

Alejandro Rodríguez Ramos

## **Low-Power and High-Fanout Bus Design Techniques**

**School of Electrical Engineering**

Espoo 31.03.2014

**Project supervisor:**

Prof. Jussi Rynänen

**Project advisor:**

PhD. Lauri Koskinen

Author: Alejandro Rodríguez Ramos

Title: Low-Power and High-Fanout Bus Design Techniques

Date: 31.03.2014

Language: English

Number of pages:7+66

Department of Micro- and Nanosciences

Professorship: Electronic Circuit Design

Code: S-87

Supervisor: Prof. Jussi Rynnänen

Advisor: PhD. Lauri Koskinen

Low-power techniques pose an important concern, when designing autonomous electronic devices. Most of the upcoming applications increasingly demand high performance and low-power consumption. In this thesis work, two low-power and high-fanout bus design techniques are reviewed. *Pulse Width Modulation* (PWM) and *Time-Domain Conversion* (TDC) approaches are elucidated. Schematic simulations (*Cadence*), quantitative and comparative results of both approaches are included. Additionally, on-chip wire theory is shown as well as some optimized bus simulation models (*MATLAB*), concluding with a summary of the main application areas for this techniques. Finally, two ready-to-use library cells are generated, as well as *Verilog* code for the *TDC* system.

Keywords: low-power, high-fanout, bus, Hardware Neural Networks, Network-On-Chip, Pulse Width Modulation, time-domain, pulse, width modulation, wire model, repeater

## Preface

The work for this thesis was carried out at the Electronic Circuit Design Laboratory (ECDL) of Aalto University. This work is part of an International Exchange Program (Erasmus), coordinated by Universidad Politécnica de Madrid (UPM) and Aalto University.

Thanks to all who have helped me along the way. First, I need to thank PhD. Lauri Koskinen for helping me to become involved with ECDL, for giving me the opportunity to participate in related projects and for his close guidance along the progress of the work. Thanks to Matthew J. Turnquist for solving all of those little questions which let me save other 2 hours on the topic. Thanks to others in the lab who helped me along the way too.

Thanks also to my friends there in Tenerife, Gran Canaria, Madrid and Helsinki for letting me live all of those amazing moments with you.

My thanks also goes to my family for supporting my education and goals the entire way. This is the end of a stage and I will be forever grateful. Thanks to my mother Mari and my father Prudencio (enumerated in alphabetic order, no jealousy!). Also, thanks to my little sister Marta, a soon-to-be electronic engineer (I am sure you are becoming the best one!).

Special thanks to my girlfriend Andrea (AKA Andrius) for being there in the distance and for not giving up along this year. You have been my inspiration all of those blue days here in Finland.

¡Chacho, que ya acabé!

Otaniemi, 07.5.2014

Alejandro Rodríguez Ramos

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Preface</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>Symbols and abbreviations</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Fundamentals of Wiring Theory</b>	<b>3</b>
2.1 The Wire . . . . .	3
2.1.1 Basics . . . . .	3
2.1.2 Capacitance . . . . .	4
2.1.3 Resistance . . . . .	5
2.1.4 Inductance . . . . .	6
2.1.5 Wire Model . . . . .	6
2.1.6 Crosstalk Effect . . . . .	7
2.2 Optimization Procedure for the Size and Amount of Repeaters . . . . .	9
2.3 Total Wire Model . . . . .	13
<b>3 Crosstalk-aware Pulse Width Modulation Bus Technique</b>	<b>17</b>
3.1 PWM-Based Signaling Concept . . . . .	17
3.2 Crosstalk Effect in PWM Signals . . . . .	19
3.3 Encoder-Decoder Circuits . . . . .	20
3.3.1 Encoder . . . . .	20
3.3.2 Decoder . . . . .	22
3.3.3 Crosstalk-Aware Circuitry . . . . .	23
3.4 Simulation and Results . . . . .	26
3.4.1 Simulation . . . . .	26
3.4.2 Results . . . . .	32
<b>4 Time-Domain Analog Conversion Bus Technique</b>	<b>34</b>
4.1 Time-Domain Signaling Concept . . . . .	34
4.2 Encoder-Decoder Circuits . . . . .	35
4.2.1 Encoder . . . . .	35
4.2.2 Decoder . . . . .	36
4.3 Synthesizable Verilog Description . . . . .	38
4.4 Simulation and Results . . . . .	38
4.4.1 Simulation . . . . .	38
4.4.2 Results . . . . .	42
<b>5 Bus Design Techniques Benchmark</b>	<b>44</b>

6 Conclusions and Future Work	47
References	48
Appendices	
A Appendix	50
B Appendix	52
C Appendix	53
D Appendix	54
E Appendix	55
F Appendix	56
G Appendix	57
H Appendix	58
I Appendix	59
J Appendix	61
K Appendix	63
L Appendix	64

