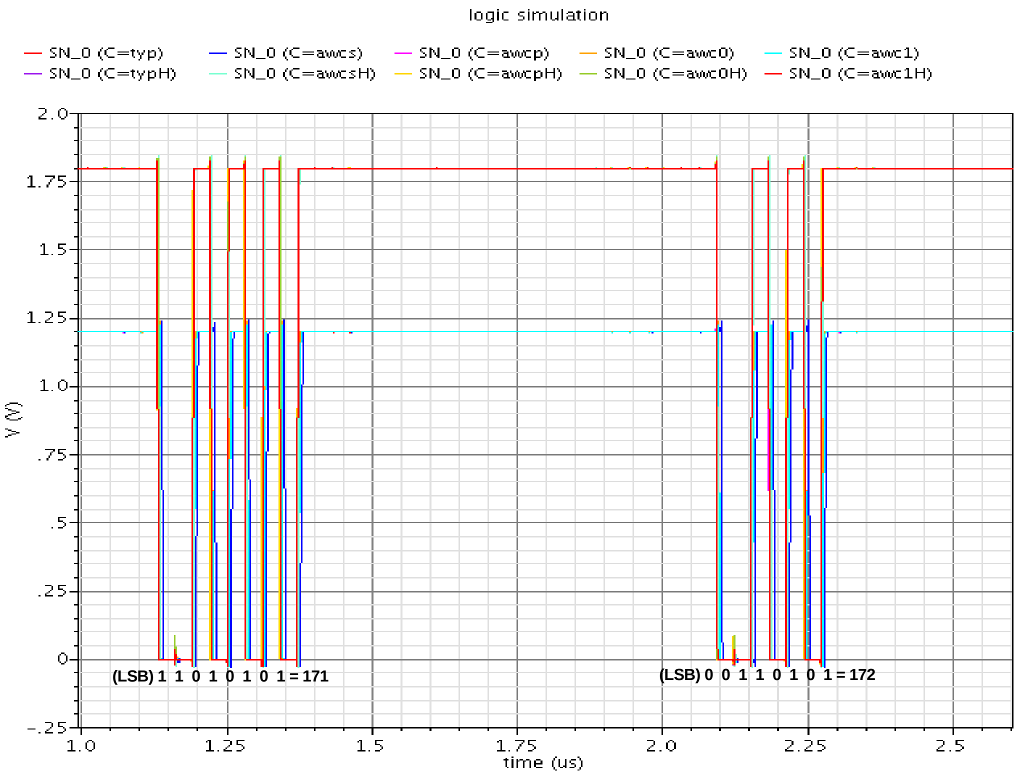


Figure 51. Post-layout simulation of Σ with 8-bit output (inverted output)

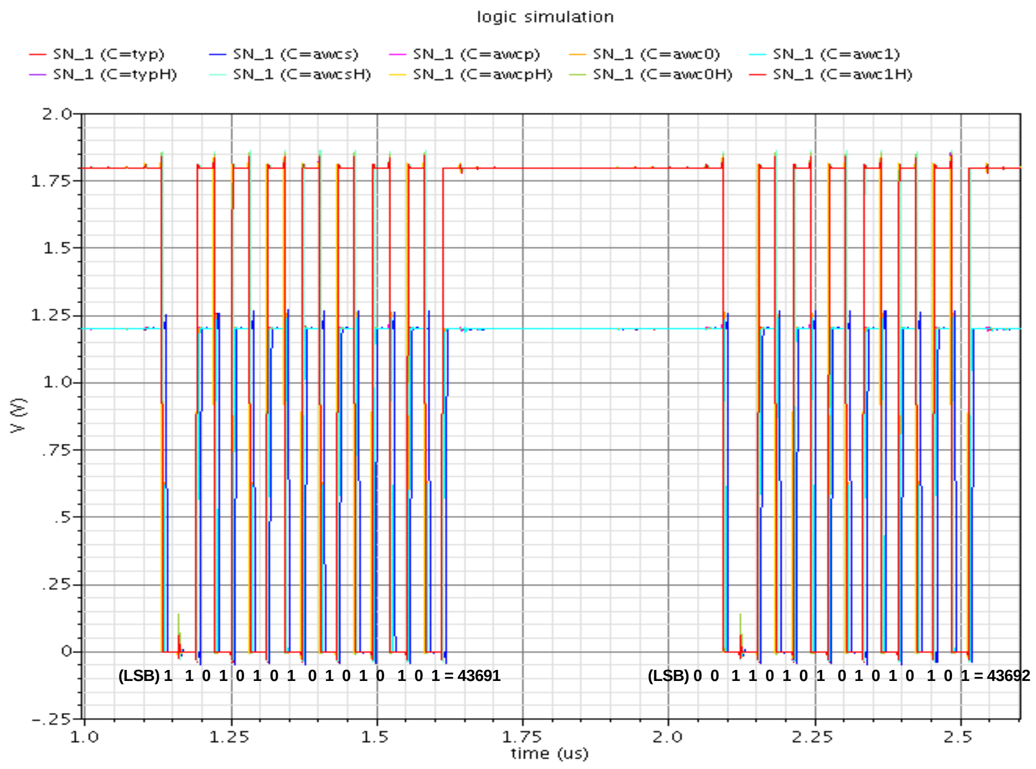


Figure 52. Post-layout simulation of Σ with 16-bit output (inverted output)

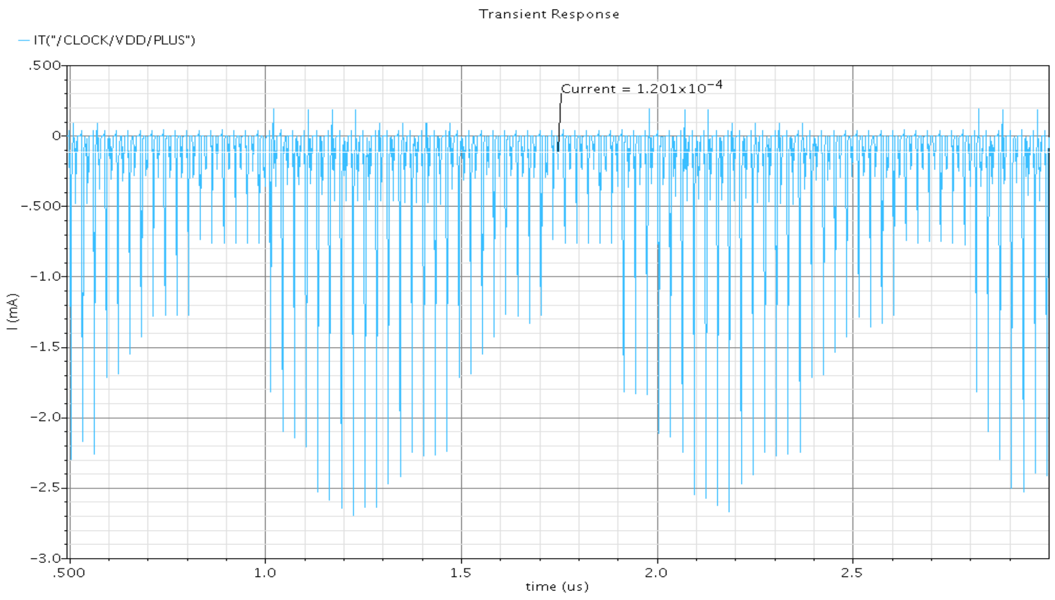


Figure 53. Current consumption of C30+CG1+2Σ

Consider a 6-bit adder in Figure 50, it uses only five full adders and latches to design a single Σ operator in Figure 44. This simple design allows maximum parallel addition for any bit-width input at the minimum area cost. Since the delay paths in these adders are least in the entire DSP, a typical mirror adder from Figure 30 can be reused from the standard library. It can be sized smallest even with a stack height of 6 and the inherent gate delay is also required to fulfil the hold time requirement of the latches. Two random post-layout simulations of the Σ operations are performed to obtain a 8-bit and 16-bit output. They are shown in Figure 51 and Figure 52 respectively with the total current consumption of the Σ operations in the DSP unit shown in Figure 53. These adders including the state controller account to about 120.1μA of current consumption.

Bit-Serial Multiplier

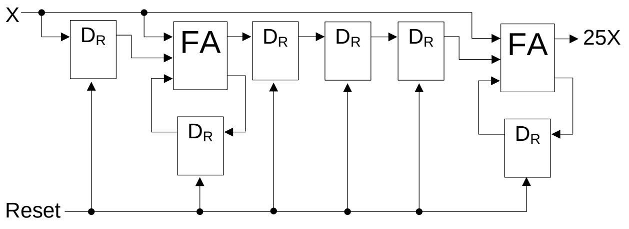


Figure 54. 25× bit-serial multiplier

The only multiplier required in (15) is a fixed N multiplication that is dependent on the window size of the processing algorithm, Figure 54. In this case, a 5×5 window size will set N=25. The easiest method to perform a multiplication is by using the shift-add method in binary form. A '1' at the k-th bit of the number will require a k-bit left shift before addition. As N = '11001' in binary, and input number will require every bit shifted by 0, 3 and 4 before addition. If a number is input bit-serially from the LSB, the input performs an addition at the 0th, 3rd and 4th shift. This is equivalent to a circuit implementation in Figure 54 where full adder is placed at the 0th position (right) and the 3rd position (3 delays after the first). The last adder at the 4th position is omitted as adding a zero is equivalent to itself. This small implementation is one of the most area efficient fixed-multiplier with the least transistor count.

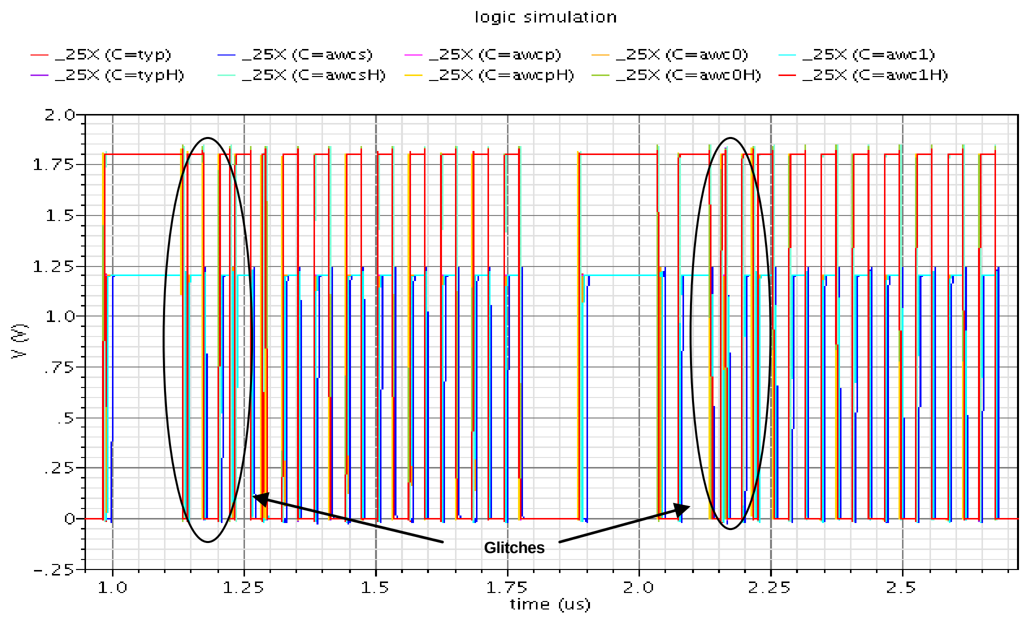


Figure 55. Post-layout simulation of 25×+1-bit subtractor

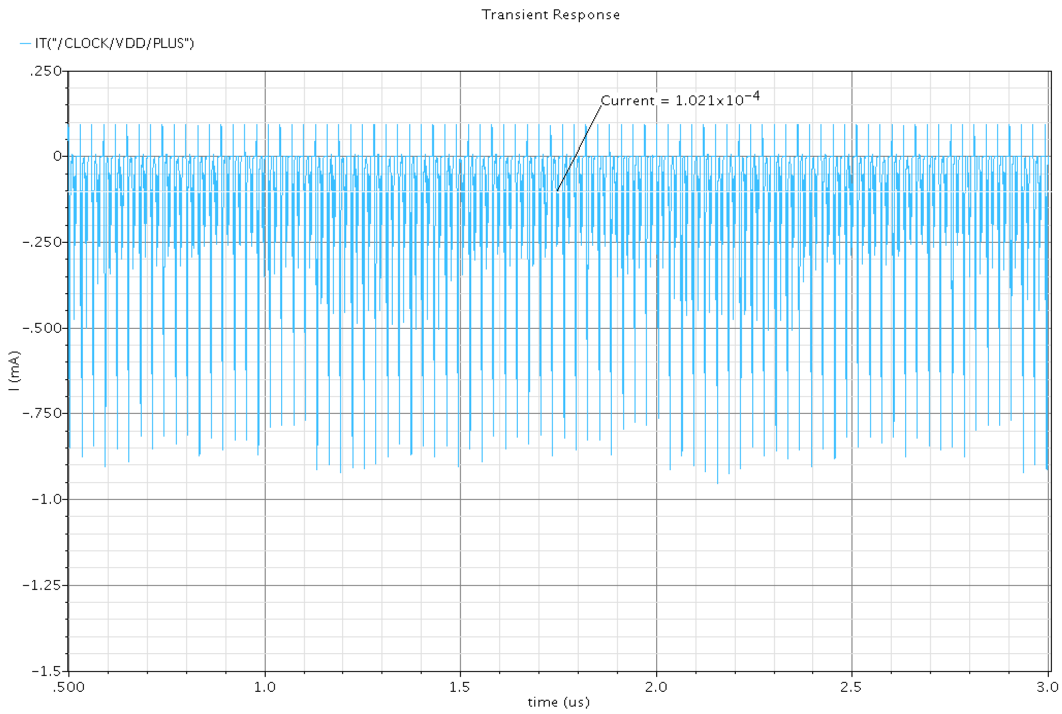


Figure 56. Current consumption of C30+CG1+25×+1-bit subtractor

The post-layout simulation of the multiplier is shown in Figure 55 and the current consumption is shown in Figure 56. As one of the 1-bit subtractor is close to the 25× BS-multiplier, both units are simulated together. A small amount of glitches are observed from

Figure 55 due to the rippling effect of the multiplier connecting to the subtractor, similarly to a 2-bit ripple carry adder. However, the ripples will be stopped by the next expected latch in a BS design. From Figure 56, the total current consumption accounts approximately to 102.1μA.

Bit-Serial Squarer

$$x = \sum_{i=0}^{n} x_i 2^i \tag{19}$$

$$\begin{aligned} x^2 &= \left(\sum_{i=0}^{n} x_i 2^i \right)^2 \\ &= \left(x_0 + \sum_{i=1}^{n} x_i 2^i \right)^2 \\ &= x_0 + 2x_0 \sum_{i=1}^{n} x_i 2^i + \left(\sum_{i=1}^{n} x_i 2^i \right)^2 \end{aligned} \tag{20}$$

Consider an n-bit number *x* represented in binary (19), where x_i is the *i*-th bit, being squared and expanded in (20). The final expanded version has three terms with the first term being itself (as squaring a bit is just itself). The second term represents an AND operation of the 0th bit with every single other bit and multiply by two. The last term is just a normal squaring operation but without the 0th bit of number *x*. The last term can be expanded similarly and thus the arithmetic design is an addition of the first two terms with the duplication of the design on the rest of the bits.

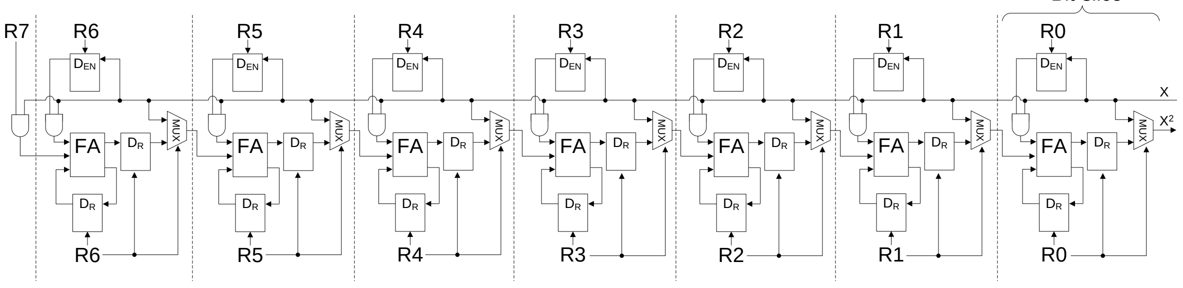


Figure 57. 8-bit bit-serial squarer

The implementation of the first two terms is shown as a single bit-slice in Figure 57, where the first bit output selects its own value, x_0, through a multiplexer and also stores the same value in a D-latch. It then performs an AND operation with other input bits and left shifts its own value by delaying one cycle (×2). The cell is then duplicated and added accordingly to form the third term. A full expansion of (20) will reduce the last term to the bit itself, resulting in an AND gate.

An 8-bit design is shown in Figure 57 and the 13-bit required BS-squarer for $(\Sigma I)^2$ in Table 1 can be obtained by padding with more bit-slices. As such, an n-bit squarer will require (n-1) bit-slice and an AND gate. It also requires $n \times n$ clock cycles to complete. Similar to the BS-adders, small sized mirror adders are used to make full use of the time slack available within the latches. This design is similar to one in [89] except that the D-latches are not all reset initially but sequentially by the control signal. Optimization of the BS-squarer is performed previously in [92] by recognizing similar computation terms and reusing the D-latches in the design. Such BS-squarer is more effective for long word-length BS-squarer and requires clocking for every cycle. In this design, the D-latch can also be recycled from a system level. Since the 13-bit BS-squarer stores ΣI within itself, data from the bit-squarer can be reused to the input of adder tree and divider as required by the algorithm and drawn in Figure 44. This will not be possible if the optimized BS-squarer is employed.

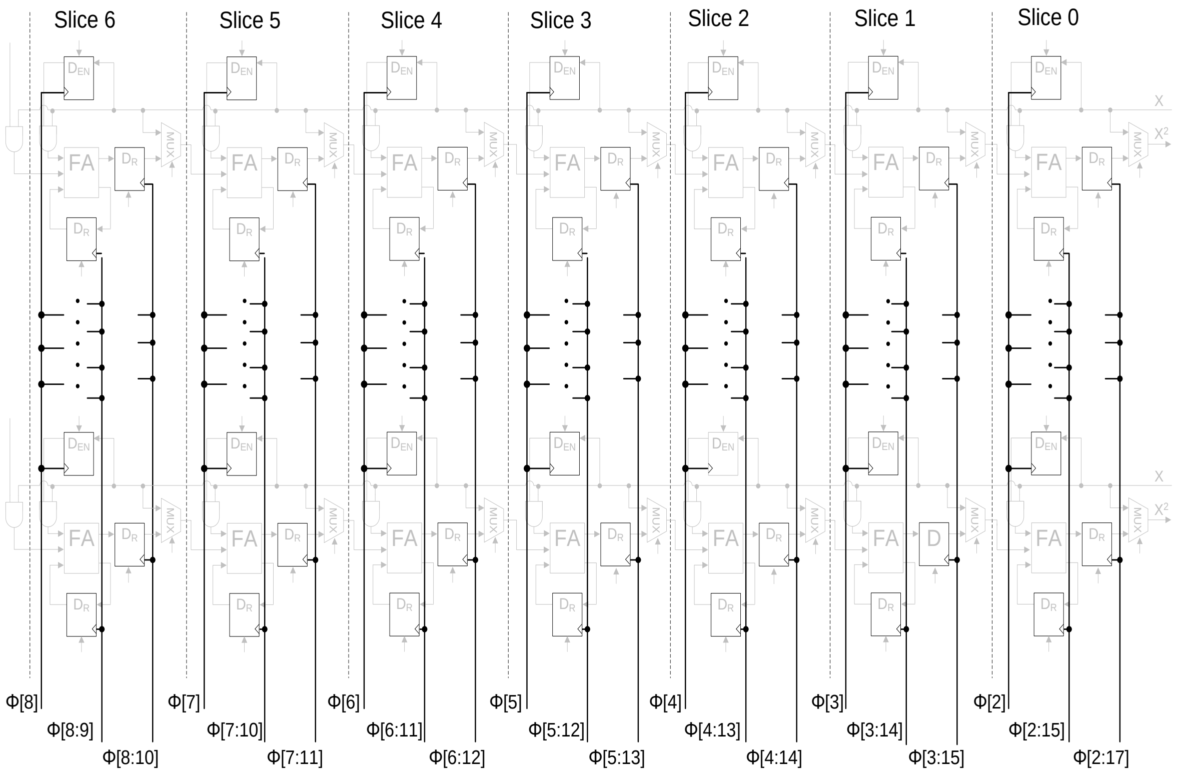


Figure 58. Clock-gating signals for bit-serial squarer

In the case of 8-bit BS-squarer, grouping ten of them together enable a precise clock-gating to the cycle requirement of every single D-latch. This is possible as the D-latches are not required to operate at every cycle due to the design of the sequentially controlled reset signals as described earlier. Consider slice 6 in Figure 58, it is only required to operate after the 6-bit input and can be turned off immediately after 3 cycles as zero is padded onwards. Subsequently, all the clock-gating signals can be derived by considering the input propagation across the bit-slices. The final required clock-gating signals are shown in Figure 58. Although the amounts of latches are significant, the actual required clock-cycles required is much lesser. Initially a total of 3 latch × 7 slices × 16 cycles × 10 units = 3360 cycles are required. After clock-gating, a total of (2+3+…+14+16+7) × 10 units = 1270 cycles are required.