# Symbols and abbreviations

## Symbols

$\Delta V$	Voltage drop
$\varepsilon_{di}$	Dielectric permittivity
$\rho$	Wire conductivity
$\omega$	width of the parallel plate capacitance
$\mu$	Permeability of surrounding dielectric
$A$	Wire cross-section
$a$	Normalized sum of NMOS and PMOS gate widths
$c_{wire}$	Wire capacitance
$c_{pp}$	Parallel-plate capacitance
$c_{fringe}$	Fringe capacitance
$cl$	Capacitance and inductance per unit product
$C_w$	Wire capacitance per unit
$C_d$	Drain capacitance
$C_g$	Gate capacitance
$d$	Wire distance
$E_{tot}$	Total energy dissipation
$E_b$	Energy per bit
$FO4$	Fan-out 4 technology value
$H$	Wire height
$i$	Current
$L$	Wire length
$L_i$	Self-inductance
$l$	Wire length
$l_{opt}$	Optimum segment length
$\hat{l}$	Segment length error
$L_{wire}$	Wire length
$L_{eff}$	Effective length
$N$	Max. binary value of a bus
$n$	Bus width (in bits)
$P_{tot}$	Total power consumption
$P_{encoder}$	Encoder power consumption
$P_{decoder}$	Decoder power consumption
$P_{wire}$	Wire power consumption
$R$	Wire resistance
$R_{\square}$	Sheet resistance
$R_w$	Wire resistance per unit
$r$	Resistance per unit
$t_{di}$	Thickness of the dielectric layer
$t$	Time
$t_d$	Delay time
$t_r$	Rise time

$T_{del}$	DTC minimum time
$T_{setup}$	Setup time
$W$	Wire width
$w$	Transistor width
$w_{opt}$	Optimum transistor width
$\hat{w}$	Transistor width error
$W1$	Minimum pulse width
$W2$	Medium pulse width
$W3$	Wider pulse width

## Opetators

$\Delta$	Delta operator
$\partial$	Partial derivation

## Abbreviations

ADC	Analogic-to-Digital Converter
CA	Crosstalk-Aware
CMOS	Complementary metal-oxide-semiconductor
DDR	Double-Data Rate
DAC	Digital-to-Analogic Converter
DTC	Digital-to-Time Converter
EDA	Electronic Design Automation
FDSOI	Fully Depleted Silicon On Insulator
HNN	Hardware Neural Network
NoC	Network-on-Chip
NMOS	Negative-channel Metal-Oxide Semiconductor
PWM	Pulse-Width Modulation
PMOS	Positive-channel Metal-Oxide Semiconductor
SoC	System-on-Chip
TDC	Time-to-Digital Converter
VLIW	Very Long Instruction Word

# Chapter 1

## Introduction

Currently, autonomous digital devices are widely extended, targeting different application areas. Most of these applications increasingly demand high performance and low power consumption. In addition, the new age of internet digital services, big data and algorithms for processing the data, raise the performance and power requirements. Since Moore's Law has continued, many-core processors have been developed and even more information must be communicated between cores and shared caches.

In this context, after a long period of research and publications, *network-on-chips* (NoCs) have begun to be incorporated to commercial designs. NoCs have proven themselves to scale better than bus-based designs. However, NoCs consume a large portion of the total system power. Some examples show a consumption of 35% of the total chip power [1]. Additionally, computing platforms are radically changing from traditional sequential to highly parallel architectures, such as VLIW (*Very Long Instruction Word*) architectures. Furthermore, *Hardware Neural Networks* (HNNs) have been proven to meet the required degree of parallelism in certain applications [2]. Nevertheless, several problems reduce the capability of successfully implementing a HNN (storing the weights, calculating the values by fast multiplication, designing a parallel architecture and providing a complex interconnectivity) [3].

As a result, on-chip global communication will remain a primary concern in terms of latency and power. In order to improve energy, performance and power consumption of full-swing global wires, a number of techniques, including encoding [4], [5], pulsed signaling [6], [7] and alternating repeaters [8], [9], have been proposed [10]. These approaches lower the energy consumption, without changing the number of wires.

Low-swing techniques have demonstrated to reduce clock power by 66% [1]. Due to the analog essence of these techniques, work until now has focused on differential signaling (both current [11] and voltage [12] sensing). First, low-swing signaling in global wires can be perilous with respect to noise margin and signal integrity, especially when full-swing logic circuits are in the vicinity and creates large coupling noise episodes. Second, a large-area and power-hungry *analog-to-digital converter* (ADC) and a *digital-to-analog converter* (DAC) are necessary for the interface between an analog system and a digital core [13]. Third, highly custom designs compose a common problem for the standard *Electronic Design Automation* (EDA) flow, which is often carried out with synthesized circuits.

In summary, long wires high-power consumption, high interconnect density in the upcoming *System-On-Chips* (SoCs), coupling capacitances between wires (which increase the minimum pitch and/or decrease the bus data rate), as well as high performance constraints, are critical in new design approaches. In order to address these problematic requirements, *Pulse Width Modulation* (PWM) and *Time-Domain*



*Conversion* (TDC) bus designs are reviewed in this thesis [10] [13]. Although they have an analog main focus, both techniques are fully compatible with the standard EDA flow. These two approaches aim to reduce the number of wires, as well as lower the power consumption, providing a *state-of-the-art* bus performance.