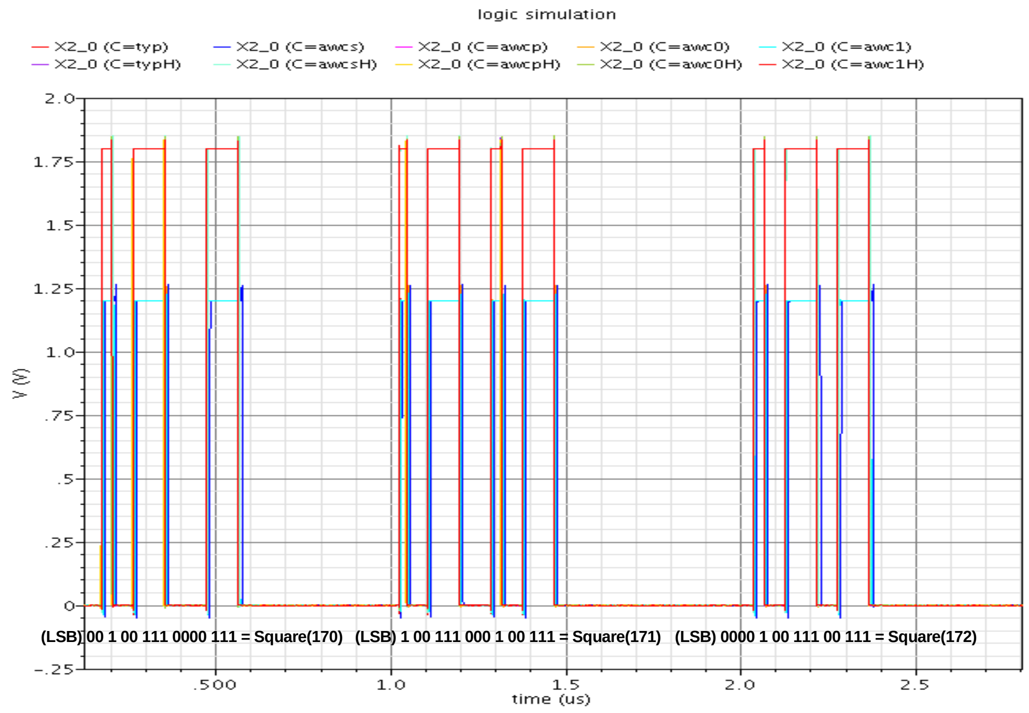


Figure 59. Post-layout simulation of 8-bit BS-squarer

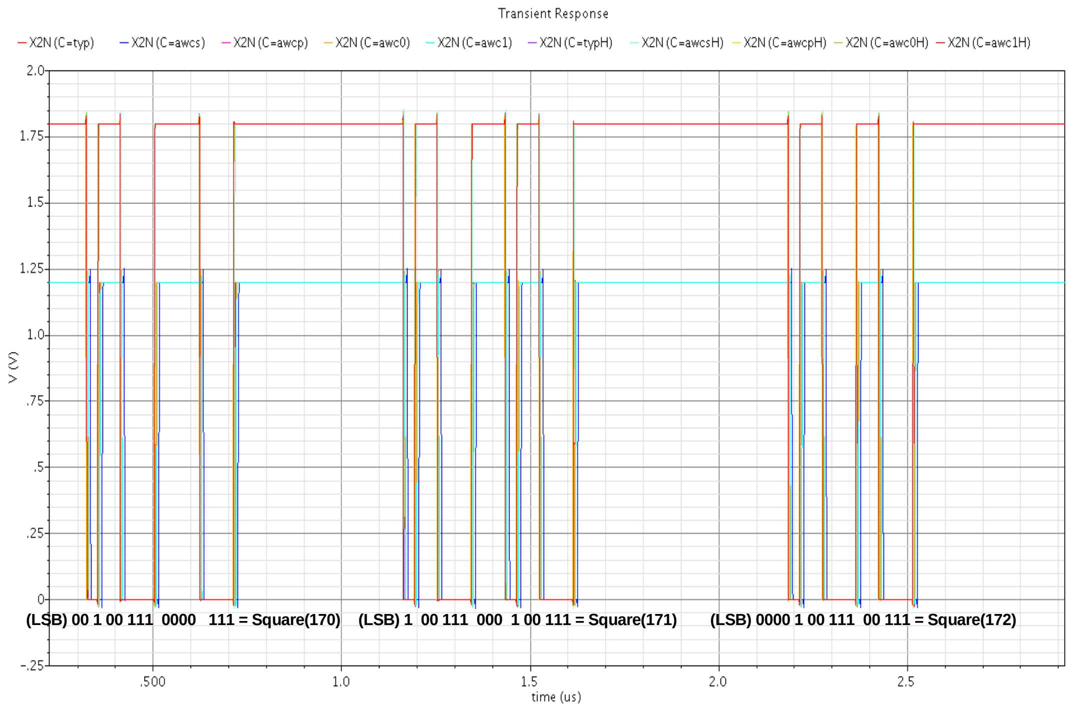


Figure 60. Post-layout simulation of 13-bit BS-squarer (inverted output)

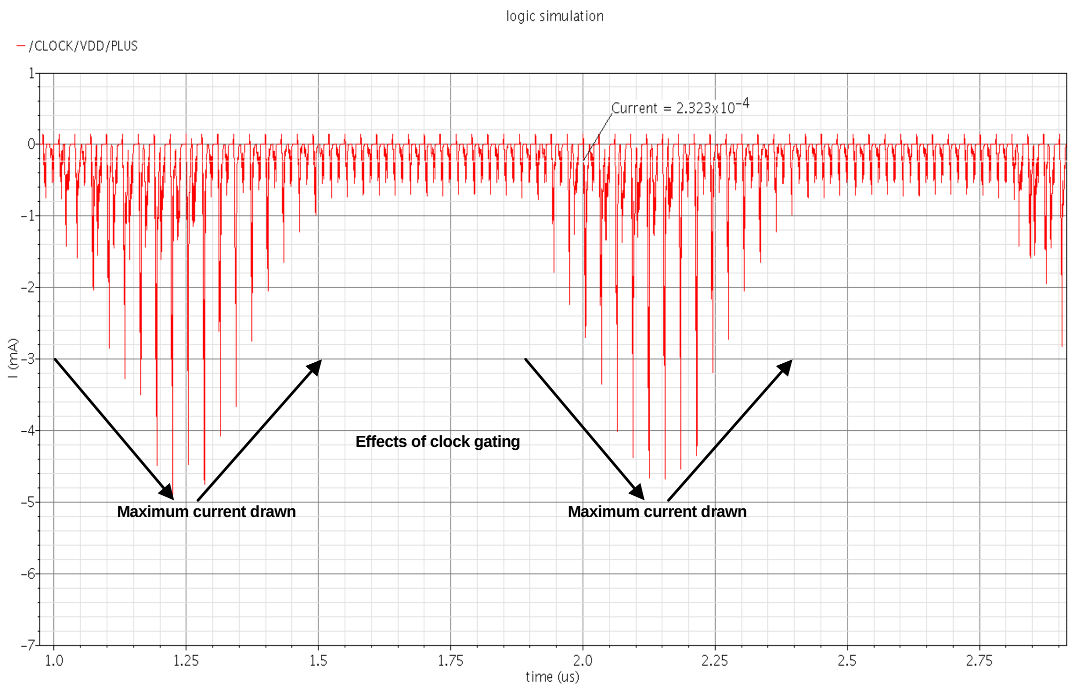


Figure 61. Current consumption of C30+CG0+10×8-bit squarer

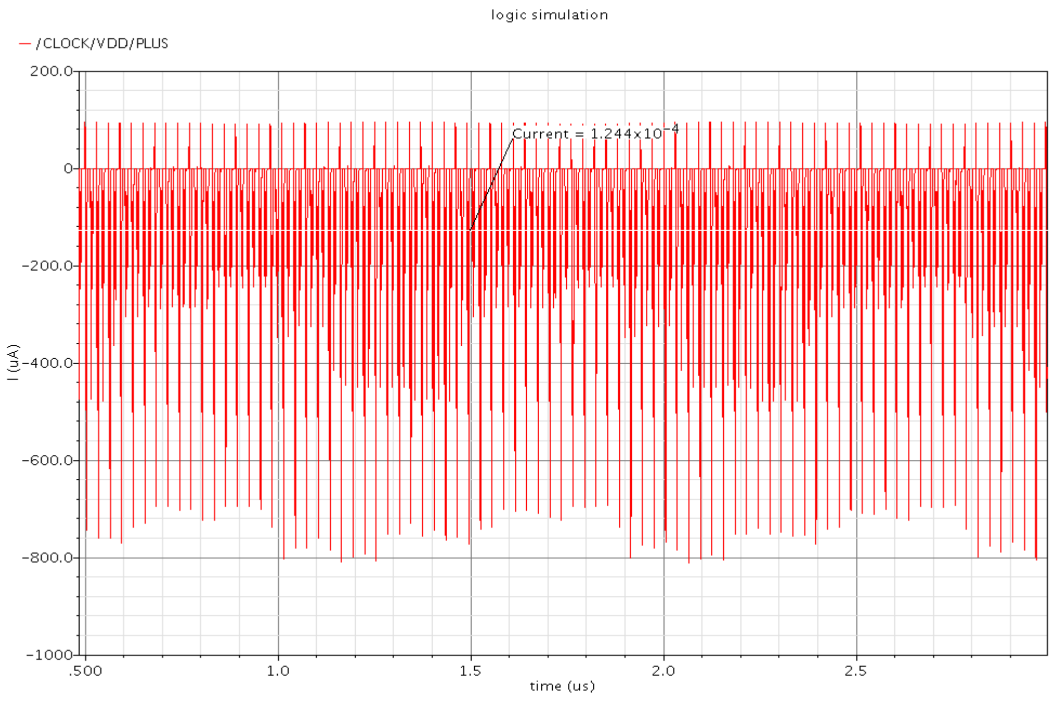
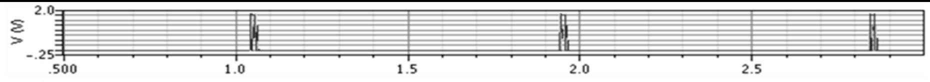
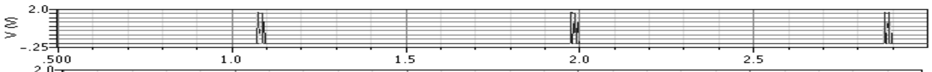
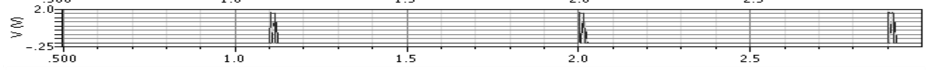
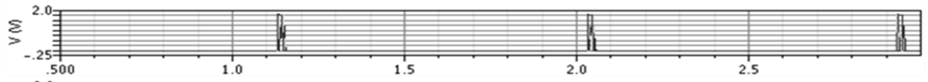
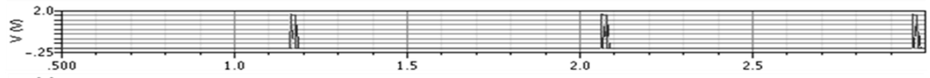
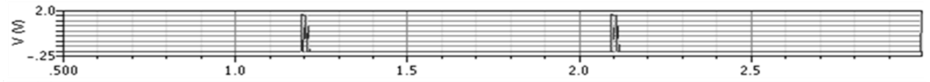
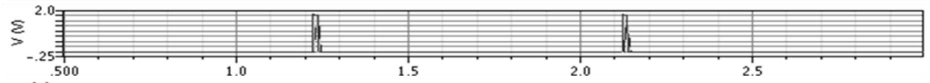
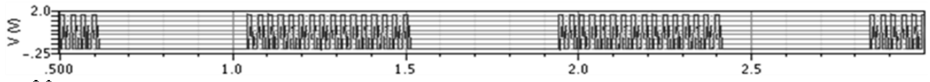
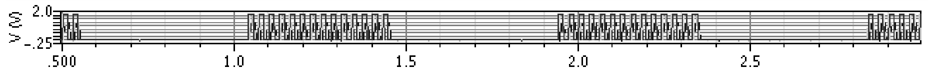
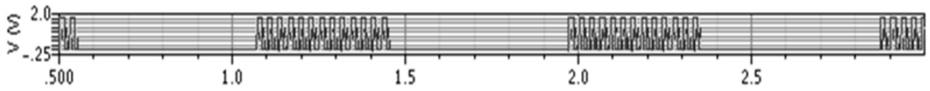
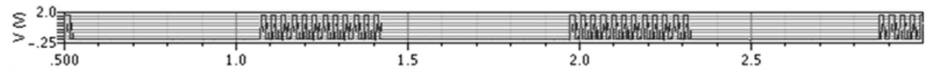
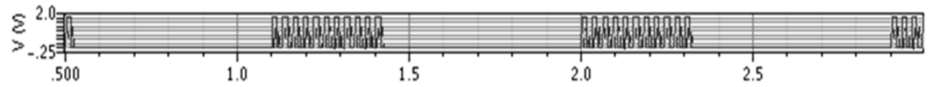
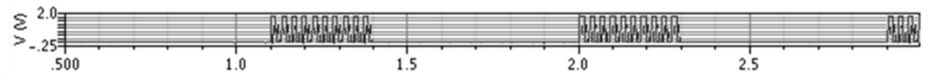
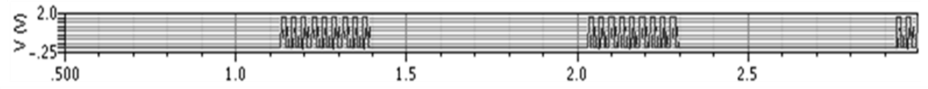
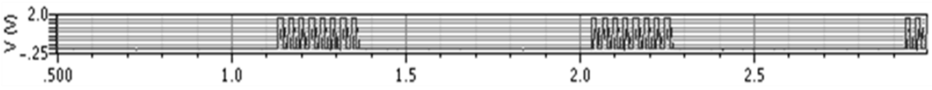
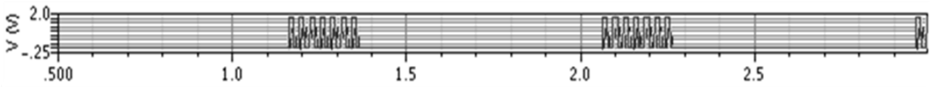
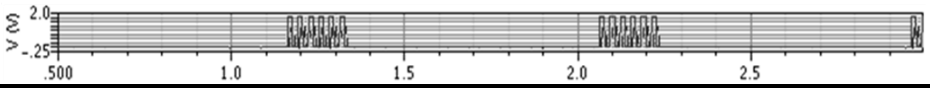


Figure 62. Current consumption of C30+CG1+13-bit squarer

A functional simulation of the 8-bit and 13-bit squarer is shown in Figure 59 and Figure 60 respectively while Figure 61 and Figure 62 show current consumption of

232.3μA and 124.7μA respectively. The effects of clock-gating can also be noticed from Figure 61 as the current drawn increase linearly to a maximum point where all the latches are clock simultaneously and decrease linearly afterwards. The post-layout simulations of the gated clocks are also shown below in Figure 63.

Unit	*Clock*	*Waveform*
8b-x^2	ϕ[2]	
	ϕ[3]	
	ϕ[4]	
	ϕ[5]	
	ϕ[6]	
	ϕ[7]	
	ϕ[8]	
	ϕ[2:17]	
	ϕ[2:15]	
	ϕ[3:15]	
	ϕ[3:14]	
	ϕ[4:14]	
	ϕ[4:13]	
	ϕ[5:13]	
	ϕ[5:12]	
	ϕ[6:12]	
	ϕ[6:11]	

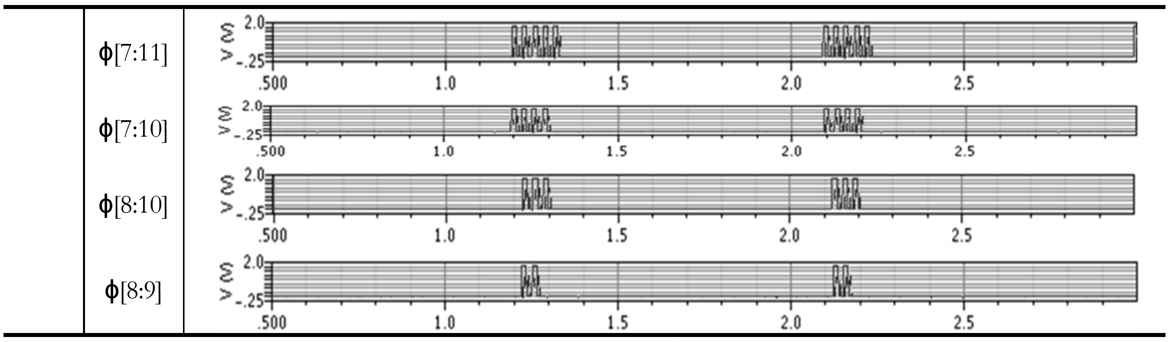


Figure 63. Post-layout simulation of gated-clocks in BS-squarer

Power Consumption

Arithmetic Units	*Power*
Σ	37.1μW
25× Multiplier	19.1 μW
x^2(10×8-bit)	135.6 μW
x^2(13-bit)	41.7 μW

Table 6. Power consumption of LSBF arithmetic units

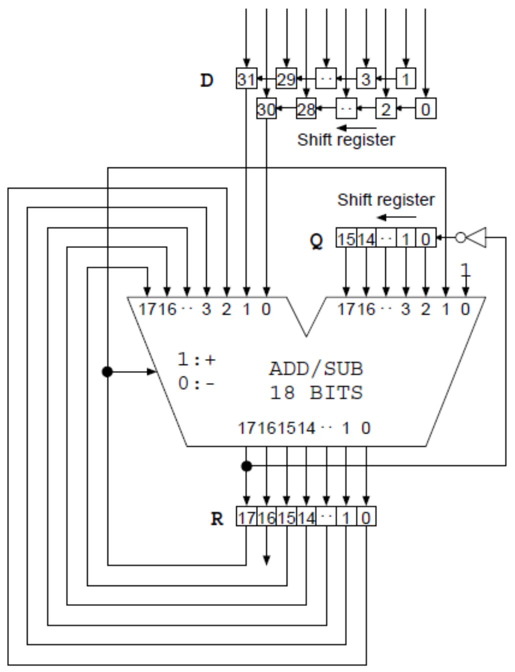
The power consumption for the LSBF arithmetic units have been extracted and summarized in Table 6.

CHAPTER V

MSBF ARITHMETIC UNITS

This chapter describes the operations and simulations of all the *Most-Significant-Bit-First* (MSBF) arithmetic blocks in the DSP unit. The MSBF arithmetic blocks include a square-root and a divider unit. The design of a bit-parallel adder is also discussed.

Bit-Serial Square-Root



<u>Pseudo code</u>

```
$d_k \leftarrow$ k-th bit of input digit to be square-rooted
$q \leftarrow 0$ (partial square-root)
$r \leftarrow 0$ (partial remainder)

for $i \leftarrow$ maximum-output-bit to 0 by 1
    if r is non-negative
        $r \leftarrow rd_{2i+1}d_{2i} - q01$
    else
        $r \leftarrow rd_{2i+1}d_{2i} + q11$
    if r is non-negative
        $q \leftarrow q1$
    else
        $q \leftarrow q0$
end
```

Figure 64. Non-restoring square-root [88]

In Figure 64, a radix-2 non-restoring square-root algorithm using a small sized iterative architecture is presented from [88]. The algorithm scans two input bits from the MSB and attempts to find the integer portion of the square-root (one bit per iteration) from the partial remainder padded with the two input bits. Since the square-root of two bits (00, 01, 10, 11) can only takes the value of 0 for ‘00’ or 1 for values greater than ‘01’, the basic idea of the algorithm is to determine the next partial remainder by subtracting its own value padded with two input bits (r$d_{2i+1}d_{2i}$) by q01 to determine if the remainder is q00 or greater than q01. If it is greater than q01 (a non-negative partial remainder) then the next partial square-root must be q1 otherwise it is q0. The addition of q11 when the remainder is negative is to compensate for the over subtraction in the previous iteration. A proof for the addition of q11 can be found in [88].

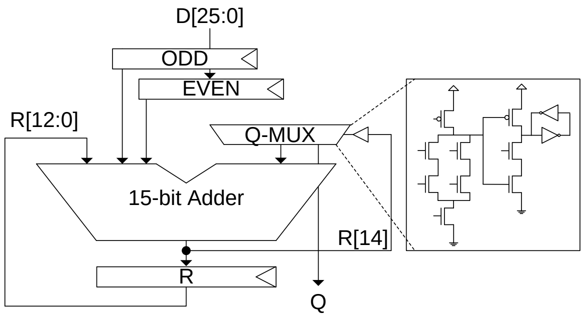


Figure 65. 26-bit square-root unit with adder front-end using dynamic multiplexer latch

This simplicity makes the design of the square-root compact by using a single selectable adder or subtractor. To perform a 13-bit output square-root, $\sqrt{(\,N\Sigma I^2\text{-}(\Sigma I)^2)}$ from Table 1, a 15-bit adder is required as two extra bits are required to add/subtract the padded the partial square root. To speed up the operation of the square-root unit, an adder front-end with dynamic multiplex latch is used to invert the bits of the partial square-root to perform a 2's complement subtraction, Figure 65. (Dynamic multiplex latch can be found in Figure 100 on page 88.) The inverting operation has been shifted to a non-critical portion of the register effectively and the critical path delay is slightly reduced to that of the 15-bit adder to the edge of the multiplexer latch. This design can also be found in modern processor's *Arithmetic-Logic-Unit* (ALU) adder front-end design where dynamic multiplexer is used to perform subtraction operation, set '0' or set'1' operations [74]. The final output Q in Figure 65 is generated one digit per cycle from the MSB as described from the algorithm.