First, PWM encoding technique combines the benefits of pulsed signaling and 2-bit signal encoding. A 2-bit bus is encoded into one pulsed signal. Also, pulses are susceptible to coupling phenomena, caused by neighboring wires. Aiming to reduce this coupling effects, a *crosstalk-aware* (CA) circuit is included in the design. Adding CA circuitry, either excessive width and separation margins or wire shields, in the global on-chip interconnect, is avoided.

Second, TDC approach encodes an N-bit bus into one time-domain wire. Since time is determinant in this technique, clock has to be propagated beside time-domain signal. However, the amount of required wires is highly reduced, even though it has a clock overhead. Also, it allows exploitation of time-domain resolution that is above voltage-domain resolution in deep-submicron CMOS technologies. Furthermore, several available calculations, which open up the possibility of a wide range of applications, are enhanced.

The remainder of this thesis is organized as follows. A review of the fundamentals of wiring theory is included in Chapter 2. Wire parameters, crosstalk effect and wire models are explained. Chapter 3 describes a *PWM* bus design technique. A second bus design (*TDC*) approach is illustrated in Chapter 4. A benchmark among these two techniques and conclusions are included in Chapter 5 and Chapter 6.

# Chapter 2

## Fundamentals of Wiring Theory

This chapter presents a brief overview of on-chip wiring. Basics are detailed, continuing with the most recent deep sub-micron CMOS considerations. The theory behind is shown, as well as some practical examples related to the main topic of this thesis. An optimization procedure for the number of required repeaters in an interconnect wire is included, explaining the calculations. Finally, a full wire model is presented, based on a nanoscale FDSOI process technology.

### 2.1 The Wire

Throughout most of the historical trajectory of integrated digital circuits, on-chip interconnect wires were considered to be second priority components. These components had only to be considered in special cases or when performing high-precision analysis. At the arrival of deep-submicron CMOS technologies, the parasitic effects which arise from interconnect wiring display a scaling behavior that differs from the previous semiconductor processes.

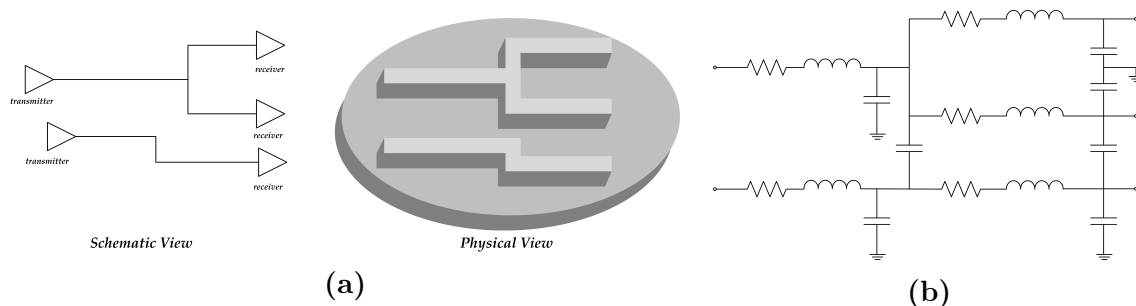
In this context, wires tend to gain in importance as device size is reduced and circuit performance is increased. This situation is complicated by the fact that the average length of an interconnect wire is substantially increasing, since production of larger dies is economically feasible. As a result, the aim of the following subsection is to take into account the fundamentals of wire study.

#### 2.1.1 Basics

The designer of an integrated circuit has multiple options in realizing the interconnections between the numerous components. Most of the standard schematic design tools view wires as nets, neglecting their effect on the whole circuit. In fact, the interconnect wires create a complex geometry that introduce capacitive, resistive, and inductive parasitics. All three have various effects on the circuit behavior.

1. The propagation delay increases, or, equivalently, performance drops.
2. The energy dissipation and the power distribution are strongly affected.
3. The reliability of the circuit is affected by the introduction of extra noise sources.

Therefore, is important to have a clear insight in the parasitic wiring phenomena, their relative importance and their models. However, very conservative models of the interconnect wires are dauntingly complex and it is only applicable to very small topologies. Hence, bringing all possible effects to bear, could turn the design and optimization process a "trial-and-error" operation rather than a focused search. This is best illustrated with the simple example, shown in Figure 2.1.



**Figure 2.1:** Figure 2.1a Schematic and physical views of a wiring of bus interconnection. Figure 2.1b Wire model of the circuit of Figure 2.1a considering most of the wire parasitics (with the exception of interwire resistance and mutual inductance).

Each wire in a bus network connects a transmitter(s) to a group of receivers and is implemented as a link of wire segments of various lengths and geometries, as is shown in Figure 2.1a. A full circuit model, taking into account the parasitic capacitance, resistance, and the inductance of the interconnections, is shown in Figure 2.1b. Notice that extra circuit elements are not located in a physical point, since they are distributed elements. In fact, these extra components become a necessity when the length of the wire gets significantly larger than its width. Fortunately, substantial simplifications can often be made. Some of them are enumerated below.

- If the resistance of the wire is significant (long wires with small cross-section), inductive effects can be neglected.
- If the wires are short, the cross-section of the wire is large (or interconnect material used has a low resistivity), a capacitance-only model can be used.
- Finally, if the separation between neighboring wires is large, or if the wires only run together for a short distance, inter-wire capacitance can be ignored.

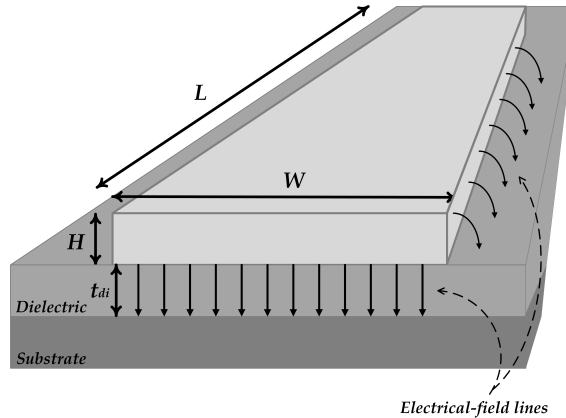
An experienced designer knows to differentiate between first and second order effects. In order to address all of this possible simplifications, is necessary to quick estimate the wire parameters, which are explained in the following sections, as well as to be able to choose correctly the most suitable wire model for a specific design.

### 2.1.2 Capacitance

Modeling the wire capacitance(s) in a integrated circuit is non-trivial, as interconnect structure of contemporary integrated circuits is three-dimensional. The capacitance of such a wire is a function of its shape, its environment, its distance to the substrate, and the distance to surrounding wires. Rather than going into the analysis of complex equations and models, a designer will typically use an advanced extraction tool to get accurate values of the interconnect of the completed layout. Nevertheless, some simple first-order models can provide basic understanding of the nature of interconnect capacitance and its parameters.

A simple model of the inherent capacitance in an interconnect wire, is shown in Figure 2.2. This model includes *fringing fields*, which are modeled by extra

capacitances called *fringing-field capacitances*. Other simplistic models, where such fields are not considered, become inaccurate while scaling technology.



**Figure 2.2:** Wire model with parallel and fringing-field coupling lines.

Therefore, the capacitance is approximated as the sum of two components: a parallel-plate capacitance determined by the orthogonal field between a wire of width  $w$  and the ground plane, in parallel with the fringing capacitance modeled by a cylindrical wire with a dimension equal to the interconnect thickness  $H$ . Resulting approximation is shown in Equation 2.1. It is simple but works fairly well in practice.

$$c_{wire} = c_{pp} + c_{fringe} = \frac{w\epsilon_{di}}{t_{di}} + \frac{2\pi\epsilon_{di}}{\log(t_{di}/H)} \quad (2.1)$$

with  $w = W - H/2$  a good approximation for the width of the parallel-plate capacitor. Parallel-plate capacitance and fringe capacitance are represented by  $c_{pp}$  and  $c_{fringe}$ , respectively.  $t_{di}$  and  $\epsilon_{di}$  are the thickness of the dielectric layer and its permittivity.

Various more accurate models have been developed, but these tend to be significantly more complex. Also, those coupling models are out of the target of introducing a comprehensible wire model, becoming excessively complex to be easily included in a schematic simulation tool.

### 2.1.3 Resistance

The resistance of a wire is proportional to its length  $L$  and inversely proportional to its cross-section  $A$ . The resistance of a rectangular conductor (Figure 2.2) can be expressed as Equation 2.2.

$$R = \frac{\rho L}{A} = \frac{\rho L}{HW} \quad (2.2)$$

where the constant  $\rho$  is the resistivity of the material ( $\Omega m$ ). Since  $H$  is a constant of a given technology, Equation 2.2 can be rewritten as follows (Equation 2.3).

$$R = R_{\square} \frac{L}{W} \quad (2.3)$$

with

$$R_{\square} = \frac{\rho}{H} \quad (2.4)$$

the *sheet resistance* of the material ( $\Omega/\square$ ). This expresses that the resistance of a square conductor is independent of its absolute size. As explained, the resistance of a semiconductor wire is considered to be linear and constant. However, at very high frequency, an additional phenomenon arises, the *Skin Effect*. High-frequency currents tend to flow primary on the surface of a conductor with the current density falling off exponentially with depth into the conductor. Nevertheless, it has been demonstrated that *skin effect* is only an issue of the widest wires.

#### 2.1.4 Inductance

In the first decades of integrated circuit design, inductance is seen as a no impact component in an interconnection. Yet, with the adoption of low-resistive interconnect materials and the increase of clock frequencies to GHz-range, inductance starts to become more pronounced on a chip. Ringing and overshoot effects, reflections of signals due to impedance mismatch, inductive coupling between lines, and switching noise due to  $L_i di/dt$  voltage drops, are some consequences of chip inductance effect.