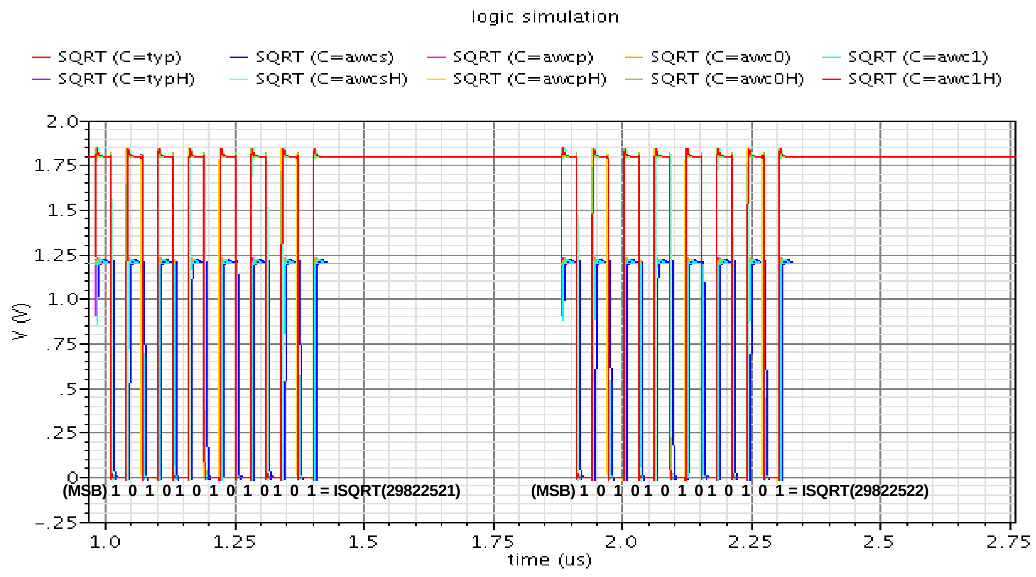


Figure 66. Post-layout simulation of square-root (inverted)

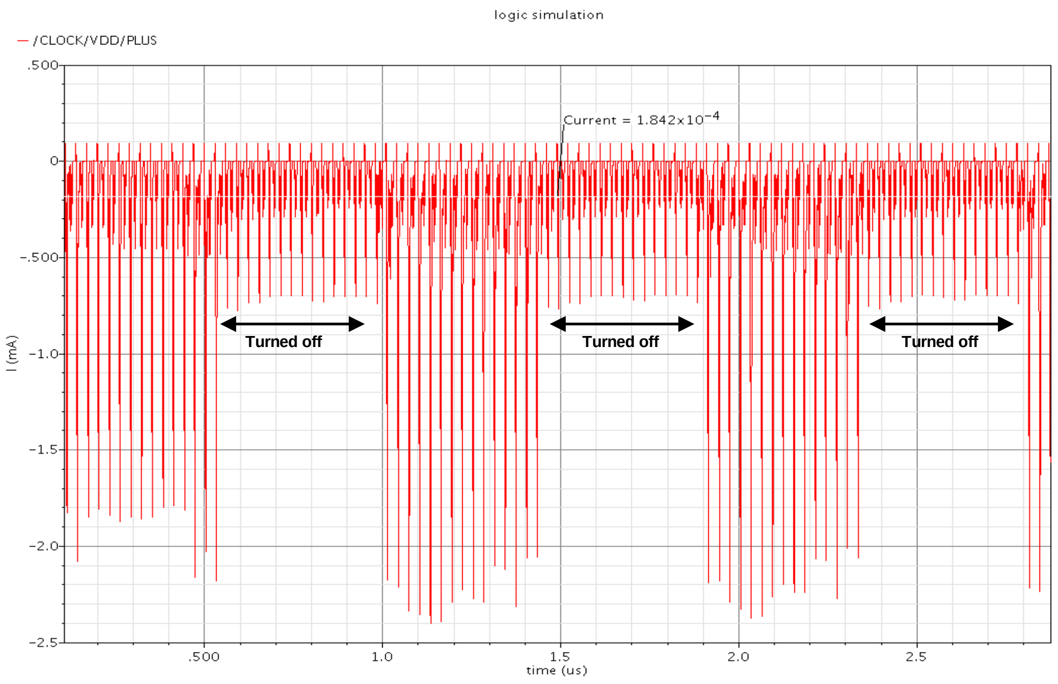
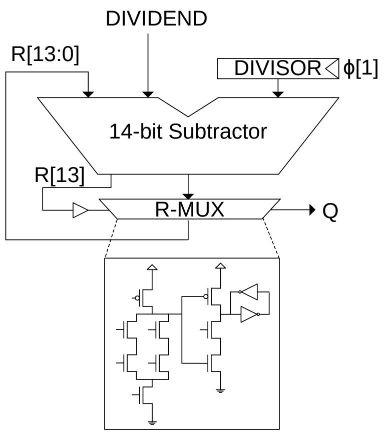


Figure 67. Current consumption of C30+CG1+square-root

The simulated outputs shown in Figure 66 and Figure 67 report a total of 184.2μA in current consumption. The effects of clock-gating are also noticeable from Figure 67 as the square root unit is turned off when it is not required to operate.

Bit-Serial Divider



Pseudo code

```
$n_k \leftarrow$ k-th bit of input dividend
$d \leftarrow$ divisor
$q \leftarrow 0$ (partial quotient)
$r \leftarrow 0$ (partial remainder)

for $i \leftarrow$ maximum-output-bit to 0 by 1
    if ($rn_i - d$) is non-negative
        $r \leftarrow rn_i - d$
        $q \leftarrow q1$
    else
        $r \leftarrow rn_i$
        $q \leftarrow q0$
    end
end
```

Figure 68. Subtractive division

In Figure 68, a subtractive divider is presented which follows the conventional time-consuming division by hand algorithm. The algorithm repeatedly subtracts the divisor from the partial remainder padded with the input dividend. When the partial remainder is greater than the divisor (non-negative subtraction), the next quotient bit is set to '1' otherwise '0'. The next partial remainder then gets loaded with the subtracted remainder or shifted to a new bit precision. Similar to the square-root unit, this loading and shifting operation can be shifted to a non-critical portion of the register by using a dynamic multiplexing latch. A 14-bit subtractor is required to perform a 13-bit division as the remainder is padded with a single bit dividend per iteration. Since the divider uses one less bit than the square-root unit in the bit-parallel operation, the dominant critical path delay still lies in the square-root unit.

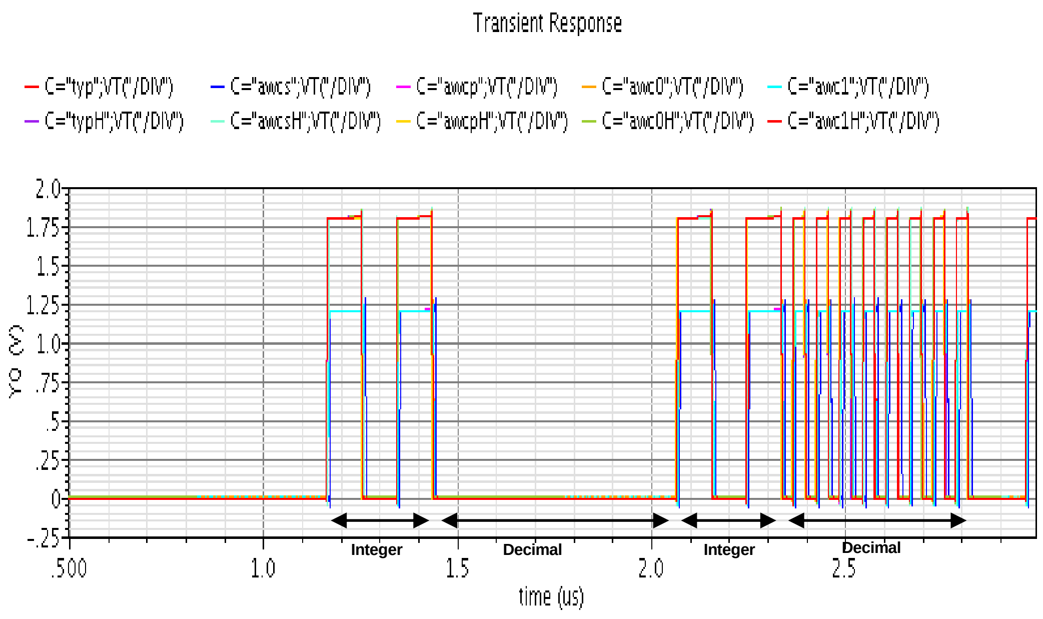


Figure 69. Post-layout simulation of divider

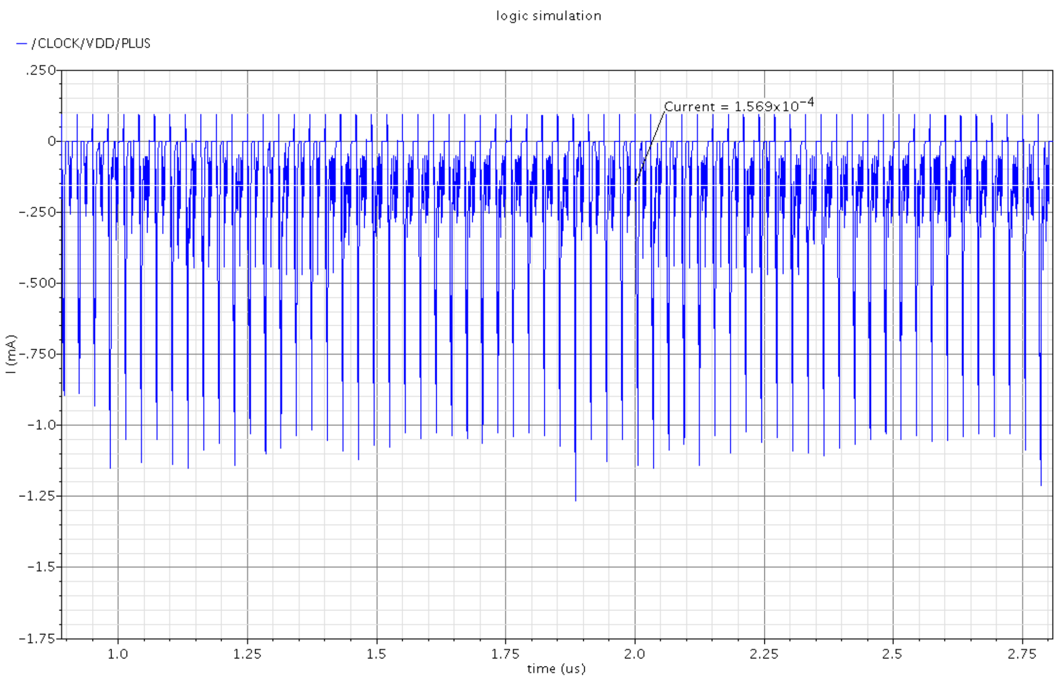


Figure 70. Current consumption of C30+CG1+divider

As the iterations can continue infinitely, the divider can perform an infinite precision of division. Two divisions are illustrated in Figure 69, where one output does not have any decimal digits and one with infinite decimal digits. The second output with decimal digits is terminated after 30 cycles have been used up. In Figure 70, simulation results in an average current consumption of 156.9μA.

Bit-Parallel Adder

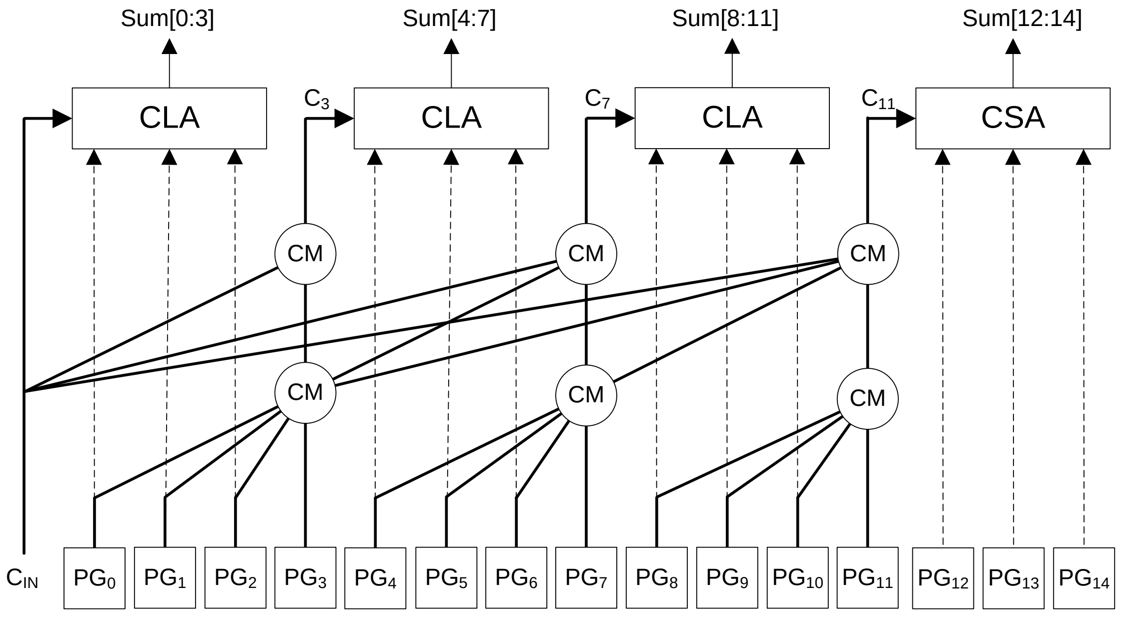
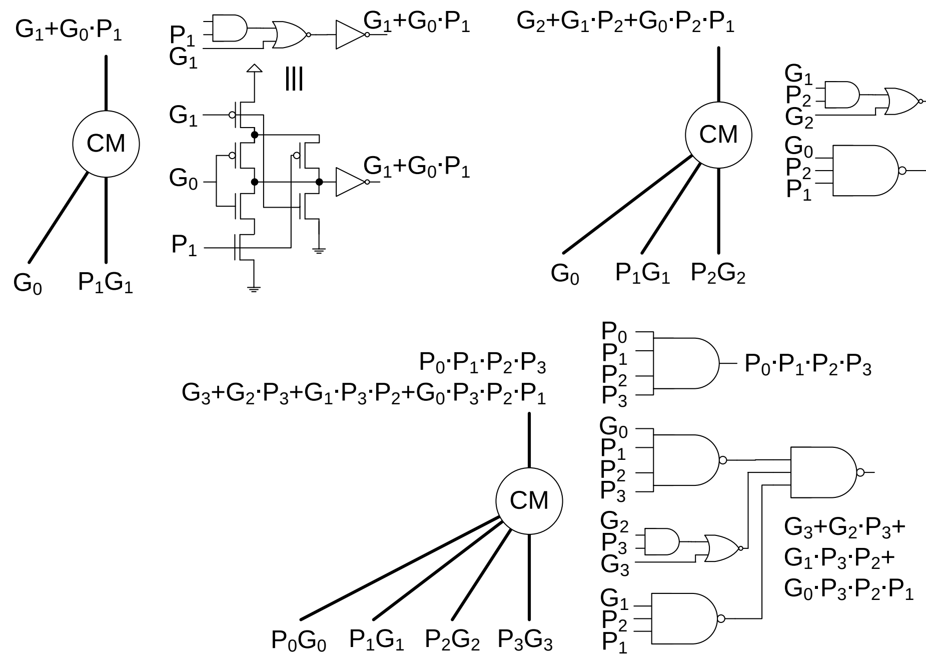


Figure 71. Sparse radix-4 15-bit CLA adder

In both the divider and square-root unit, bit-parallel adders/subtractors are being used. In actual realization, a subtractor is simply a 2s-complement adder and thus the logic design in the divider and square-root unit can be reused. In this work, the critical path of the DSP lies in the 15-bit adder of the square-root unit and an optimised adder will determine the performance of the DSP. Due to the small bit-width requirement, complex adder architecture is not required. However, a fast and energy-efficient adder working at low voltage is still required. Previously, sparse CLA adders have demonstrated energy savings in adder architectures due to fewer *Carry-Merge* (CM) operations in the adder tree [86]. In Figure 71, a sparse radix-4 tree design is illustrated to demonstrate the calculation of the every fourth carry in the adder (C_0, C_3, C_7, C_{11}). The carry signals are then used to generate the sum output with a slower group of non-critical *Propagate-Generate* (PG) signals corresponding to the dotted lines in Figure 71. In this radix-4 tree, the CM stages are effectively reduced to two in the critical path and are closer to the optimal number of stages with lower output loads [86].



$G_1+G_0 \cdot P_1$
CM
G_0
P_1G_1
P_1
G_1
$G_1+G_0 \cdot P_1$
G_0
$G_1+G_0 \cdot P_1$
$G_2+G_1 \cdot P_2+G_0 \cdot P_2 \cdot P_1$
CM
G_0
P_1G_1
P_2G_2
$G_2+G_1 \cdot P_2+G_0 \cdot P_2 \cdot P_1$
$P_0 \cdot P_1 \cdot P_2 \cdot P_3$
$G_3+G_2 \cdot P_3+G_1 \cdot P_3 \cdot P_2+G_0 \cdot P_3 \cdot P_2 \cdot P_1$
CM
P_0G_0
P_1G_1
P_2G_2
P_3G_3
$P_0 \cdot P_1 \cdot P_2 \cdot P_3$
$G_3+G_2 \cdot P_3+G_1 \cdot P_3 \cdot P_2+G_0 \cdot P_3 \cdot P_2 \cdot P_1$

Figure 72. CM operations and their CMOS implementation

The performance of the tree now lies in the implementation of PG and CM stages where the sparseness has reduced the number of fan-outs required per stage compared to a full tree. In Figure 72, the CM operations and their equivalent static CMOS implementation are shown. Dynamic logics are not considered due to their higher power consumption in their pre-charge and self-timed clock trees even though it is the preferred choice in modern microprocessor. All CM stages have an equal amount of two gates delay and are rearranged with balanced network paths that are less vulnerable to glitches.

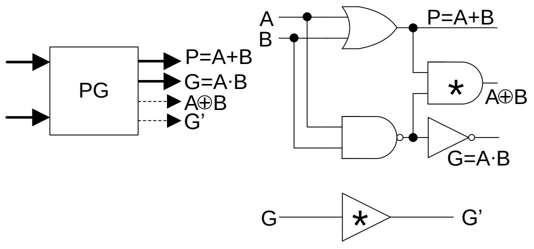


Figure 73. Propagate and generate (*: minimum sized)

To avoid an XOR gate in the critical path, an OR gate is used to create the propagate signal along the critical path and the XOR operation is reconstructed using minimum sized transistors along the non-critical path. The generate signal is created from a normal AND gate and buffered to form the non-critical path. The dotted lines in Figure 73 are the non-critical path and are represented similarly as dotted lines in Figure 71.

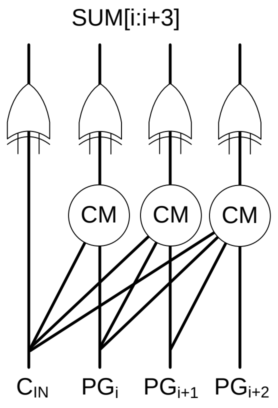


Figure 74. 4-bit full radix-4 CLA adder

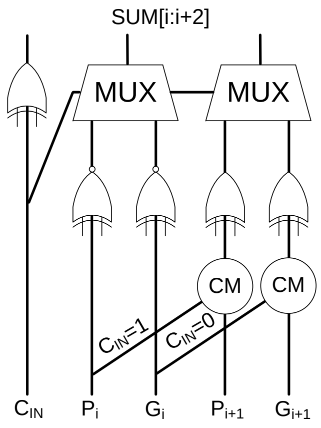


Figure 75. 3-bit non-critical sum generator

In the square-root unit, the longest path depends on the MSB bit to perform a multiplexing select operation. Therefore the sum generation for the remaining bits can be further relaxed to reduce on the node operations. A 4-bit full radix-4 CLA with lesser operation node count can be used instead of a CSA adder. In the critical MSB block, a similar CSA scheme to [93] is employed to reduce the critical path of the MSB bit to one multiplexer stage, Figure 75, instead of one CM plus one XOR stage, Figure 74. In both the CSA and CLA adders, most of the logic gates lie in the non-critical paths and a smaller transistor sizing can be used.