Regarding the definition of the inductance of a section of a circuit, which states that a voltage drop  $\Delta V$  is generated by a changing current passing through an inductor, the inductance can be expressed as Equation 2.5.

$$\Delta V = L_i \frac{di}{dt} \quad (2.5)$$

It is possible to calculate the wire inductance directly from its geometry and its environment. A simple approach propose that the capacitance  $c$  and the inductance  $l$  (per unit length) of a wire are related by the following expression (Equation 2.6).

$$cl = \varepsilon_{di}\mu \quad (2.6)$$

with  $\varepsilon_{di}$  and  $\mu$  respectively the permittivity and permeability of the surrounding dielectric. Notice that for this expression to be valid, the conductor must be completely surrounded by a uniform dielectric medium. However, even when the wire is embedded in different dielectric materials, its is possible to adopt *average* dielectric constants such that Equation 2.6 still can be used to get an approximate value of the inductance.

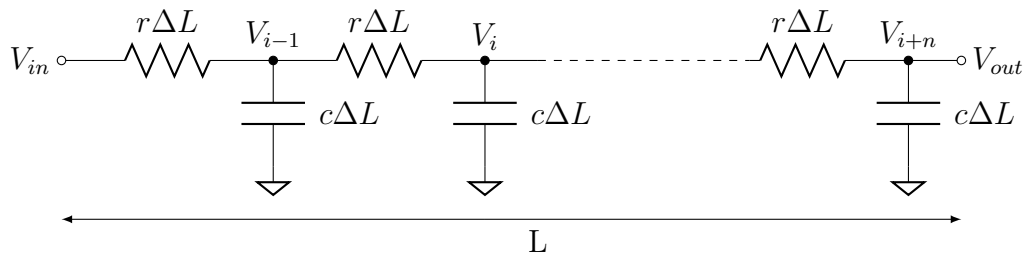
#### 2.1.5 Wire Model

Parasitic elements, studied in previous section, have an important impact on the electrical behavior of the circuit and influence its delay, power dissipation, and reliability. In order to simulate these effects, an introduction of electrical models which estimate and approximate the real behavior of a wire as a function of its parameters, is required.

In schematics, wires are assumed as *equipotential regions*, a voltage change at one end of the wire propagates instantly to its other ends. While this ideal-wire model is

simplistic, it has its value, especially in the early phases of the design process when the designer wants to concentrate on the properties and the grossly behavior of the design. Furthermore, when studying small circuit components such as gates, wires tend to be very short and their parasitics neglected, but with long wires parasitics arise as a considerable effect.

There is a wide variety of wire models, targeting different levels of accuracy, such as the *Lumped Model*, *Lumped RC Model* or the *Transmission Line Model*. However, the aim of this section is to introduce a first approximation of a wire model. It is not extremely complex but neither excessively simple, providing an acceptable level of accuracy.



**Figure 2.3:** Distributed RC Model.

In this model,  $L$  represents the total length of the wire, while  $r$  and  $c$  stands for the resistance and capacitance per unit length. The voltage at node  $i$  of this network can be determined by solving the following set of partial differential equations (Equation 2.7).

$$c\Delta L \frac{\partial V_i}{\partial t} = \frac{(V_{i+1} - V_i) + (V_i - V_{i-1})}{r\Delta L} \quad (2.7)$$

The correct behavior of a distributed  $rc$  line is then obtained by reducing  $\Delta L$  asymptotically to 0. For  $\Delta L \rightarrow 0$ , Equation 2.8 becomes the well-known *diffusion equation*.

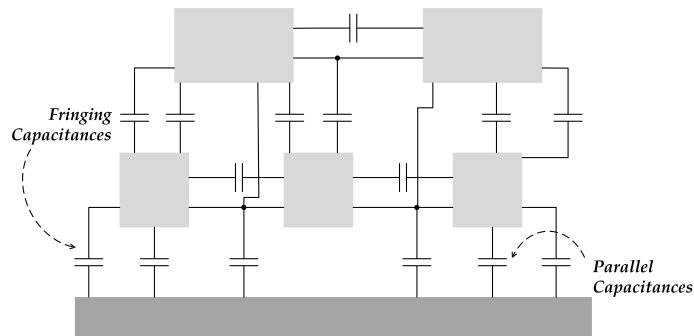
$$rc \frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} \quad (2.8)$$

where  $V$  is the voltage at a particular point in the wire, and  $x$  is the distance between this point and the signal source. Analytic utilization of this model can lead to complex calculations. Nevertheless, it can provide a simple first approximation while using EDA tools.