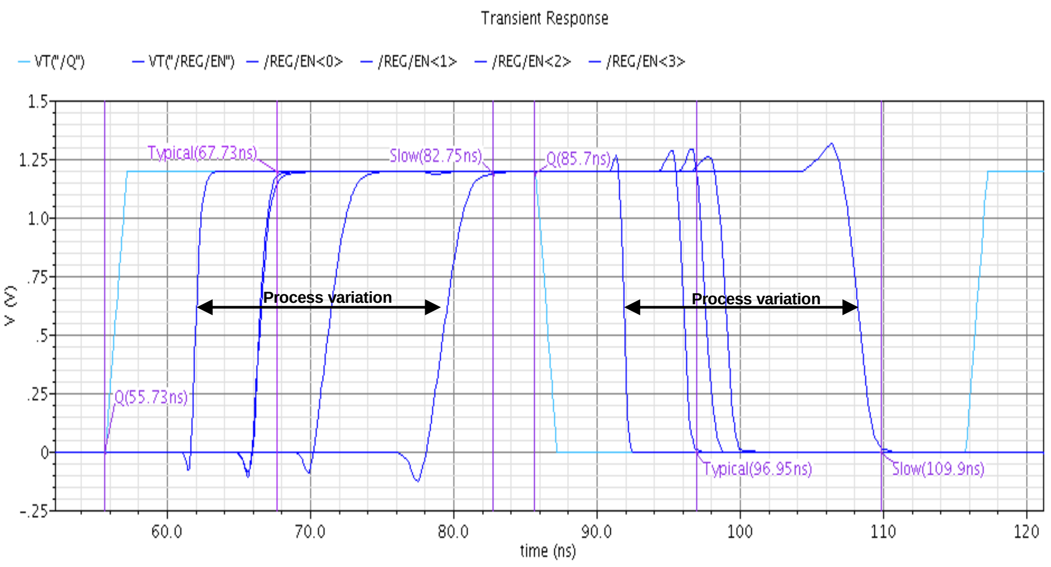


Figure 76. Critical path delay of adder in square-root at Vdd=1.2v

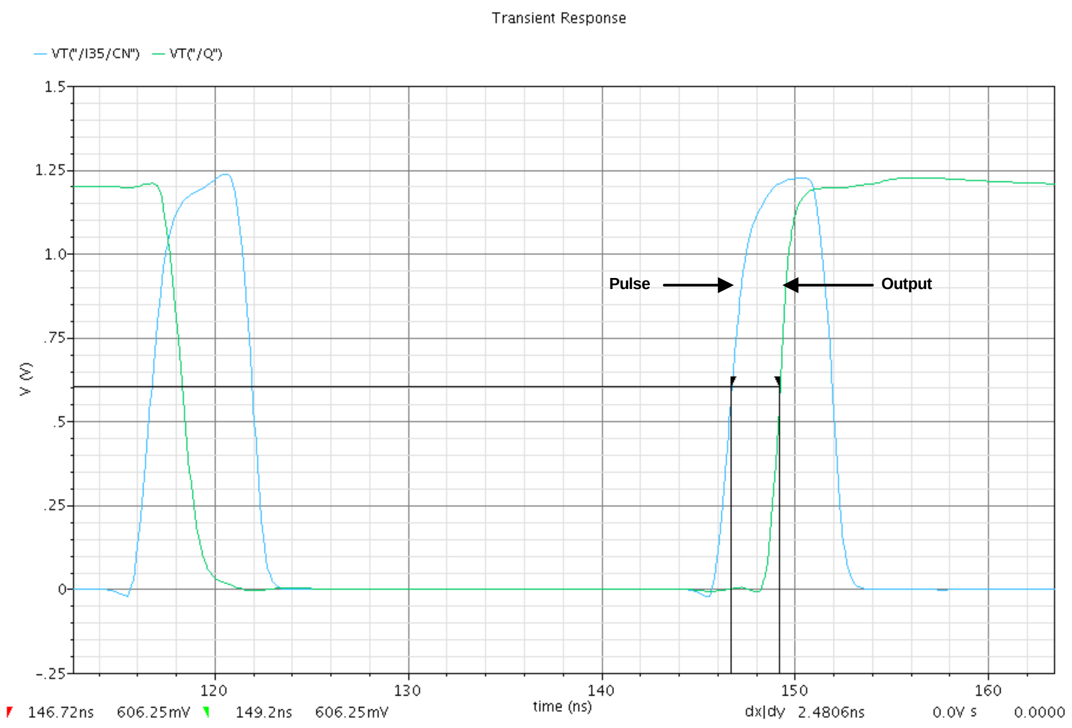


Figure 77. Latch delay at worst process corner

The critical path of the adder can be measured by triggering a carry propagation chain from the LSB as there is a maximum fan-out in the CM stages. The propagation delay from carry in can be ignored since it is only merged on the second stage of the adder tree. With consideration of the CM stage arrangement in actual implementation, the worst input pattern can be estimated from A(000…10),B(111…01) → A(111…11),B(000…10) with Cin(1). The post-layout simulation of the critical path of the adder to the edge of the multiplexer latch in the square-root unit is shown in Figure 76. Operating at supply voltage of 1.2V, the critical path delay suffers from a wide process variation. The slowest process corner yields a delay path of around 27ns while it only yields a 12ns delay in typical process corner. Coupled with a near 0s setup time and 2.5ns worst delay latch, Figure 77, the total worst delay is approximately 29.5ns and fits perfectly into a 30MHz clock.

Power Consumption

Arithmetic Units	*Power*
√	101.2 μW
÷	73.9 μW

Table 7. Power consumption of MSBF arithmetic units

The power consumption for the MSBF arithmetic units have been extracted and summarized in Table 7.

CHAPTER VI

SYSTEM DESIGN

This chapter describes the design and simulations from the system perspective. The system level design includes the design of the state controller, the memory interface and the clocking strategy. They have been highlighted below in the top level architecture block diagram, Figure 78. The functional verification is also presented to outline the functional correctness of the system design.

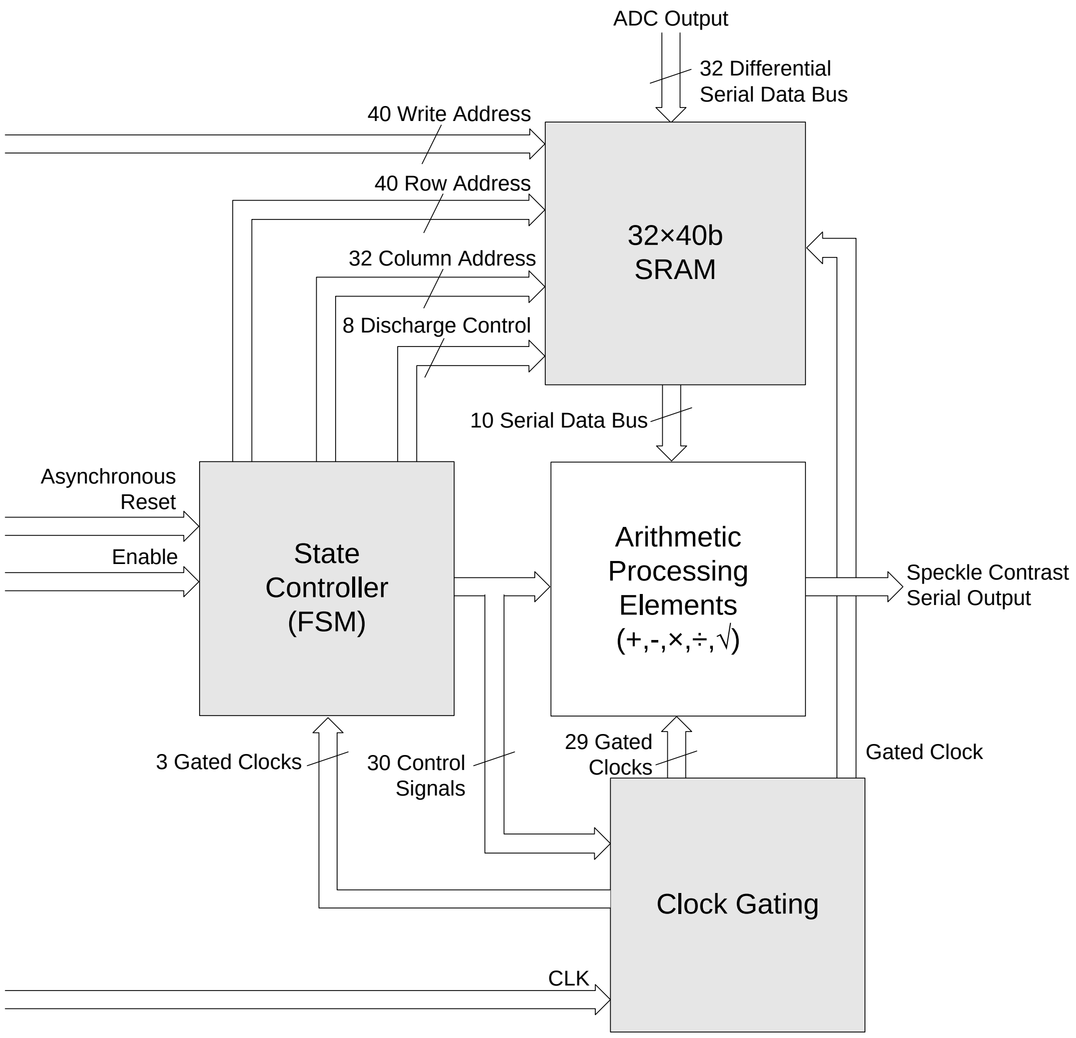


Figure 78, Top level architecture block diagram

Finite State Machine

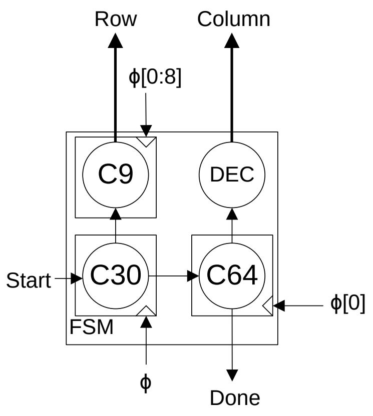


Figure 79. Finite state machine block diagram

The *Finite-State-Machine* (FSM) of the system, Figure 79, is made up of four sub-blocks, three counters and a 5-to-32 decoder. The components are clock-gated to reduce the power consumption and to simplify the design logic.

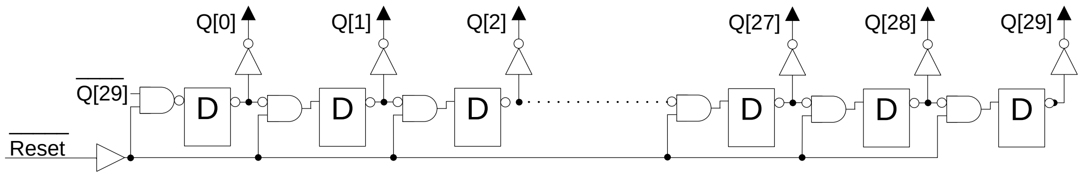


Figure 80. 30-bit shift register

The first counter, C30, is formed by a 30-bit shift register, Figure 80, and is used to track the number of cycles executing in both the pipeline stages. The main functionality is to provide control signals and clock gating signals for the rest of the circuits. It uses a one-hot encoding scheme where one bit represents a single state of the pipeline execution. In this case, a single state represents a single cycle and is propagated from cycle 0 to 29. The choice of one-hot encoding over the other encoding methods is primarily due to the requirement of faster control signals in some of the arithmetic blocks. Since there is only one latch propagating a '1' at any time, the corresponding pre-charge latch from Figure 97 (page 88) is preferred over Figure 98 (page 88) without significant increase in power consumption. Additional attention is also required to ensure the path delay between the latches fulfils the hold time requirement.

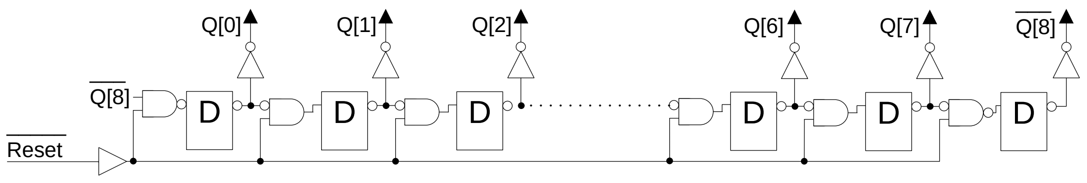


Figure 81. 9-bit shift register

Similar to the 30-bit shift register, the second counter, C9, is a 9-bit shift register, Figure 81, and uses a one-hot encoding scheme. The main functionality of this counter is used to perform the row select for the memory block. The first 8-bits are used to select the data from the memory block in a bit-serial manner while the last bit is used to discharge the memory bus when it is not in used. Subsequently, the clock required to power the counter is turned off and thus it operates from cycle 0 to 8. To achieve a known state during power up procedure, the counter is reset to one so that it is able to perform a read operation on the first bit at cycle 0.

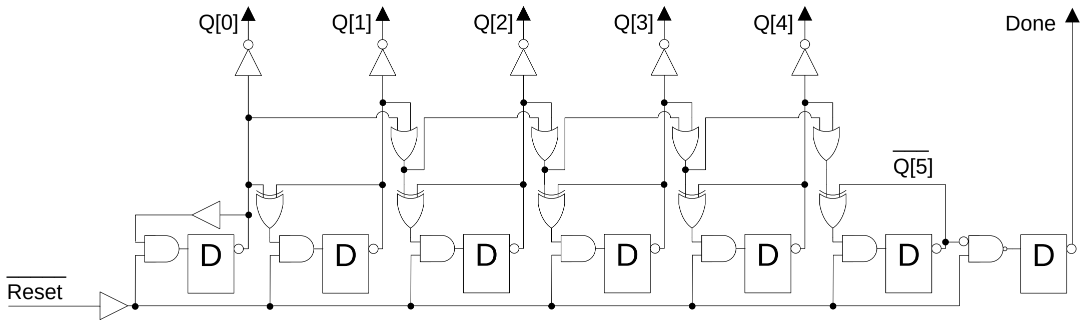


Figure 82. 6-bit synchronous count up counter

The third counter, C64, is a 6-bit synchronous count up counter, Figure 82, and uses a binary encoding scheme that counts from 0 to 63. The first four bits of the counter counts from 0 to 31 in binary and is used to select the column of the memory block after decoding. Since the column of the memory block is only required to change per generated contrast, it is clocked only once every 30 cycles at cycle 0. Similar to the previous counter, it is reset to zero to achieve a known state during the first start sequence. The last bit is then delayed by one gated-clock to indicate a complete scanning sequence of the data in the memory block. This one gated-clock delay is to force the last processing data from the first stage to the second stage of the pipeline execution.

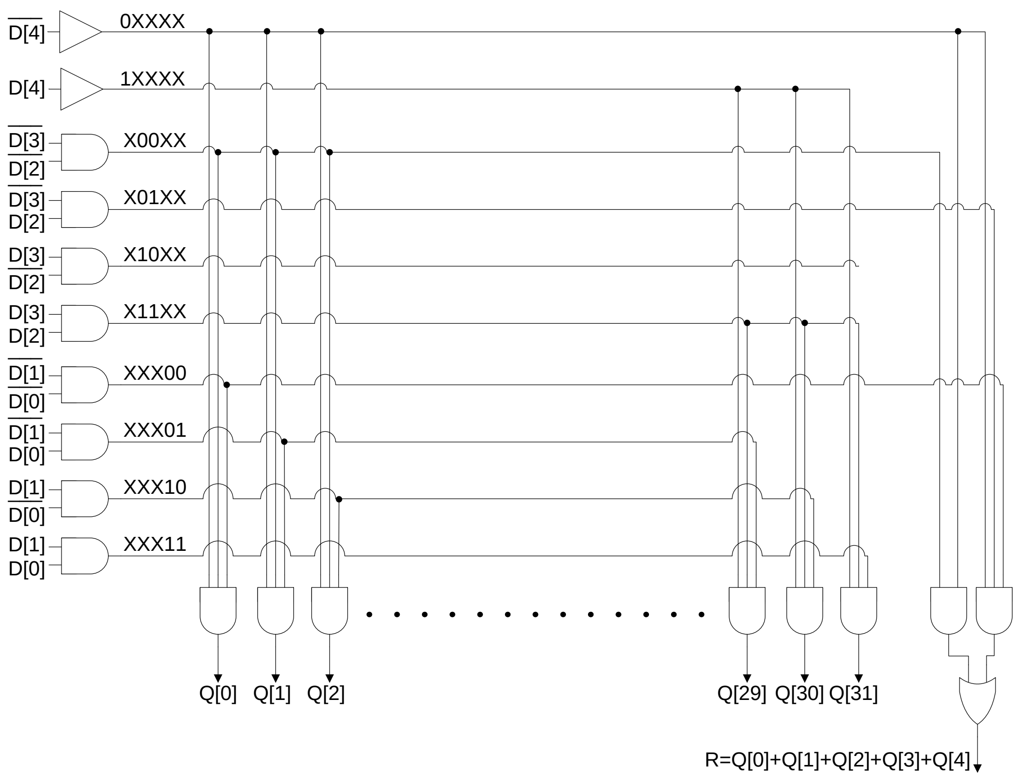


Figure 83. A 5-to-32 decoder

The last component of the state machine is the 5-to-32 decoder, Figure 83, and decodes a 5-bit binary encoding scheme to a 32-bit one-hot encoding scheme. The decoder first decodes the 5-bit input into 10 different number sequences as indicated on the left side of Figure 83, where 0/1/X represents 0/1/"don't care" respectively. The one-hot encoding output is then decoded using 3-input AND gate using 3 of the 10 different number sequences. This method reduces each column to contain only a 3-input AND gate instead of a 5-input logic decoder if a direct 5-to-32 decoding scheme is used. The last output decoding logic R is used to place a '0' on the SUB bus of Figure 87 for the first five columns readout as described later.

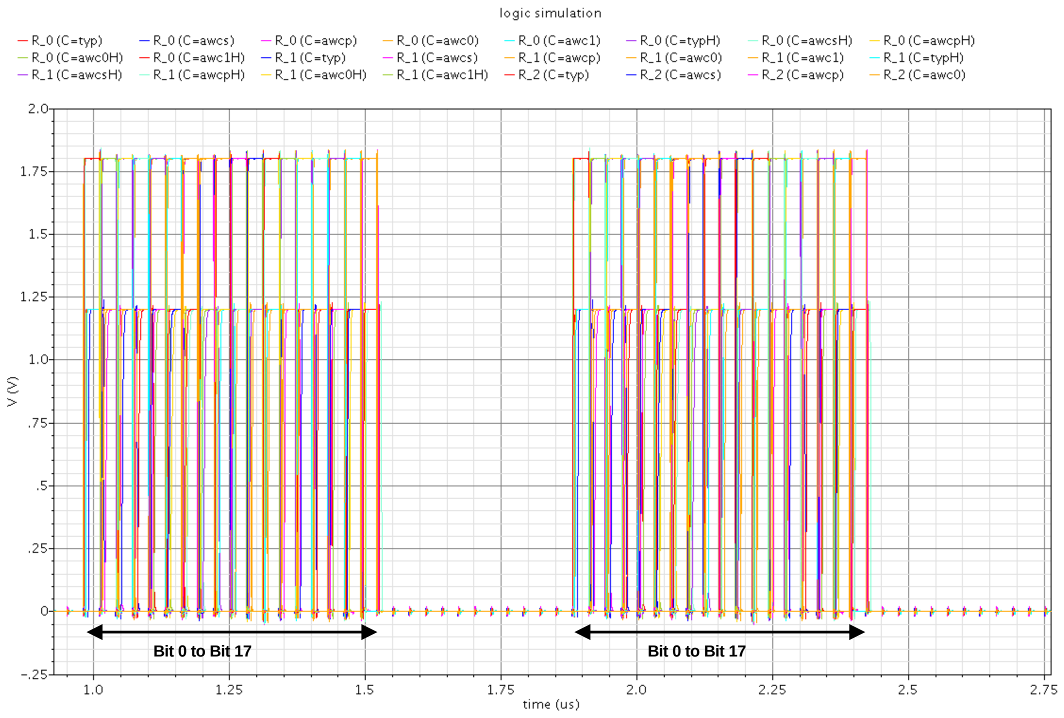


Figure 84. Post-layout simulation of C30

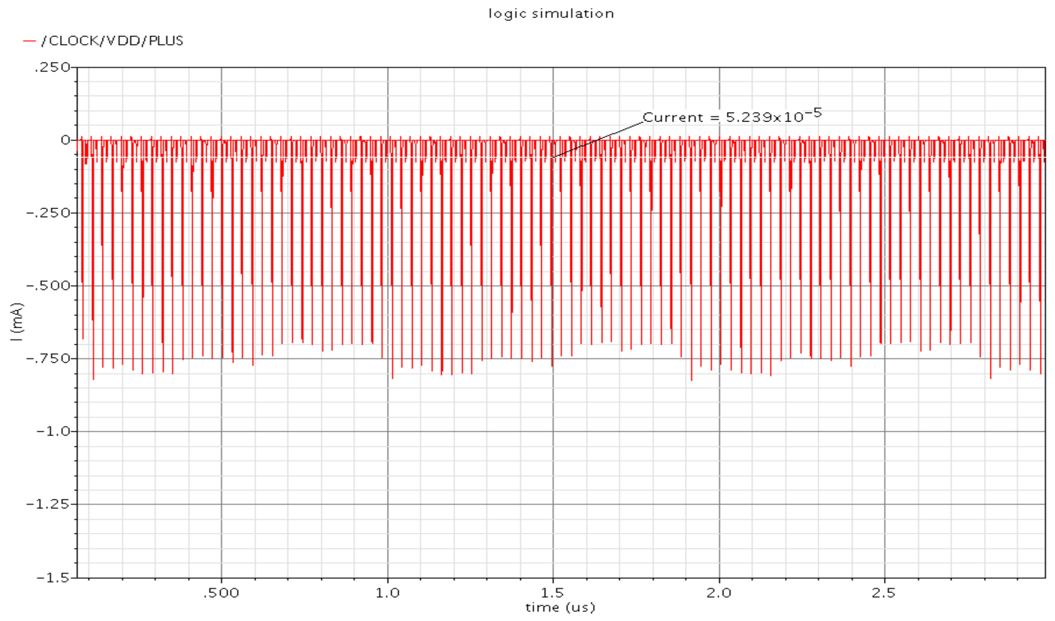


Figure 85. Current consumption of C30

Simulation results of C30, Figure 84, indicates a bit shifting of 0 to 17 and is repeated. Other bits that are not required in the controlling are left out of the waveform result. It is able to perform the role shift register using a pulse-latch clocking methodology and provide enough driving capability across corner variation to control the rest of the arithmetic block. The current consumption is approximately 52.4µA, Figure 85. It is observed that a large amount of current is consumed by the buffering of the control signals.

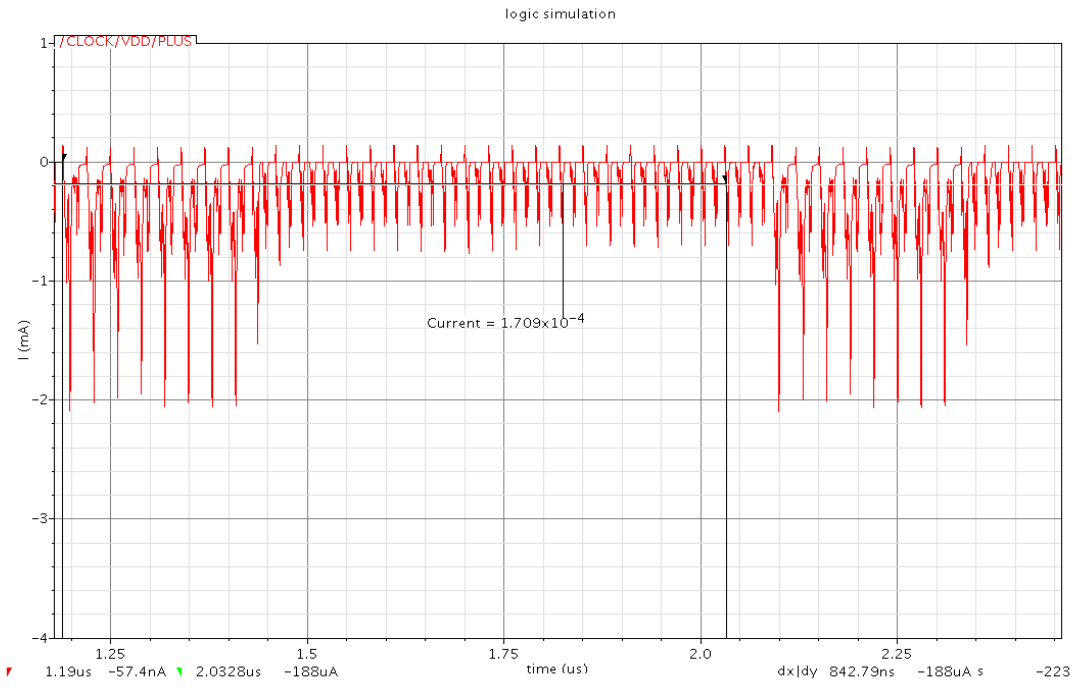


Figure 86. Current consumption of C30+CG0+CR9+CR64+DEC+SRAM

As mentioned earlier, the state machine is highly integrated with the memory block and thus the current consumption is simulated jointly as a unit. The total current consumption drawn by C30+CG0+CR9+CR64+DEC+SRAM is approximately 170.9µA, Figure 86.

Memory Interface

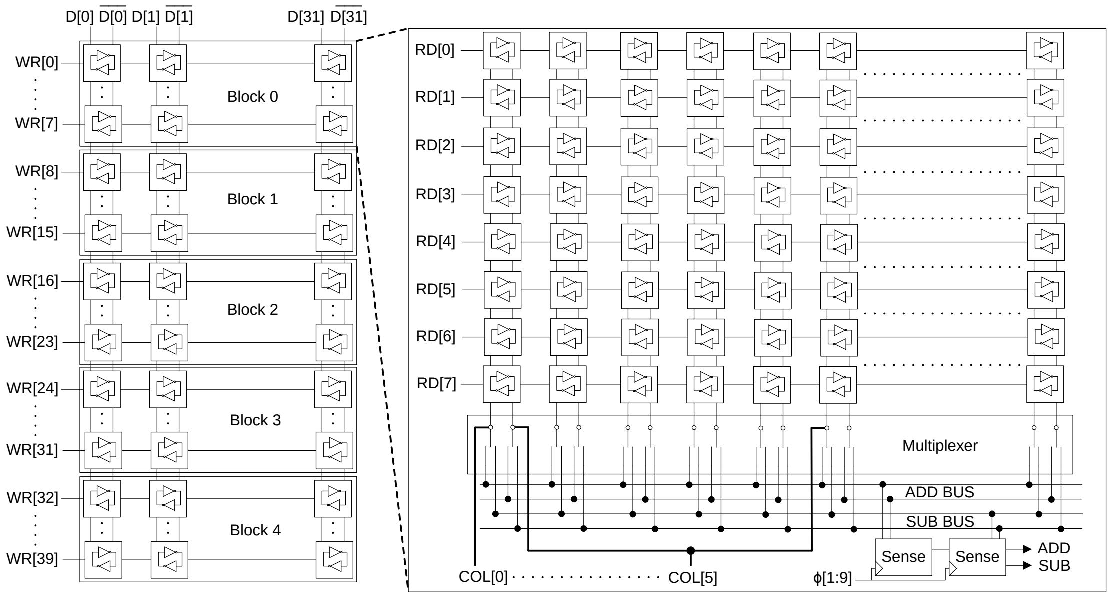


Figure 87. Arrangement of SRAM

The memories are arranged in 32 columns × 40 rows as illustrated on the left of Figure 87 and provide an interface for data communication between the sensor and the processing unit. It can also be viewed as 5 blocks of memory drawn in Figure 87 where each block corresponds to a row of pixel data (8-bits) in Figure 42 and each column corresponds to a column of pixel data in Figure 42. Due to the rolling readout of the sensor, the sensor controller can write its output digital data in parallel to a different block of memory every time a row of A2D conversion is completed. This operation enables the memory to retain the last five rows of pixel data at any time to perform a windowing operation. The sequence of writing data into the DSP can be as follows:

1. ADC is completed and data are written into block 0.
2. ADC is completed and data are written into block 1.
3. ADC is completed and data are written into block 2.
4. ADC is completed and data are written into block 3.

5. ADC is completed and data are written into block 4.
6. The processing of the last five rows of pixel data is triggered.
7. ADC is completed and data are over-written in block 0.
8. The processing of the last five rows of pixel data is triggered.
9. Step 7 and 8 is repeated by writing into a different memory block until all the rows on the sensor have been converted and written.

This arrangement will make the DSP to be compatible to any form of column-parallel readouts whether it is ASP sensor or DPS sensor, and thus isolates itself from any incompatibility. The input signals for writing are arranged such that the sensor is able to write independent into each row of the memory using the differential inputs signal. Thus, the input buses along each column are connected together while the write signals are separately joined along the row. Similar to 6T SRAM, data is written to the memory by placing a differential input on D and ~D and triggering a '1' on WR.

The readout architecture is slightly more complex to achieve a bit-serial readout directly instead of word-parallel readout followed by a bit-serial conversion using shift registers. This design eliminates the need for shift registers such as those found in the distributed arithmetic unit of Figure 35 from [83]. Also, all the pixel data in both add and subtract columns of Figure 42 are required to be accessed in parallel bit-serially. The internal connections of the memory block are illustrated on the right of Figure 87 to achieve a bit-serial readout directly. In each block, the read ports are joined together and connected to a multi-input single-pass-transistor switched multiplexer. A column controller then selects the multiplexer to connect one of the column buses to the ADD bus and another to the SUB bus. On the first five column reads, the SUB bus is set to zero to accumulate ΣI and ΣI^2 initially. On the sixth column reads, the first column is placed on the SUB bus. Subsequently, the readouts are swept from the left to the right. Individual bits can then be readout by setting a '1' on the RD control. A differential output is then sensed and transferred to the arithmetic processing elements. Each block outputs a single ADD and SUB signal which totals to five pairs of ADD and SUB signals as required.

No. of Logic Transitions	*Count*
0	2
1	14
2	42
3	70
4	70
5	42
6	14
7	2

Table 8. Number of logic transitions in all 8-bit digits

Previously in [68], non pre-charge memory has demonstrated a 74% power reduction based on the fact that the most significant bits of a bit-parallel video data is highly correlated. Similar design of non pre-charge memory is also applicable to a bit-serial readout. For example, a bit-serial readout of binary value 00000000 has zero logic transitions and binary value 10101010 has seven logic transitions. By considering all possible binary combinations, the number of logic transitions in all 8-bit digits can be tabulated. From Table 8, the expected number of logic transitions for random data is 3.5 and thus the use of a non pre-charge SRAM has a theoretical power savings of 50% in a bit-serial readout.

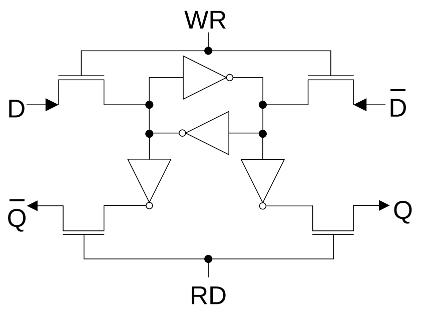


Figure 88. Non pre-charge, differential SRAM

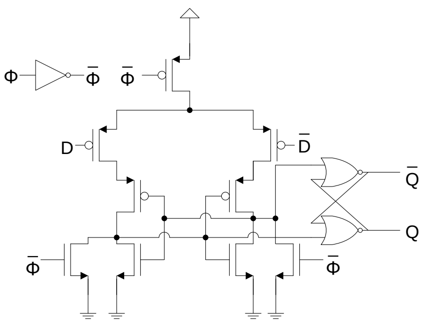


Figure 89. Sense-amplifier flip-flop

As compared to the 90nm and 45nm process in [68] and [69] respectively, the use of an inverter to drive a switch with high bus capacitance at low voltage and high V_{th} is slow and undesirable. Therefore a differential with non pre-charge output SRAM, Figure 88, is used at the cost of more area and additional *Sense-Amplifier-Flip-Flop* (SAFF), Figure 89. Using only a single pass-transistor switch, the differential output is also limited to V_{dd} - $V_{th,nmos}$ and eliminates the need to pre-charge the memory bus to the full supply voltage.

In order for the SAFF to switch without error, a few hundred milli-volts difference between the inputs is required to overcome the offset voltage and noise. This will limit the minimum supply voltage to be around $V_{th,nmos}+V_{difference}$. In the worst case scenario using max $V_{th}\approx 0.8V$ and a voltage difference of 0.4V, this will set the limit of the supply voltage to 1.2V. This voltage difference of 0.4V also provides some headroom for other circuits to operate efficiently throughout the DSP unit. As the SRAM is operating in non pre-charging mode, the voltage difference is not always at 0.4V as it may not be charged or discharged completely within the operating frequency. A worst case input pattern will be seven '1's followed by one '0' or vice versa to discharge a maximum charged memory bus at the last bit and it can be used to estimate the worst voltage difference.

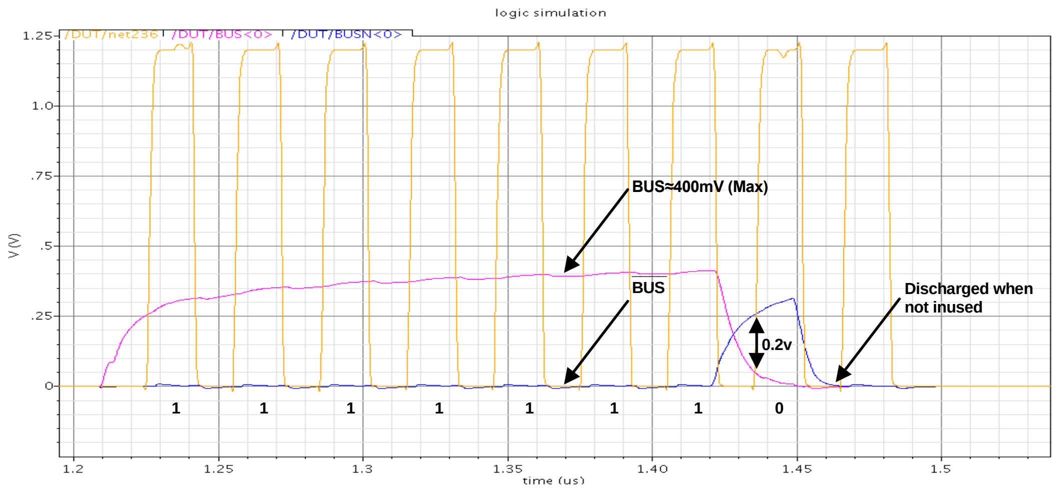


Figure 90. Worst case voltage difference on memory bus at 30MHz

Since the worst possible scenario can only happen with slow transistors (maximum V_{th}), a post-layout simulation of the memory block is only performed with the worst corner variation, Figure 90. Simulated result is as expected with an approximate maximum voltage difference of 400mV and an approximate minimum voltage difference is found to be 200mV. The sampled output of the above simulation with its output inverted is shown in Figure 93. A monte-carlo simulation of 1000 samples, Figure 94, are collected with an input voltage difference of 200mV at worst process corner to ensure that SAFF switched correctly in the presence of offset voltage at process mismatch condition.

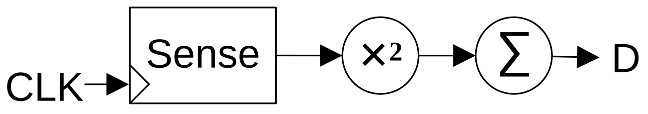


Figure 91. Critical path from memory block to arithmetic block

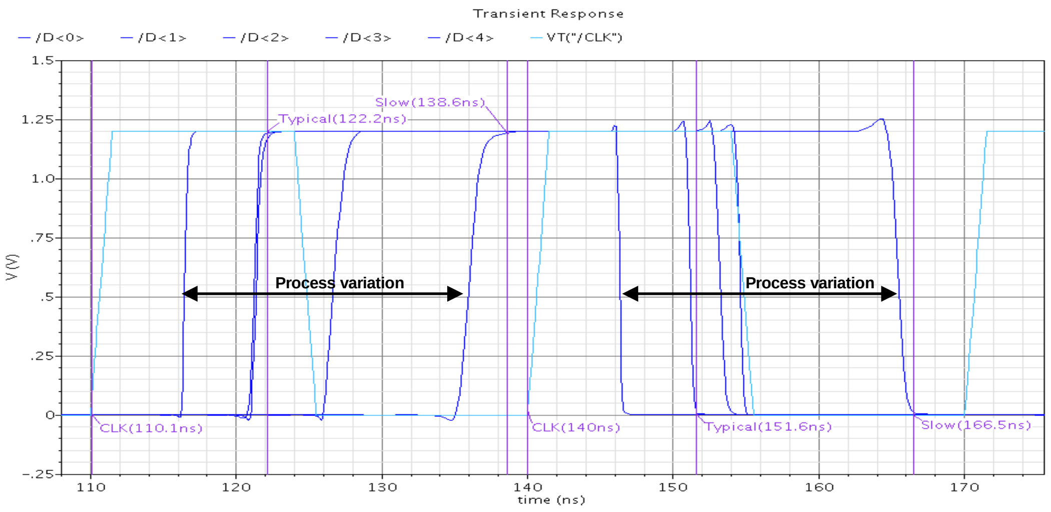


Figure 92. Critical path delay from memory block to arithmetic block at 1.2v

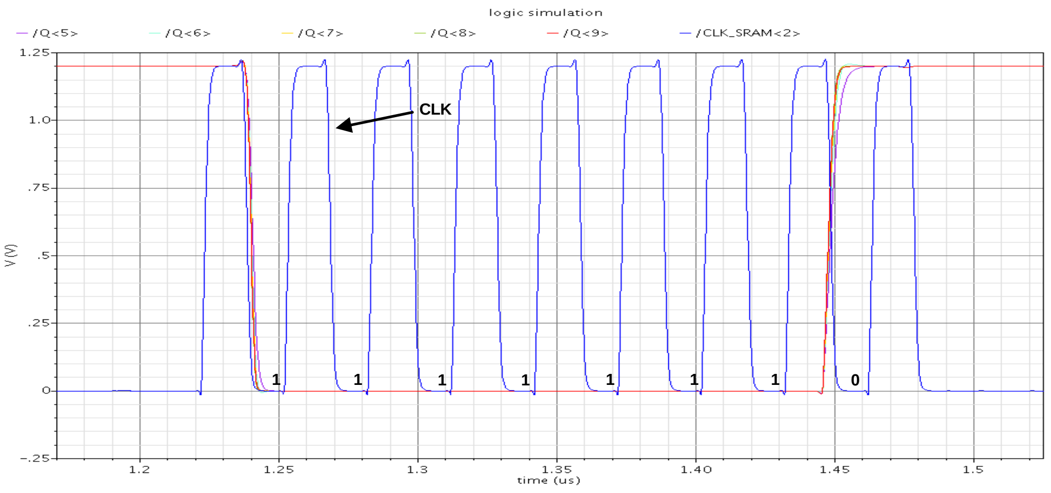


Figure 93. Inverted output of memory block for ‘01111111’ (LSBF)

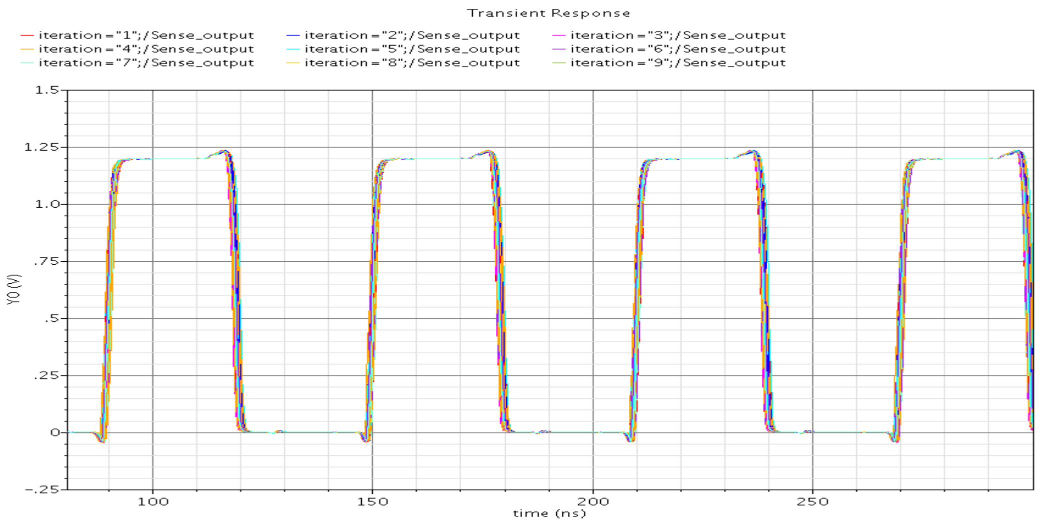


Figure 94. Monte-carlo simulation of 1000 samples of SAFF

The SAFF, Figure 89, is referenced from [91] using a first stage pre-charge sense amplifier followed with a set-reset latch. The slow performance of the set-reset latch degrades the delay path but is absorbed by the fast processing signal path of the BS-arithmetic nodes. The critical path of Figure 44 from the memory block is redrawn in Figure 91 and a post-layout simulation measuring from input CLK to output D is shown in Figure 92. This critical path delay measures about 28.5ns at worst process corner, 12.2ns at typical process corner and fits nicely into a 30MHz operating frequency even at worst process corner. The functional simulations of the memory blocks and SAFF are shown in Figure 93 and Figure 94 respectively.

Clocking Strategy

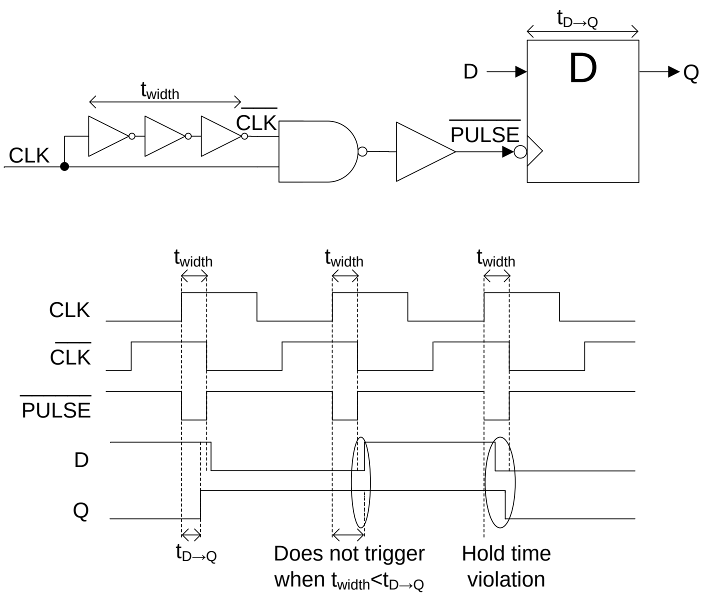


Figure 95. Inverted pulse generator and its hazard.