### 2.1.6 Crosstalk Effect

So far, this analysis has been based on a single rectangular conductor placed over a ground plane. This structure, called *microstripline*, can be an acceptable model for CMOS interconnections when the number of interconnect layers were restricted to 1 or 2. Current processes offer many more layers of interconnect, which packed quite densely in addition. In this scenario, the assumption that a wire is completely isolated from its surroundings structures and is only capacitively coupled to

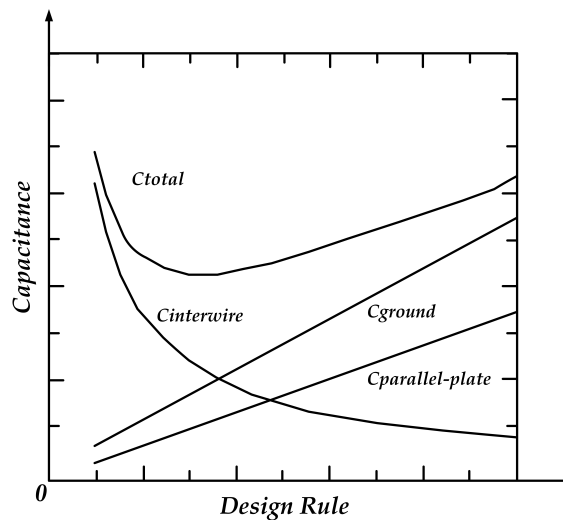
ground, becomes unrealistic. This is illustrated in Figure 2.4, where the capacitance components of a wire embedded in an interconnect hierarchy are identified.



**Figure 2.4:** Capacitive coupling between wires in interconnect hierarchy.

Each wire is not only coupled to the grounded substrate, but also to the neighboring wires on the same layer and on adjacent layers. To a first approximation, the total capacitance connected to a given wire is still similar. The main difference is that not all its capacitive components do terminate at the grounded substrate, but that a large number of them connect to other wires, which have dynamically varying voltage levels. These *floating* capacitors can either form a source of noise (crosstalk) or have a negative impact on the performance of the circuit. This effect becomes more important for wires in the higher interconnect layers, since these wires are farther away from the substrate.

The increasing contribution of the interwire capacitance to the total capacitance with decreasing wire sizes, is best illustrated by Figure 2.5.



**Figure 2.5:** Interconnect capacitance as a function of design rules. It consists of a capacitance to ground and a crosstalk capacitance. Adapted from [14].

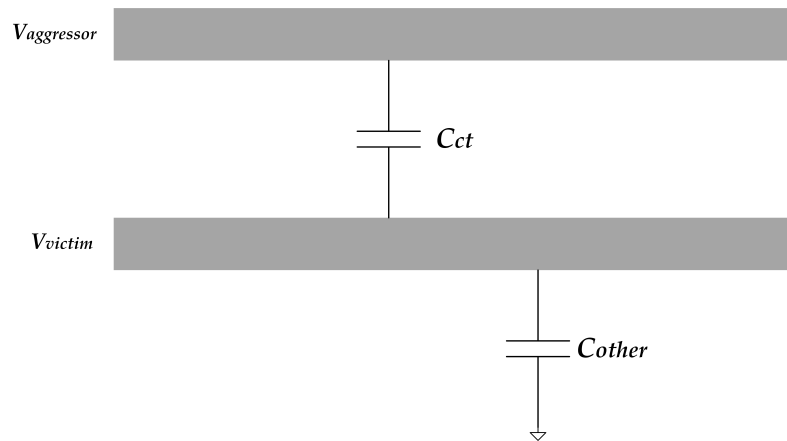
In Figure 2.5, which qualitatively plots the capacitive components of a group of parallel wires routed above a ground plane, it is assumed that dielectric and wire

thickness are held constant while scaling all other dimensions. When  $W$  becomes smaller than  $1.75H$ , the crosstalk capacitance starts to dominate.

In addition crosstalk effect is seen as an induced voltage between wires. This effect can be approximated by Equation 2.9.

$$V_{victim} = \frac{C_{ct}}{C_{ct} + C_{other}} \cdot V_{aggressor} \quad (2.9)$$

Where  $V_{victim}$  is the induced crosstalk voltage and  $V_{aggressor}$  is the generator of the aggressor signal. The crosstalk coupling capacitance between wires and the remaining coupling capacitances are represented by  $C_{ct}$  and  $C_{other}$ , respectively. The situation described by Equation 2.9 is represented in Figure 2.6.



**Figure 2.6:** Simplified model for induced crosstalk voltage variance between two parallel wires.

For signaling purposes, crosstalk induces delay (*jitter*) proportional to the induced voltage. Additionally, the rising edge and/or falling edge slope of the victim signal can be reduced. In conclusion, the interwire crosstalk contribution compose an important issue for the following PWM bus approach, since the length of a pulse is heavily affected by crosstalk. This fact is deeply analyzed in the following sections.

## 2.2 Optimization Procedure for the Size and Amount of Repeaters

As described in previous sections, wires have resistance, capacitance and inductance parasitics. Also, wire interconnects behave as a *transmission line* while increasing working frequency [14]. In this scenario, propagation delay and rise time are quadratic with wire length. This fact, which is aggravated in long wires, is shown in Equation 2.10 and 2.11.

$$t_d \approx 0.4 \cdot d^2 R_w C_w \quad (2.10)$$

$$t_r \approx d^2 R_w C_w \quad (2.11)$$

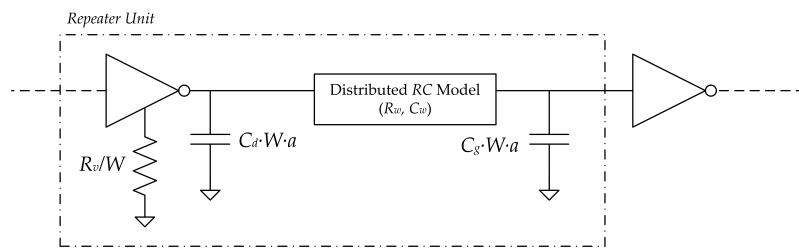
Where  $t_d$  and  $t_r$  are propagation delay and rise time, respectively.  $d$  represents wire length and  $R_w C_w$  is the wire resistance and capacitance product.