Inverted pulse generators, Figure 95, have been inserted into the clock network to achieve a pulse-latch clock strategy in this work. A buffer is placed after the NAND gate to enable more latches to share a single pulse-generator. Consider a simple level-triggered latch used with the inverted pulse generator, the inherent timing requirement will be t_{width} >

$t_{D\rightarrow Q}$, such that the input D is able propagate to output Q. Also, the transparent latch requires a hold time of t_{width} to ensure that D is stable while propagating to Q. To meet the timing requirements, t_{width} is adjusted by changing the sizes and stages of the inverter from simulation such that the design latches meet the inherent timing requirement.

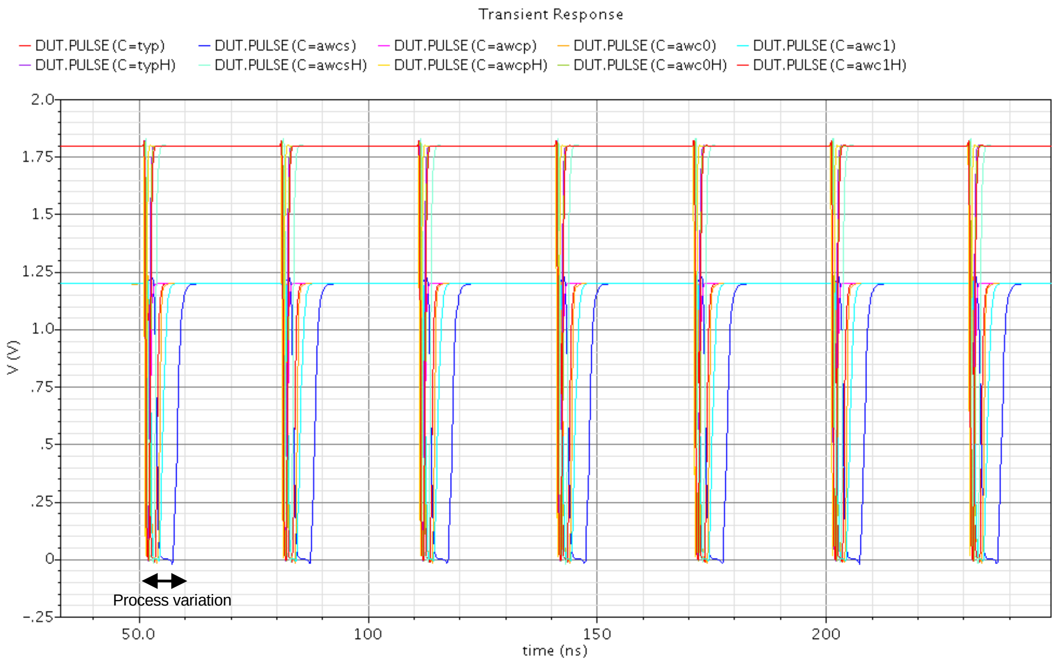


Figure 96. Post-layout simulation of inverted pulse generator.

After performing a post-layout simulation of the pulse-generator in Figure 96, a large variation of the pulse width is found against process corner variation, especially at low voltage. Instead of performing a detailed hold time analysis with a single hold time, a better approach is to ensure that the number of gate delay from latch-to-latch to be greater than the number of stages of inverter in the pulse-generator. This will in fact track the required delay of the pulse-generator. A rule of thumb in this design is to increase the gate delay by one to cater for clock skew. Subsequently, a functional simulation with timing analysis can be performed to avoid timing failure.

To facilitate the use of the pulse generator, compatible custom designed latches are required. Over the years, many latch designs have existed and have been used in many applications such as [65], [66], [72], and [90]. One of the advantages of using self-design latches is the ability to embed complex static logic, such as incorporating enable function [66], and [72] and multiplexing function [73]. Since the latches are aligned at every clock edge, dynamic logic can be embedded without the need of power hungry self-timed clocks and clock-trees. The use of dynamic logic is demonstrated in the multiplexing latch, Figure 100. Also, to make the design latches more operable and noise tolerable at low operating voltage, design considerations are made according to [79].

A summary of design strategies is as follows:

- Balancing input clock load for different types of latch by employing similar latch structure to reduce clock skew.
- Transistor sizing are chosen to meet the inherent timing requirement of $t_{width} > t_{D\rightarrow Q}$ across process corners and to provide enough driving capability
- Avoiding stacking of PMOS along signal path to improve the performance at low voltage.
- Ensuring that "A dynamic node is not allowed to drive another dynamic node" by either making the dynamic node static or enhancing its noise immunity using weak feedback [79].
- Use a tri-state feedback instead of weak feedback to avoid the need for strong forward drive and unnecessary contention power.

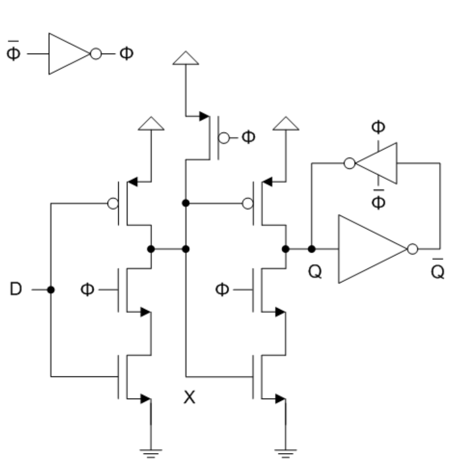


Figure 97. Latch with internal pre-charge

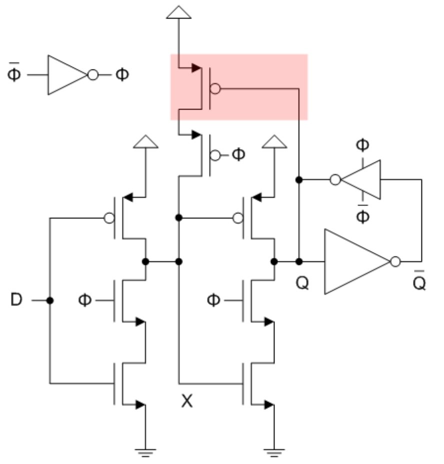


Figure 98. Latch with a tri-state feedback

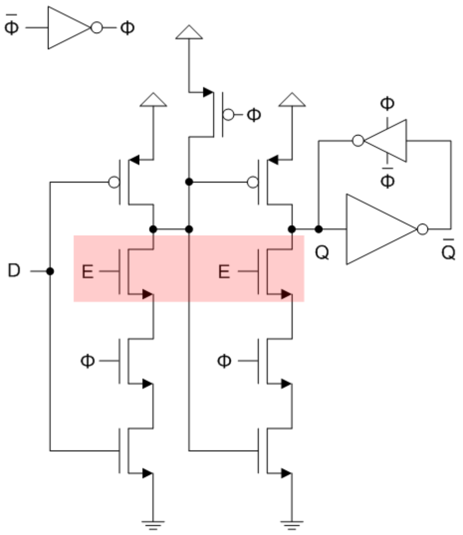


Figure 99. Latch with enable

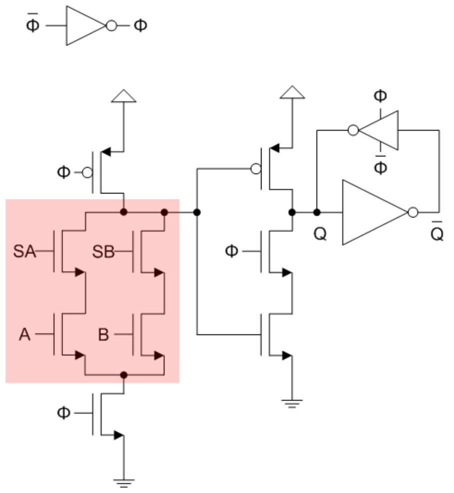


Figure 100. Latch with multiplex input

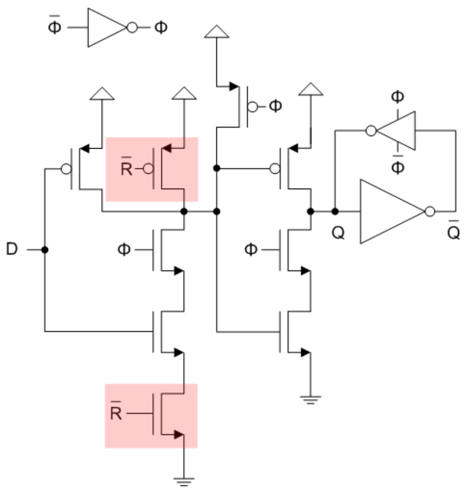


Figure 101. Latch with reset

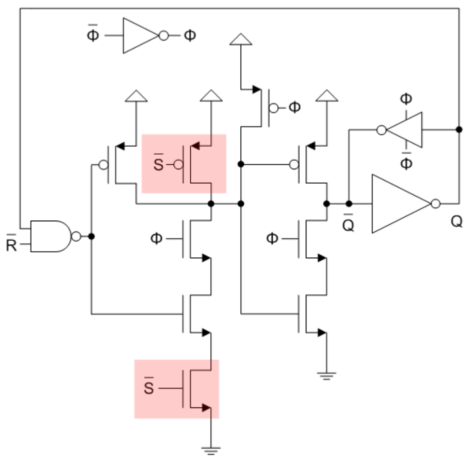


Figure 102. Latch with set and reset

By ensuring the dynamic nodes in the latch to be static, these nodes will be fully charged even just after power up or disabled for a long period. Therefore, any static cells cascaded succeeding the node will never be partially turned on, hence preventing a short circuit from occurring. Although a fully static latch will consume more power, for example the latch in Figure 97 has to charge the internal node X at every clock cycle as compared to Figure 98, one can always design the logic such that this pre-charge power consumption can be minimised by theoretical observation. For example, if a signal '0' has a high probability of propagating across the latch in Figure 97, the internal node X will likely to be always charged. Also, if one is required to hold a data for multiple clock cycles using Figure 99, then the internal node will likely to be always pre-charged as well.

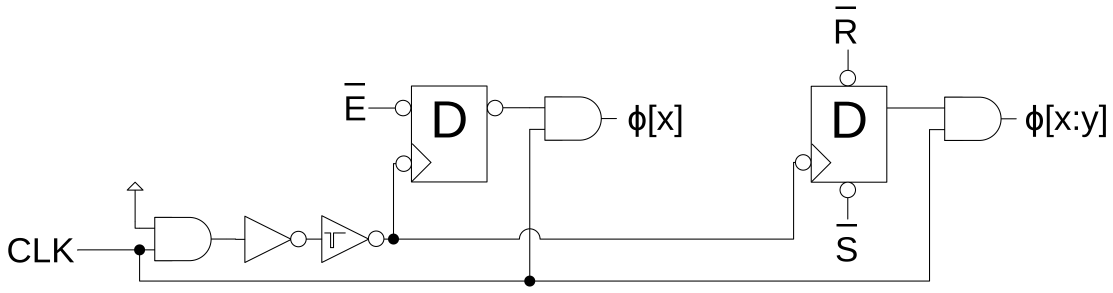


Figure 103. Pulse-latch clock gating

Although employing a latch with enable, Figure 99, already eliminates the need of multiplexers for conventional memory units, one can still enjoy the additional dynamic power savings along the clock path if clock gating is applied. In addition, these latches also enjoy the power savings in places where grouping of memorizing units are not possible for clock gating. Without clock-gating cells in the standard library, two types of pulse-latched clock-gating cells, Figure 103, have been used to generate the gated-clock signals that are required in the architecture. The left and right latches are Figure 98 and Figure 102 respectively, where the clock input signal is delayed and inverted before being fedto the pulse generator. This delay ensures that the control signals, generated synchronously from the positive-edge triggered logic circuits, are able to reach the negative-edge triggered (CLK) clock-gating cells. The hold time requirement can be ignored in the clock-gating cells as the control signals will only switch at a positive-edge. The left latch is used to generate a single cycle clock signal where ϕ[x] denotes a single clock at the x-th cycle of the 30-cycle pipeline execution while the right latch is used to generate a range of clock signals where ϕ[x:y] denotes a clock signal spanning from x-th to the y-th cycle of the 30-cycle pipeline execution.

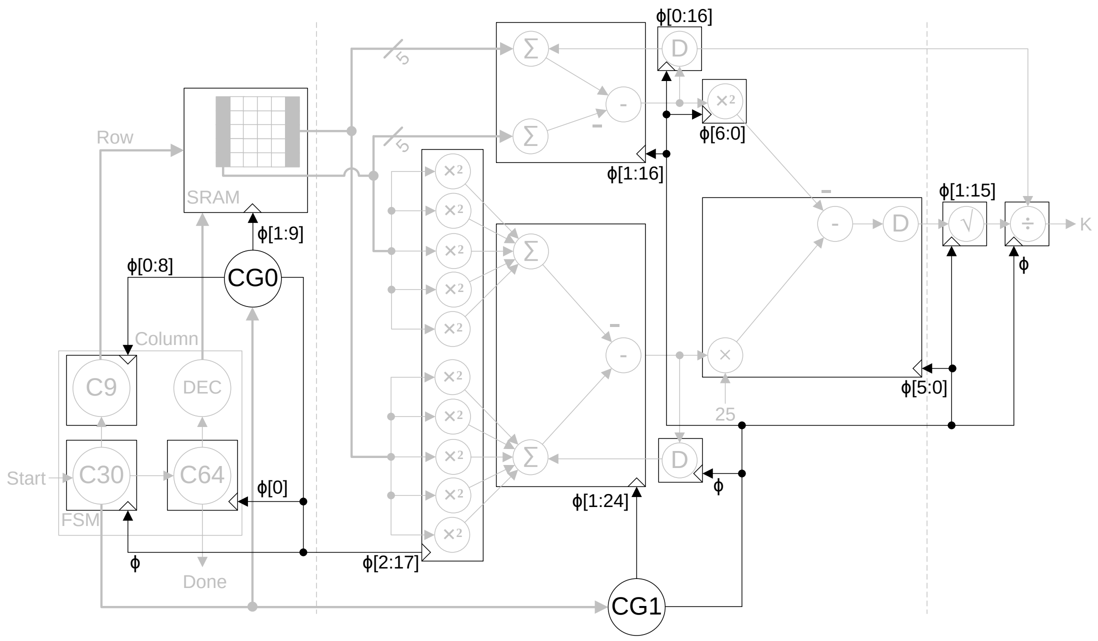


Figure 104. Clock gating signals

In a bit-serial architecture, arithmetic blocks can be turned off once the expected number of output bits is rolled out. In some scenarios, one additional cycle is required before the signal enters the block to perform loading/resetting and one additional cycle is required after the output signal to zero/sign-extend the expected output to the next bit-serial arithmetic blocks. On occasion where the arithmetic blocks contain too little latches to be clock-gated, they can be grouped with the neighbouring blocks to form an isolated clock-gating group. Taking into considerations of all the blocks, the required clock signals of each individual group have been drawn in Figure 104. Due to physical proximity, the clock-gating block is divided into two blocks, CG0 and CG1. This is purely a convenience decision to perform simulation and not a design rule.

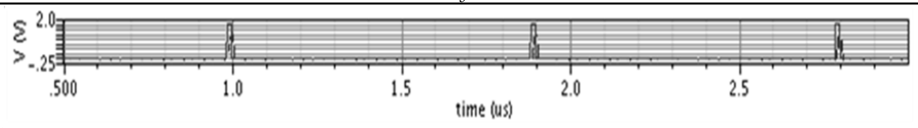
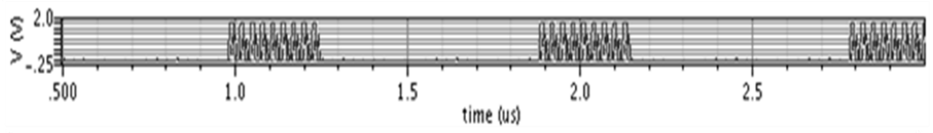
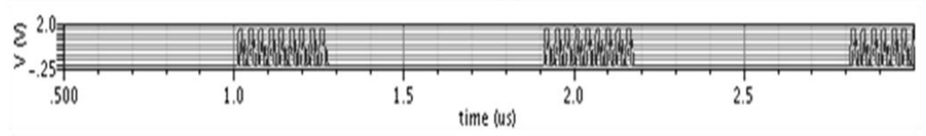
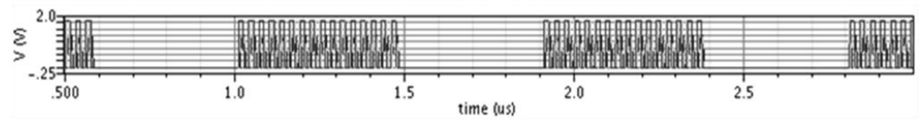
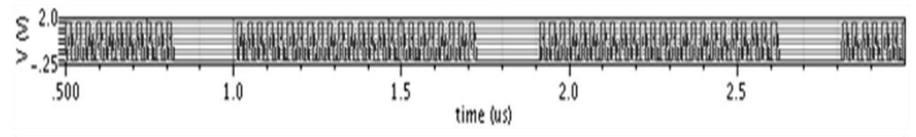
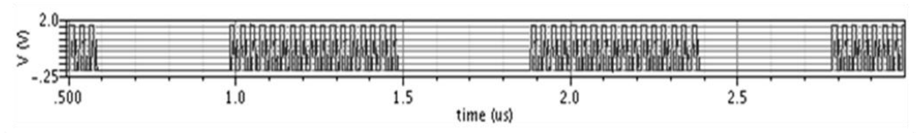
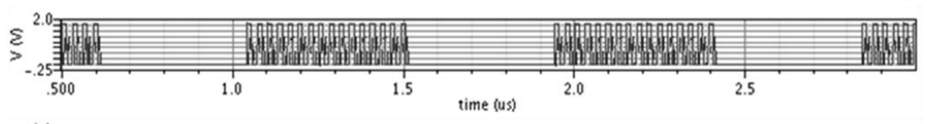
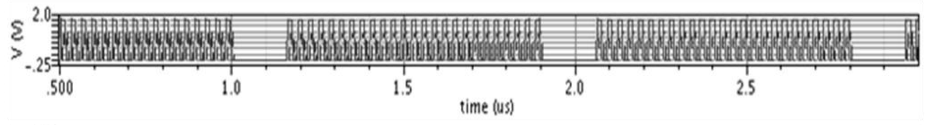
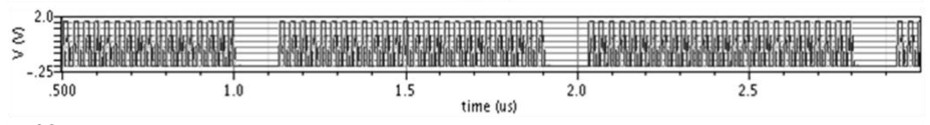
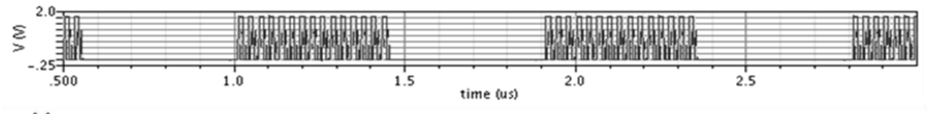
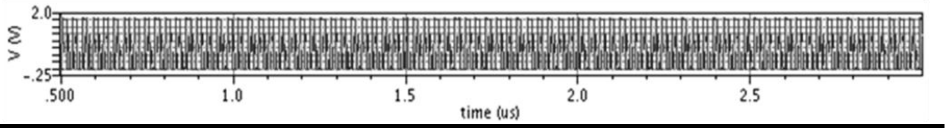
Unit	*Clock*	*Waveform*
CR64	$\phi[0]$	
CR9	$\phi[0:8]$	
SRAM	$\phi[1:9]$	
Σ	$\phi[1:16]$	
Σ	$\phi[1:24]$	
D	$\phi[0:16]$	
x^2	$\phi[2:17]$	
x^2	$\phi[6:0]$	
$\times$	$\phi[5:0]$	
$\surd$	$\phi[1:15]$	
Others	ϕ	

Figure 105. Post-layout simulation of gated-clocks

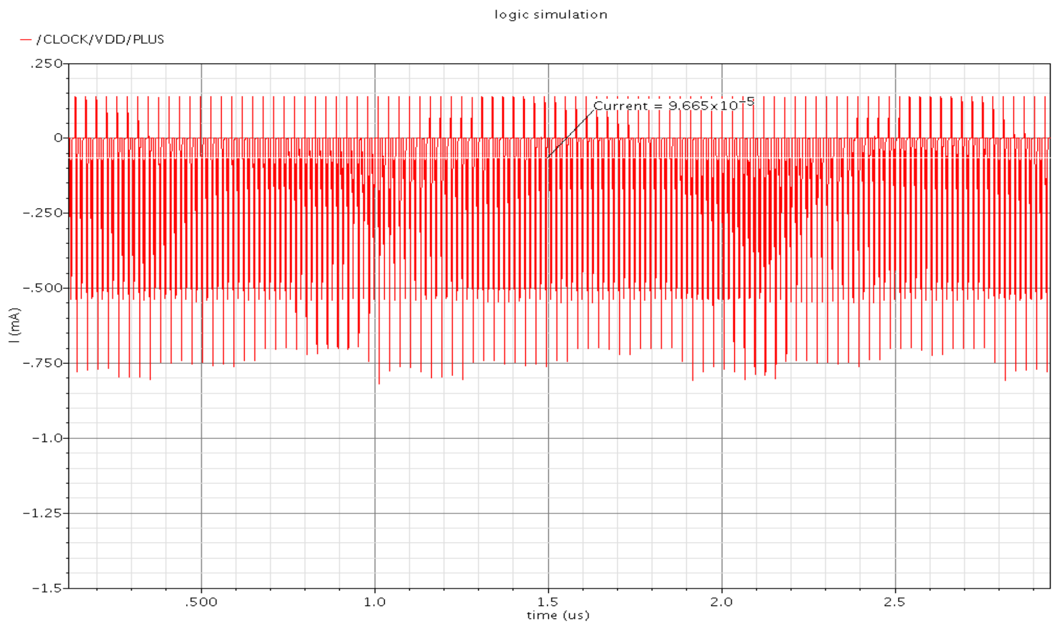


Figure 106. Current consumption of C30+CG0

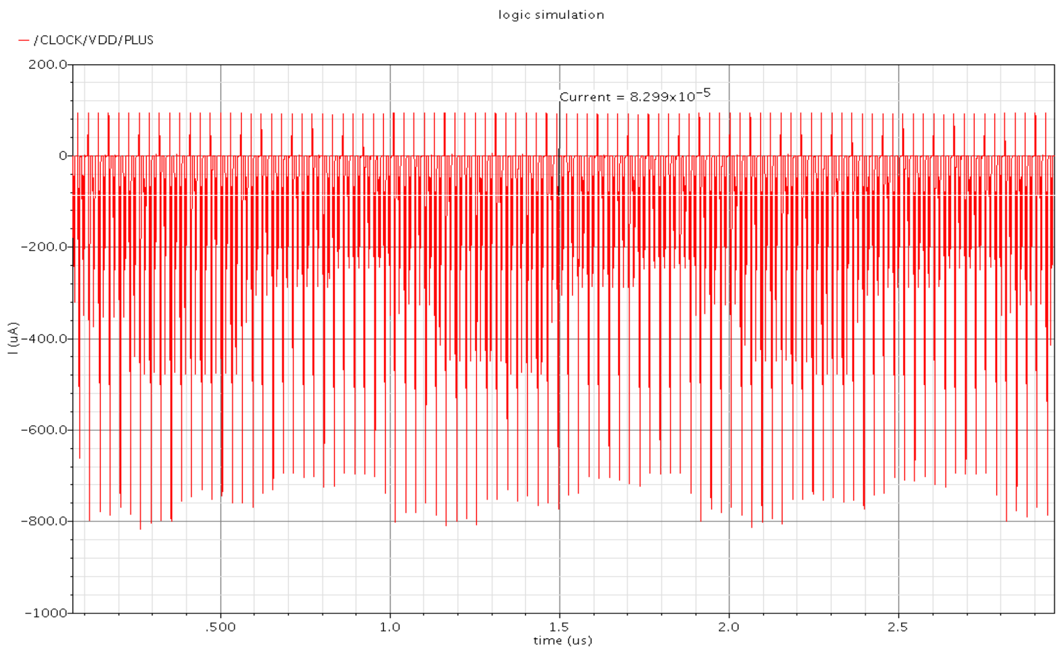


Figure 107. Current consumption of C30+CG1

The post-layout simulated gated-clocks are shown in Figure 105 with the current consumption illustrated in Figure 106 and Figure 107. The correctness of the simulated clock-gating signals in Figure 105 across corner variation also indicates the fulfilment of the timing requirement of the latches. As the control signals are generated from C30, the current consumption in both figures includes the current drawn by C30. The absolute average current drawn by C30 was presented earlier and the current consumption drawn by C30+CG0 and C30+CG1 is approximately 96.7μA and 83.0μA respectively.

Functional Verification

Two raw speckle images are used for functional simulation of the C/C++ and the Verilog models of the design. In both Figure 108 and Figure 109, the raw speckle images are on the left and the simulated outputs are on the right. In Figure 108, the image was generated by right-shifting the output of the DSP unit by 7-bits. In Figure 109, the image was generated by right-shifting the output of the DSP unit by 5-bits. Both models produce the same output and the Verilog code is said to be functional cycle accurate and the DSP is capable of generating more than the required precision in both images. Both simulated speckle contrast images are 8-bit bitmap, using the most precise 8-bits of the DSP output.

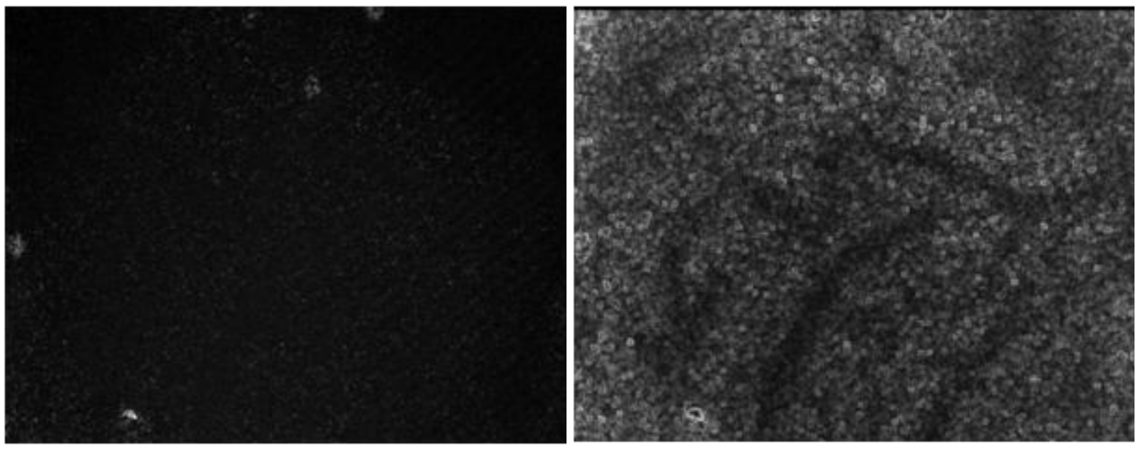


Figure 108. Simulation I - (a) raw speckle image; (b) speckle contrast [1]

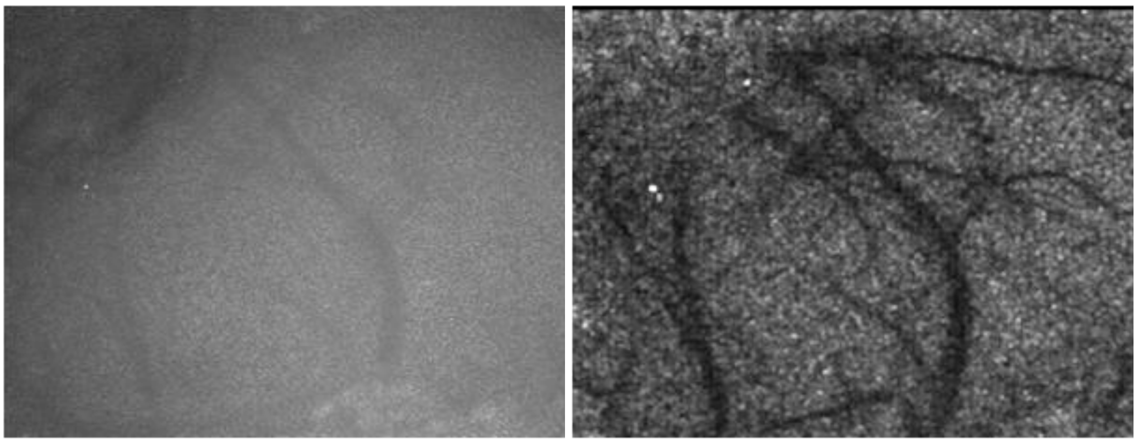


Figure 109. Simulation II - (a) raw speckle image; (b) speckle contrast [94]

This right-shifting operation does not affect the absolute precision of the DSP output and is only a means of extracting the most precise 8-bits. However, the number of right-shifting positions indicates the required amount of precision to be generated. Table 9 tabulates the required minimum and maximum precision.

Description	*Requirement*	*Precision*
Figure 108	Min	Q13.8
Figure 109	Min	Q13.10
DSP output	Max	Q13.15

Table 9. Precision requirement

In both simulations, the useful data lies in the fractional bits and the accuracy of the division is important to generate a reasonable viewing image. Although the DSP generates five more bits of precision than the minimum requirement, this is not a sufficient condition to generalise that all images required such amount of precision and the required precision should not be set based on these two images.

CHAPTER VII

CONCLUSION

This chapter concludes the thesis with a summary of the design work, an assessment of strength and weakness, and considerations for future works.

Design Summary

The simulated average current consumed at typical process totals up to 739µA which accounts for a power consumption of 887µW operating at 1.2V and 30MHz. The current distribution of the system is shown in Figure 110. It is observed that little power is consumed at the input clock buffer as most of the clock trees have been shifted into the clock-gating logic circuits.

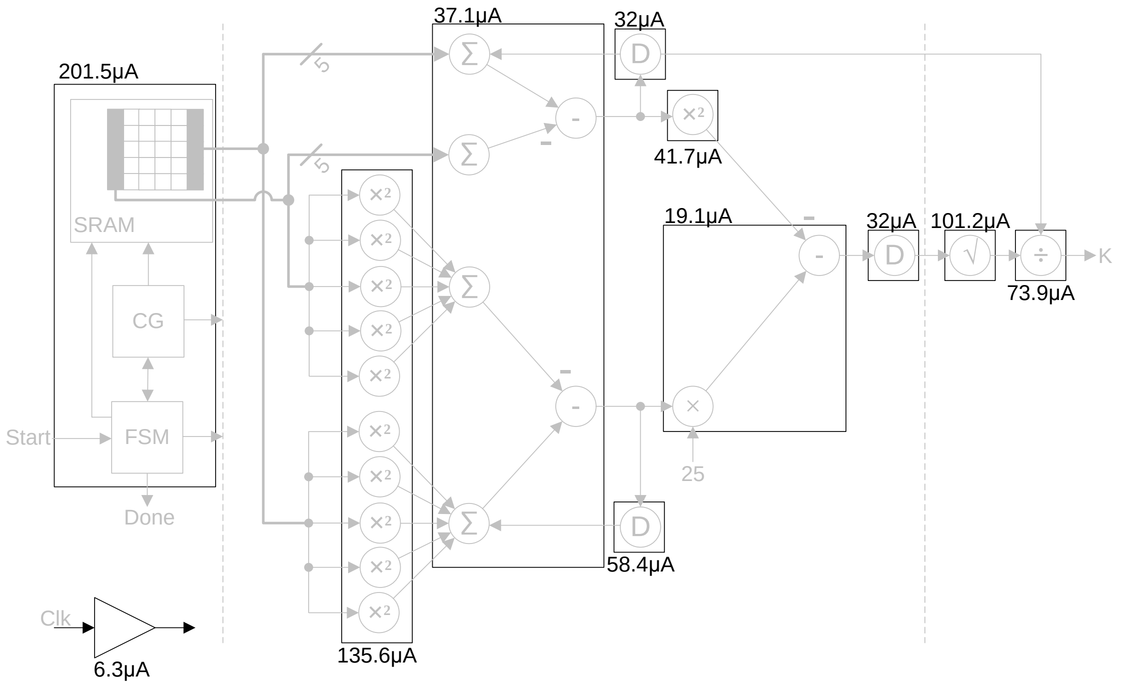


Figure 110. Current consumption distribution

Table 10 reveals some of the existing low-power digital circuits for different applications and is shown as an informative reference. Note that it is difficult to draw conclusions due to the different complexity involved in each algorithm.

Description	*Process*	*Supply*	*Clock*	*Resolution*	*Fps*	*Power*	*FOM*	*Transistor*
MPEG-4 video decoder LSI [67]	0.18μm	1.5v	27MHz	176×144	15	8.5mW	22.4nW/fp	11M
JPEG-LS for endoscopic capsule [95]	0.18μm	1.8v	40MHz	320×288	8	6.2mW	8.4nW/fp	70.4k
DCT processor with variable Vth [96]	0.3μm	0.9v	150MHz	≈150M	1	10mW	66pW/fp	120k
Energy harvesting heartbeat DSP [83]	0.6 μm	1.5v	1.2kHz	1-D samples	(*)160	560nW	(**)3.5nW/fp	190k
Single unit in this work	0.35μm	1.2v	30MHz	≈1M	1	887μW	962pW/fp	36k

Table 10. Performance comparison
(**) Re-calculated based on power per (*) sampling rate

Description	*Specification*
Supply voltage	1.2v – 1.8v
Frequency	30 MHz @ ($\|V_{tn}\|+\|V_{tp}\|$=1.55v)
Layout	560μm × 1300μm
Transistors	≈36k
Window size	5×5 (*N*=25)
Pipeline	2 stages
Latency	60 cycles
Throughput	1/30 speckle/cycles
Input precision	8-bit
Output precision	Q13.15
Power	887μW @ 1.2v,30MHz,typical
Rate	≈1 million pixels per second

Table 11. Simulated specification of a single unit

While it may not be the most energy-efficient application in Table 10, it does achieve a low-power design of 887μW in simulation, Table 11. The main contributing factor of low-power design is due to the aggressive lowering of supply voltage. However, as the supply voltage goes below $|V_{th}|+|V_{tn}|$, process corner variation starts to widen and it is difficult to maintain high clock frequency. A large amount of time is spent on performing corner simulation to ensure that the dynamic circuits and pulse-latches are operable within the specifications. Significant power savings are also observed from clock-gating of the BS-squarer and square-root unit. For an average resolution used in JPEG-LS from Table 10, a single unit of this design is sufficient to operate within its frame-rate. The low-power consumption also makes it easier to be duplicated in a column-parallel architecture.

In this work, higher performance is maintained through the use of bit-serial arithmetic units and these units include adder, multiplier, squarer, square-root and divider. This design is implemented in 0.35μm and a post-layout simulated power consumption of 887μW is achieved at a supply voltage of 1.2V while maintaining 30MHz at worst corner variation. This translates to approximately 1 million speckle contrast computations per second and a FOM of 962pW/fp. Although it is not as energy-efficient as [96], leakage problem can be avoided. Besides, lowering threshold voltage in [96] can also be easily done through better technology and might not be considered in future applications. More work is also required to customise the standard library [96] as the source of the transistors cannot be connected to the body. The use of narrow bit-width adders through bit-serial circuits is the most critical factor that limits the operating frequency of the DSP unit. Although the bit-parallel adders in the design may not be fastest adder in literature, it does achieve its purpose by using static CMOS circuits.

Figure 111 shows the custom top-level layout and the location of the arithmetic blocks and the empty spaces are filled with decoupling capacitors to reduce switching noise. Horizontal and internal cell are routed using metal 1. Metal 2 is used mainly for vertical routings and vertical power strips and the top metal 3 is used only in areas when routing is unachievable. However, metal 3 is also used to route the input horizontal write signals in the SRAM. Besides, an asynchronous reset signal for the SRAM read control signal is provided near the write signal to prevent short circuit current during power up. The input SRAM differential signals are routed using metal 2 vertically from the top and other input control signals are routed to the bottom left. The output signals are routed to the bottom right. Additional unit can be placed beside and connected horizontally to share the write and asynchronous reset signal to form a column-parallel architecture if required.

The total transistors count approximates to 36K including SRAM. This is much smaller than the estimates of 9.4K gates≈37.6K transistors excluding SRAM in Table 4. Note that SRAM accounts for a large transistor count and area from Figure 111. This reduction is mainly due to the latch replacement of master-slave flip-flops and the use of an output bit-serial SRAM instead of shift registers. Such achievement is only possible through the use of custom digital design as compared to synthesize methodology.

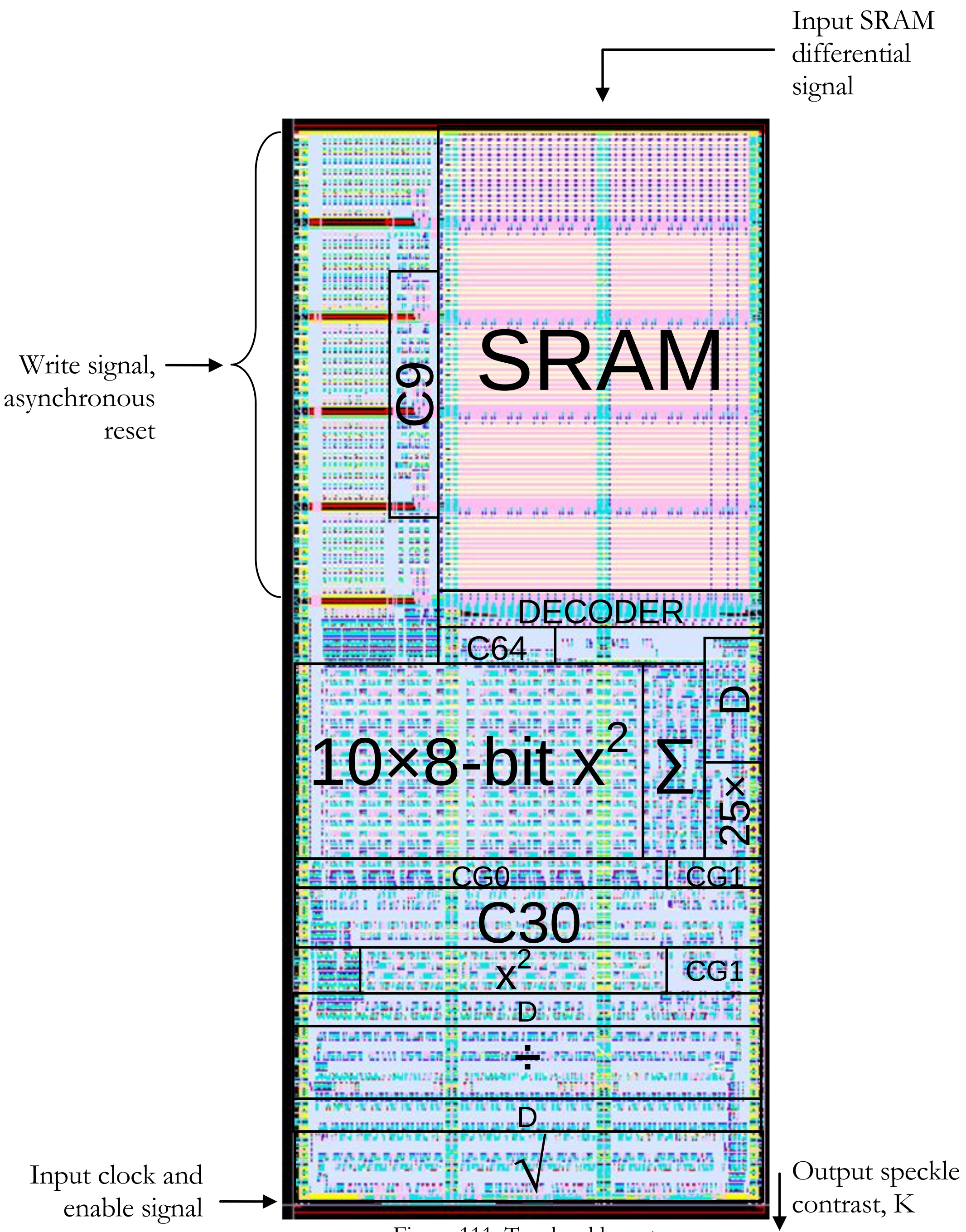

Input SRAM differential signal
Write signal, asynchronous reset
SRAM
C9
DECODER
C64
10×8-bit x²
Σ
D
25×
CG0
CG1
C30
x²
CG1
D
÷
D
√
Input clock and enable signal
Output speckle contrast, K

Figure 111. Top-level layout

Assessment

In this thesis, the first hardware design for cortical blood flow monitoring is presented. A fully custom digital design methodology and the algorithm derivation for an optimised implementation have been outlined. The suitability of the different LSI algorithms has been carefully analysed and all present methods are found to be measuring the same coefficient of variation but not mentioned in previous literature. A single low powered DSP unit measuring this coefficient of variation is achieved and is ready to be integrated with a CMOS image sensor. The use of a memory interface in this design will resolve any incompatibility to future development of CMOS sensor as data can be easily written into the DSP unit with the sensor master controller.

The precision of the generated speckle contrast is argued from a 13-bit division operation and this has resulted in a Q13.15 output. Since the coefficient of variation has not been mentioned in previous literature, it is interesting to note that it has an inherent property of having the range [0, 1] in this application. Being a fraction, the unit can be modified to generate a Q0.28 precision at the expense of more power hungry logic circuits and is not mentioned in this research work.

Bulk of the research work lies in custom digital design and it is extremely time consuming and effort driven. Although this has resulted in a much lower transistor count when compared to an automatic synthesis process, such methodologies are not suitable for actual fast turnover implementation.

Being the first design in literature, this work has been accepted for presentation at International Symposium on VLSI Design, Automation and Test 2009. Due to the busy work schedule, no one was able to attend for presentation.

Future Works

Possible research directions include the following:

1. The foremost direction is to integrate the design with a CMOS sensor so that fabricating and testing is achievable. This includes researching into low power CMOS sensors working at low supply voltages where techniques are mentioned in literature but not available.
2. When a single unit is tested successfully, more research work can be performed on parallel implementation.
3. Digit-serial and parallel implementation can also be considered in the future.
4. A lot of design time is actually wasted due to the lack of design tools available for the design kit and this has actually resulted in many work arounds in the design flow. A more streamlined design flow integrated with more advanced tools is definitely achievable when these tools are available. This includes researching into automatic placement and routing for custom digital cells and automatic timing closure on custom cells. This will reduce the turnaround time if this work is to be considered for manufacturing as multiple fabrication phases are required for testing.
5. Due to the limited time and design tools, custom digital cells to replace the existing standard library are not considered. Many transmission gate logics with fewer transistor counts have been reported in literature but are not available. Another research area is to create a more optimised digital library to facilitate any form of digital design.