Therefore, to address this quadratic effect, the most common solution is to split a wire into a certain number of segments connected by repeaters. Each repeater regenerates the signal, providing a "new starting point", where propagation delay and rise time is still held linear with distance. Now, a new trade-off between number of repeaters in a wire and energy-delay optimization arises. Excessive number of repeaters leads to exorbitant energy consumption, and a very reduced amount causes again a quadratic delay and rise time, leading to a loss in performance.

In order to correctly optimize the delay and energy dissipation of a repeated wire, all the calculations have to be supported on a well-known wire model. In this section, using the *Distributed RC Model* introduced in previous sections, an energy-delay optimization for the size and amount of repeaters is presented.

This model consists on a simple set of linearized resistances and capacitances (Figure 2.7) broken up by repeaters. Repeaters used, to a first approximation, are usual CMOS inverters (instead of using more complex structures as buffers, tristates, etc.), so that the main topic of this thesis is not forgotten.



**Figure 2.7:** Distributed RC Wire Model broken up by inverters.

First, there are some considerations:

- Transistor width  $w$  ( $\mu m$ ), wire length  $l$  ( $\mu m$ ) and normalized sum of NMOS and PMOS gate widths  $a$  ( $w_{nmos}/w_{min} + w_{pmos}/w_{min}$ ) in an inverter, are technology-independent parameters.
- Wire resistance  $R_w$  ( $\Omega/\mu m$ ), wire capacitance  $C_w$  ( $F/\mu m$ ) and transistor drive and parasitics parameters  $R_v$  ( $\Omega \cdot \mu m$ ),  $C_d$  ( $F$ ) and  $C_g$  ( $F$ ), are technology-dependent parameters.
- Given a particular ratio of PMOS to NMOS device size ( $a$ ), designers can typically only vary  $w$  and  $l$  to optimize performance.

This representation has many limitations, as it cannot model effects of rise-times upon delay, non-linear transistor currents, or bias-dependent capacitances. However, for the aim of this thesis, this model provides enough accuracy, as is demonstrated in following sections.

Center-skewed inverters results in  $a = 3$  (PMOS twice NMOS size) and  $C_d = 0.5C_g$ . Following the derivation in [15], delay can be optimized with respect to both the number of stages  $N$  as well as the driver width  $w$ , ending up with the solution in

Equations 2.12, 2.13 and 2.14 [15]. In these expressions, the wire length and driver widths are in units of microns ( $\mu m$ ).

$$l_{opt} = 3\sqrt{\frac{R_v C_g}{R_w C_w}} \quad (2.12)$$

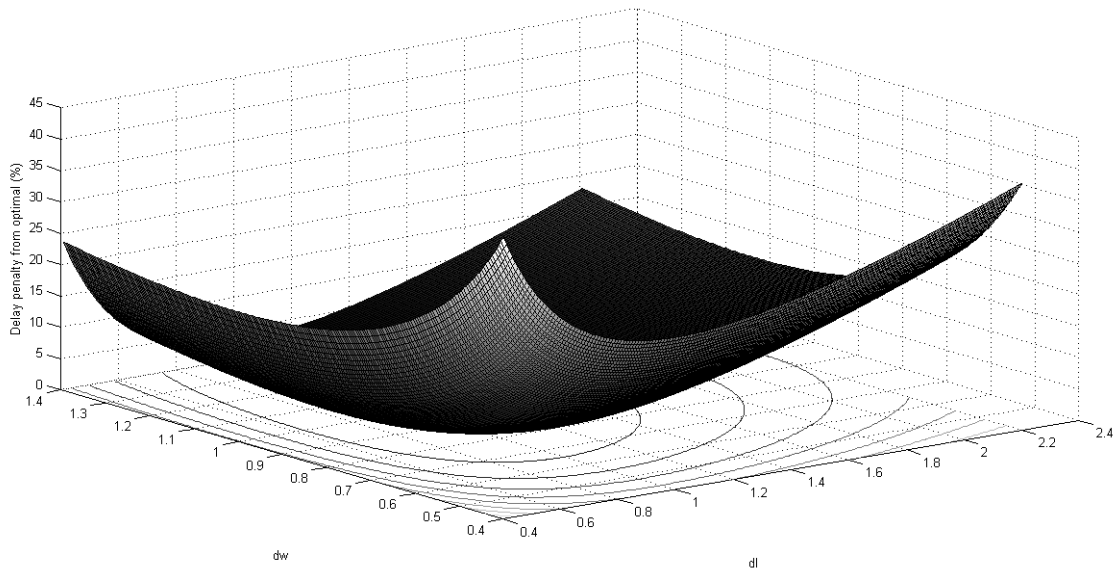
$$w_{opt} = \frac{1}{\sqrt{3}}\sqrt{\frac{R_v C_w}{R_w C_g}} \quad (2.13)$$