BIBLIOGRAPHY

[1] T.M. Le, J.S. Paul, H. Al-Nashash, A. Tan, A.R. Luft, F.S. Sheu, and S.H. Ong, "New insights into image processing of cortical blood flow monitors using laser speckle imaging," *IEEE Transactions on Medical Imaging*, vol. 26, no. 6, pp. 833–842, June 2007.

[2] AMI Semiconductor Inc, *C035U (0.35μm) core CMOS design rules DES-0005*, Rev. 5, July 2007.

[3] AMI Semiconductor Inc, *C035U (0.35μm) core ESD layout rules manual 07-0104*, Rev. 2, July 2007.

[4] AMI Semiconductor Inc, *I3T25/ C035U specific (0.35μm) design rules 1000115*, Rev. B, May 2007.

[5] C.C. Wang, C.C. Huang, J.S. Liou, Y.J. Ciou, I.Y. Huang, C.P. Li, Y.C. Lee, and W.J. Wu, "A Mini-Invasive Long-Term Bladder Urine Pressure Measurement ASIC and System," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 1, pp. 44-49, Mar. 2008.

[6] X. Tao, K. Chakrabarty, and S. Fei, "Defect-Aware High-Level Synthesis and Module Placement for Microfluidic Biochips," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 2, pp. 50-62, Mar. 2008.

[7] C. Stagni, C. Guiducci, L. Benini, B. Ricco, S. Carrara, B. Samori, C. Paulus, M. Schienle, M. Augustyniak, and R. Thewes, "CMOS DNA Sensor Array With Integrated A/D Conversion Based on Label-Free Capacitance Measurement,' *IEEE Journal of Solid-State Circuits*, vol. 41, (12), pp. 2956-2964, Dec. 2006.

[8] A. El Gamal, and H. Eltoukhy, "CMOS Image sensors," *IEEE Circuits and Devices Magazine*, vol. 21, (3), pp. 6-20, May-June. 2005.

[9] H. Eltoukhy, K. Salama, A. El Gamal, M. Ronaghi, and R. Davis, "A 0.18 μm CMOS 10–6 lux Bioluminescence Detection System-on-Chip," *Proceedings of 2004 IEEE Int. Solid-State Circuits Conference.*, San Francisco, CA, pp.222–223, 2004.

[10] M. Schwarz, R. Hauschild, B.J. Hosticka, J. Huppertz, T. Kneip, S. Kolnsberg, L. Ewe, and K.T. Hoc, "Single-chip CMOS Image Sensors for a Retina Implant System," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 7, pp. 870-877, Jul. 1999.

[11] J.N. Burghartz, T. Engelhardt, H.-G. Graf, C. Harendt, H. Richter, C. Scherjon, and K. Warkentin, "CMOS Imager technologies for biomedical applications," *IEEE International Solid-State Circuits Conference*, pp. 142-143, Feb. 2008.

[12] E.R. Fossum, "CMOS image sensors: electronic camera-on-a-chip," *International Electron Devices Meeting*, pp. 17-25, Dec. 1995.

[13] N. Faramarzpour, M. El-Desouki, M.J. Deen, Q.Y. Fang, S. Shirani, and L.W.C. Liu, "CMOS Imaging for biomedical applications," *IEEE Potentials*, vol. 27, no. 3, pp. 31-36, May-June 2008,

[14] Samsung, "Samsung CIS Roadmap," [Online]. Available: http://image-sensors-world.blogspot.com/2007/ 05/samsung-cis-roadmap.html, May 2007.

[15] L. Albanese, "How to manage a derivative SoC project," *EETimes*, July 2007.

[16] J.D. Briers, "Laser Doppler, speckle, and related techniques for blood perfusion mapping and imaging", *Physiological Measurement*, 22: 35-66, 2001.

[17] J.D. Briers, and S. Webster, "Laser Speckle Contrast Analysis (LASCA): A non-scanning, full-field technique for monitoring capillary blood flow," *Journal of Biomedical Optics*, 1(2):174-179, 1996.

[18] A.K Dunn, H Bolay, M.A Moskowitz and D.A Boas, "Dynamic imaging of cerebral blood flow using laser speckle", *Journal of Cerebral Blood Flow and Metabolism*, 21:195-

201, [Online]. Available: http://www.nmr.mgh.harvard.edu/~adunn/speckle/software/speckle_software.html, 2001.

[19] H. Cheng, Q. Luo, S. Zeng, S. Chen, J. Cen, and H. Gong, "Modified laser speckle imaging method with improved spatial resolution," *Journal of Biomedical Optics*. 8(3): pp.559-564, 2003.

[20] D.E. Clarke, R. Perry, and K. Arora, "Characterization of CMOS IC photodiodes using focused laser sources," *Proceedings of the IEEE Southeastcon '96. 'Bringing Together Education, Science and Technology*, pp. 381-384, April 1996.

[21] X. Zhao, F. Boussaid, and A. Bermak, "Characterization of a 0.18μm CMOS color processing scheme for skin detection," *IEEE Sensors Journal*, vol. 7, no. 11, pp. 1471-1474, 2007.

[22] R.B. Merrill, "Color separation in an active pixel cell imaging array using a triple-well structure," U.S. Patent 5,965,875, Oct. 1999.

[23] A. Theuwissen, "CMOS image sensors: State-of-the-art and future perspectives," *IEEE European Solid State Circuits Conference*, pp. 21-27, Sept. 2007.

[24] A.El. Gamal, "Trends in CMOS image sensor technology and design," *IEEE International Electron Devices Meeting*, pp. 805-808, 2002.

[25] S. Kleinfelder, S.H. Lim, X.Q. Liu, and A.El. Gamal, "A 10000 frames/s CMOS digital pixel sensor," *IEEE Journal of Solid-State Circuits*, pp. 2049-2059, Dec. 2001.

[26] M. Kasano, Y. Inaba, M. Mori, S. Kasuga, T. Murata, and T. Yamaguchi, "A 2μm pixel pitch MOS image sensor with an amorphous Si film color filter," *IEEE International Solid-State Circuits Conference*, vol. 1, pp. 611-617, Feb. 2005.

[27] C.C. Cheng, C.H. Lin, C.T. Lim, C.J. Hsu, and L.G. Chen, "iVisual: An intelligent visual sensor SoC with 2790fps CMOS image sensor and 205GOPS/W vision processor," *IEEE Internaional Solid-State Circuits Conference*, pp. 306-307, Feb. 2008.

[28] Y. Zheng, V. Gruev, and J.V. der Spiegel, "Current-mode image sensor with 1.5 transistors per pixel and improved dynamic range," *IEEE International Symposium on Circuits and Systems*, pp. 1850-1853, May 2008.

[29] Y. Zheng, V. Gruev, and J.V. der Spiegel, "A CMOS linear voltage/current dual-mode imager," *IEEE International Symposium on Circuits and Systems*, pp. 3574-3577, 2006.

[30] R.M. Philipp, and R. Etienne-Cummings, "A 1V current-mode CMOS active pixel sensor," *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 4771-4774, May 2005.

[31] R.M. Philipp, D. Orr, V, Gruev, J,V, der Spiegel, and R. Etienne-Cummings, "Linear current-mode active pixel sensor," *IEEE Journal of Solid State Circuits*, pp. 2482-2491, Nov. 2007.

[32] S. Kawahito, "Signal processing architectures for low-noise high resolution CMOS image sensors," *IEEE Custom Integrated Circuits Conference*, pp. 695-702, Sept. 2007.

[33] M.F. Snoeij, A.J.P. Theuwissen, J.H. Huijsing, and K.A.A. Makinwa, "Multiple-ramp column-parallel ADC architectures for CMOS image sensors," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, Dec. 2007.

[34] A.I. Krymski, and N.R. Tu, "A 9V/luxs 5000 frame/s,512×512 CMOS sensor," *IEEE Transactions on Electron Devices*, pp. 136-143, Jan. 2003.

[35] M. Furuta, Y. Nishikawa, T. Inoue, and S. Kawahito, "A high-speed, high-sensitivity digital CMOS image sensor with a global shutter and 12-bit column-parallel cyclic A/D converters," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 4, pp. 766-774, Apr. 2007.

[36] J. Nakamura, B. Pain, T. Nomoto, T. Nakamura, and E.R. Fossum, "On-focal-plane signal processing for current-mode active pixel sensors," *IEEE Transactions on Electron Devices*, vol. 44, no. 10, Oct. 1997.

[37] K. Kagawa, S. Shishido, M. Nunoshita, and J. Ohta, "A 3.6pW/frame-pixel 1.35V PWM CMOS imager with dynamic pixel readout and no static bias current," *IEEE International Solid-State Circuits Conference*, pp. 54-55, Feb. 2008.

[38] B. Fowler, A.E. Gamal, and D.X.D. Yang, "A CMOS area image sensor with pixel-level A/D conversion," *IEEE International Solid-State Circuits Conference*, pp. 226-227, Feb. 1994.

[39] D.X.D. Yang, B. Fowler, and A.E. Gamal, "A nyquist-rate pixel-level ADC for CMOS image sensors," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, Mar. 1999.

[40] M.L. Zhang, A. Bermak, X.W. Li, and Z.H. Wang, "A low power CMOS image sensor design for wireless endoscopy capsule," *IEEE Biomedical Circuits and Systems Conference*, pp. 397-400, Nov. 2008.

[41] K.B. Cho, A. Krymski, and E.R. Fossum, "A 1.2V micropower CMOS active pixel image sensor for portable applications," *IEEE International Solid-State Circuits Conference,* pp. 114-115, 2000.

[42] K.B. Cho, A. Krymski, and E.R. Fossum, "A 3-pin 1.5V 550uW 176×144 self-clocked CMOS active pixel image sensor," *IEEE International Symposium on Low Power Electronics and Design*, pp. 316-321, 2001.

[43] B.J. Hosticka, "Analog circuits for sensors," *IEEE European Solid State Device Research Conference*, pp. 97-102, Sep. 2007.

[44] M. Barbaro, P.-Y. Burgi, A. Mortara, P. Nussbaum, and F. Heitger, "A 100×100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, Feb. 2002.

[45] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni, "A CMOS Image Sensor With Programmable Pixel-Level Analog Processing," *IEEE Transactions on Neural Networks,* vol. 16, (6), pp. 1673-1684, Nov. 2005.

[46] D. Jerome.,G. Dominique, and P. Michel, "A Single-Chip 10000 Frames/s CMOS Sensor with In-Situ 2D Programmable Image Processing," *The International Workshop on Computer Architecture for Machine Perception and Sensing*, pp. 124-129, Aug. 2006.

[47] Y. Oike, M. Ikeda, and K. Asada, "High-Sensitivity and Wide-Dynamic-Range Position Sensor Using Logarithmic-Response and Correlation Circuit," *IEICE Transactions on Electronics,* vol. E85-C, (8), pp. 1651-1658, Aug. 2002.

[48] R.M. Philipp, and R. Etienne-Cummings, "A 128×128 33mW 30 frames/s single-chip stereo imager," *IEEE International Solid-State Circuits Conference*, pp. 506-507, Feb. 2006.

[49] S. Kawahito, Y. Tadokoro, and A. Matsuzawa, "CMOS image sensors with video compression," *Proceedings of the ASP*-DAC, pp. 595-600, Feb. 1998.

[50] V. Gruev, and R. Etienne-Cummings, "Implementation of steerable spatiotemporal image filters on the focal plane," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 4, pp. 233-244, Apr. 2002.

[51] V. Brajovic, K. Mori, and N. Jankovic, "100 frames/s CMOS range image sensor," *IEEE International Solid-State Circuits Conference*, pp. 256-257, 2001.

[52] K. Yoonm C. Kim, B. Lee, and D. Lee, "Single-chip CMOS image sensor for mobile applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 12, pp. 1839-1845, Dec. 2002.

[53] S. Smith, J. Hurwitz, M. Torrie, D. Baxter, A. Holmes, M. Panaghiston, R. Henderson, A. Murray, S. Anderson, and P. Denyer, "A single-chip 306×244-pixel CMOS NTSC video camera," *IEEE Solid-State Circuits Conference*, pp. 170-171, Feb. 1998.

[54] S.S. Chen, A. Bermak, Y, Wang, and D. Martinez, "A CMOS image sensor with combined adaptive-quantization and QTD-based on-chip compression processor," *IEEE Custom Integrated Circuits Conference*, pp. 329-332, Sept. 2006.

[55] T. Eki, S. Kawahito, and Y. Tadokoro, "An on-sensor bit-serial column-parallel processing architecture for high-speed discrete fourier transform," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 8, pp. 642-646, Aug. 2006.

[56] T. Morris, E. Fletcher, C. Afghahi, S. Issa, K. Connolly, and J.C. Korta, "A Column-based processing array for high-speed digital image processing," *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, pp. 21-24, March 1999.

[57] Y. Nishikawa, S. Kawahito, M. Furuta, T. Tamura, "A high-speed CMOS image sensor with on-chip parallel image compression circuits," *IEEE Custom Integrated Circuits Conference*, pp. 833-836, Sept. 2007.

[58] S. Uramoto, Y. Inoue, J. Takeda, A. Takabatake, H. Terane, and M. Yoshimoto, "A 100MHz 2D discrete cosine transform core processor," *IEEE Symposium on VLSI Circuits*, pp. 35-36, 1999.

[59] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K.J. Shi, *Low power methodology manual*, 1st ed., Springer, 2007.

[60] A. Wang, and A. Chandrakasan, "A 180mV FFT processor using subthreshold circuit techniques," *IEEE International Solid-State Circuits Conference*, pp. 292-293, Feb. 2004.

[61] J. Rabaey, *Low Power Design Essentials*, 1st ed., Springer, 2009.

[62] A.P. Chandrakasan, S. Sheng, and R.W. "Low power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, Apr. 1992.

[63] P. Zuchowski, "Design strategies for low power ASICs," *IBM Technology Group New England Design Forum*, June 18, 2003.

[64] S. Shibatani and A. Li, "Pulse-latch approach reduces dynamic power," *EE Times-India*, August 2006.

[65] J. Tschanz, S. Narendra, Z.P. Chen, S. Borkar, B. Sachdev, and De Vivek, "Comparative delay and energy of single edge-triggered and dual edge-triggered pulsed flip-flops for high-performance microprocessors," *IEEE International Symposium on Low Power Electronics and Design*, pp.147-152, 2001.

[66] H. Partovi, R. Burd, U. Salim, F. Weber, L. DiGregorio, and D. Draper, "Flow-through latch and edge-triggered flip-flop hybrid elements," *IEEE International Solid-State Circuits Conference*, pp. 138-139, Feb. 1996.

[67] M. Ohashi, T. Hashimoto, S.I. Kuromaru, M. Matsuo, T. Mori-iwa, M. Hamada, Y. Sugisawa, M. Arita, H. Tomita, M. Hoshino, H. Miyajima, T. Nakamura, K.I. Ishida, T. Kimura, Y. Kohashi, T. Kondo, A. Inoue, H. Fujimoto, K.Watada, T. Fukunaga, T. Nishi, H. Ito, and J. Michiyama, "A 27MHz 11.1mW MPEG-4 video decoder, LSI for mobile application," *IEEE Solid-State Circuits Conference*, vol. 1, pp. 366-474, 2002.

[68] H, Noguchi, Y. Iguchi, H. Fujiwara, Y. Morita, K. Nii, H. Kawaguchi, and M. Yoshimoto, "A 10T Non-Precharge Two-Port SRAM for 74% Power Reduction in Video Processing," *IEEE Computer Society Annual Symposium on VLSI*, pp. 107-112, March 2007.

[69] H, Noguchi, Y. Iguchi, H. Fujiwara, Y. Morita, K. Nii, H. Kawaguchi, and M. Yoshimoto, "Which is the best dual-port SRAM in 45nm process technology? – 8T, 10T single end,

and 10T differential," *IEEE Integrated Circuit Design and Technology and Tutorial*, pp. 55-58, June 2008.

[70] R. Zimmermann, and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1079-1089, July 1997.

[71] J. Yuan, and C. Svensson, "High-speed CMOS circuit techniques," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 62-70, Feb. 1989.

[72] D. Draper, M. Crowley, J. Holst, G. Favor, A. Schoy, J. Trull, A. Ben-Meir, R. Khanna, D. Wendell, R. Krishna, J. Nolan, D. Mallick, H. Partovi, M. Roberts, M. Johnson, and T. Lee, "Circuit techniques in a 266-MHz MMX-enabled processor," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1650-1664, Nov. 1997.

[73] J.B. Kuang, T.C. Buchholtz, S.M. Dance, J.D. Warnock, S.N. Storino, D. Wendel, and D.H. Bradley, "A Double-Precision Multiplier with Fine-Grained Clock-Gating Support for a First-Generation CELL Processor," *IEEE International Solid-State Circuits Conference,* pp. 378-605, Feb. 2005.

[74] S.B. Wijeratne, N. Siddaiah, S.K. Mathew, M.A. Anders, R.K. Krishnamurthy, J. Anderson, M. Ernest, and M. Nardin, "A 9-GHz 65-nm Intel Pentium 4 Processor Integer Execution Unit," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, Jan. 2007.

[75] R. Zimmermann, and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, July 1997.

[76] D. Wang, M.F. Yang, W. Cheng, X.G. Guan, Z.M. Zhu, and Y.T. Yang, "Novel low power full adder cells in 180nm CMOS technology," *IEEE Conference on Industrial Electronics and Applications*, pp. 430-433, May 2009.

[77] P. Ng, P.T. Balsara, and D. Steiss, "Performance of CMOS differential circuits," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 841-846, June 1996.

[78] K. Chu, and D. Pulfrey, "A comparison of CMOS circuit techniques: differential cascade voltage switch logic versus conventional logic," *IEEE Journal of Solid-State Circuits*, vol. 22, pp. 528-532, Aug. 1987.

[79] L. Patrik, S. Christer, "Noise in digital dynamic CMOS circuits,", *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 655-662, June 1994.

[80] V.D. Agrawal, "Low-power design by hazad filtering," *IEEE International Conference on VLSI Design*, pp. 193-197, Jan. 1997.

[81] Y.L. Lu, and V.D. Agrawal, "Total power minimization in glitch-free CMOS circuits considering process variation," *IEEE International Conference on VLSI Design*, pp. 527-532, Jan. 2008.

[82] N.Rollins, and M.J. Wirthlin, "Reducing energy in FPGA multipliers through glitch reduction", *MAPLD International Conference*, Sept. 2005.

[83] R. Amirtharajah, and A. P. Chandrakasan, "A micropower programmable DSP approximate signal processing based on distributed arithmetic," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 337-347, Feb. 2004.

[84] R. Amirtharajah, J. Collier, J. Siebert, B. Zhou and A. Chandrakasan, "DSPs for energy harvesting sensors: applications and architectures," *IEEE Pervasive Computing*, vol. 4, no. 3, pp. 72-79, July-Sept. 2005.

[85] Y.N. Chang, J.H. Satyanarayana, and K.K. Parhi, "Systematic design of high-speed and low-power digital serial multipliers," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 12, Dec. 1998.

[86] S. Kao, R. Zlatanovici, and B. Nikolic, "A 240ps 64b carry-lookahead adder in 90nm CMOS," *IEEE Solid-State Circuit Conference*, pp. 1735-1744, Feb. 2006.

[87] P. Soderquist, and M. Leeser, "Division and square root: choosing the right implementation," *IEEE Micro*, vol. 17, no. 4, Aug. 1997.

[88] Y.M. Li, and W.M. Chu, "A new non-restoring square root algorithm and its VLSI implementations," *International Conference on Computer Design*, Oct. 1996.

[89] Lars Wanhammar, *DSP Integrated Circuits*. Academic Press, 1999.

[90] H. Mahmoodi-Meimand, and K. Roy, "Dual-edge triggered level converting flip-flops," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp.661-664, May 2004.

[91] V. Stojanovic, and V.G. Oklobdzija, "Comparative analysis of master-slave latches and flip-flops for high performance and low-power systems," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 4, pp. 536-548, Apr. 1999.

[92] E. Chaniotakis, P. Kalivas, and K.Z. Pekmestzi, "Long number bit-serial squarers," *IEEE Symposium on Computer Arithmetic*, pp. 29-36, June 2005.

[93] S.B. Wijeratne, N. Siddaiah, S.K. Mathew, M.A. Anders, R.K. Krishnamurthy, J. Anderson, M. Ernest, and M. Nardin, "A 9-GHz 65-nm Intel Pentium 4 processor integer execution unit," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 26-37, Jan. 2007.

[94] A.K Dunn, H Bolay, M.A Moskowitz and D.A Boas, "Dynamic imaging of cerebral blood flow using laser speckle", *Journal of Cerebral Blood Flow and Metabolism*, 21:195-201, [Online]. Available: http://www.nmr.mgh.harvard.edu/~adunn/speckle/software/speckle_software.html, 2001.

[95] X. Xie, G. L. Li, X. K. Chen, X. W. Li, and Z. H. Wang, "A low-power digital IC design inside the wireless endoscopic capsule," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 11, pp. 2390-2400, Nov. 2006.

[96] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakuma, and T. Sakurai, "A0.9V, 150 MHz, 10mW, 4mm^2 2-D discrete cosine transform core processor with variable threshold voltage (VT) scheme,", *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1770-1779, Nov. 1996.