$$Delay/Unit - length = 1.47\sqrt{FO4 \cdot R_w C_w} \quad (2.14)$$

In practical designs, a precise optimal segment length between repeaters is rarely used, because the total end-to-end wire length is not often an integer number of optimal-length segments. Likewise, exact optimal device size is not used due to cell libraries which have limited sizing granularity. Therefore, actual segment length and device width can be expressed as  $l = \hat{l} \cdot l_{opt}$  and  $w = \hat{w} \cdot w_{opt}$ , respectively.  $\hat{l}$  and  $\hat{w}$  represent segment length error and width error. As a result, delay equation becomes independent of the absolute value of the segment length and device width, as is shown in Equation 2.15 [15].

$$Delay = [0.34(\hat{l} + \frac{1}{\hat{l}}) + 0.4(\hat{w} + \frac{1}{\hat{w}})]\sqrt{FO4 \cdot R_w C_w} \quad (2.15)$$

There is a penalty for grossly undersizing  $w$  and  $l$ , which is greater than grossly oversizing them. Nevertheless, slightly mis-sized  $w$  and  $l$  lead to only slight performance penalties (Figure 2.8).



**Figure 2.8:** Delay sensitivity (3% contours).

In figure 2.8 the delay penalty variance, due to the segment length error ( $dl = \hat{l}$ ) variance and the transistor width error ( $dw = \hat{w}$ ) variance, is represented by a

surface. The loss in delay performance with slight variances in  $\hat{w}$  and/or  $\hat{l}$  remains reduced.

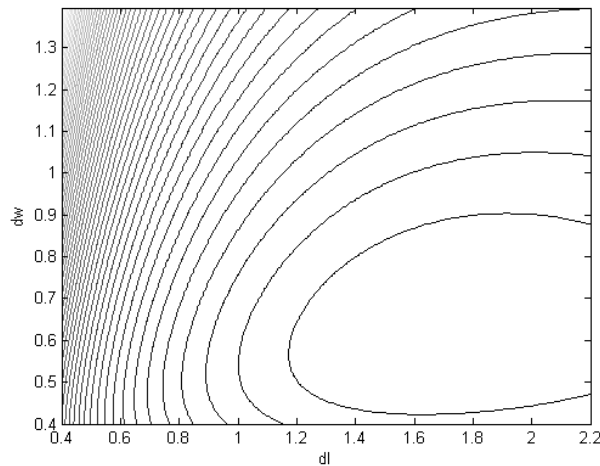
Delay-optimal solutions seem to be inefficient, as they naturally sacrifice large power and energy costs in order to gain marginal delay benefits. In conclusion, for a better power-saving efficiency, the optimal result is obtained by an energy-delay optimization. This discussion is focused primarily on switched capacitances, ignoring leakage currents. Short-circuit currents are approximated by a fixed multiplicative factor  $c$ . All this parameters result in Equation 2.16.

$$E_{tot} = C_w LV^2 \left( 1 + \frac{4.5}{3\sqrt{3}} c \frac{\hat{w}}{\hat{l}} \right) \quad (2.16)$$

Where  $L$  describes the wire length and  $V$  supply voltage. Contours shown below, in Figure 2.9, illustrate the optimal  $\hat{l}$  and  $\hat{w}$  for energy-delay product (Equation 2.17).

$$E_{tot}(\hat{w}, \hat{l}) \cdot Delay(\hat{w}, \hat{l}) \quad (2.17)$$

Energy-delay product is represented by Equation 2.17. Contours shown in Figure 2.9 illustrate the energy-delay value for each  $\hat{l}$  and  $\hat{w}$  pair. Region in the range of  $\hat{l} \approx [1.2, 3]$  and  $\hat{w} \approx [0.4, 0.9]$  shows the minimum energy-delay area.



**Figure 2.9:** 3% contours for energy-delay optimization.

At the energy-delay minimum, when  $\hat{l} = 1.59$  and  $\hat{w} = 0.62$ , the system has a delay of 11.5% worse than the delay-optimal value, and an energy of 28.4% better than that at the delay-optimal point.

Therefore, regarding previous optimal results  $(\hat{l}, \hat{w})$ , an optimal segment length and transistor width, for used process technology, is calculated. Final values can be obtained combining Equations 2.7 and 2.13, as well as considering following technology parameters (Table 2.1).

Parameter	Value	Unit
$L_{eff}$	$L \cdot x$	$\mu m$
$R_v$	$2500 \cdot Lx$	$\Omega \cdot \mu m$
$C_g$	$2 \cdot 10^{-15}$	$F/\mu m$
$C_d$	$0.5 \cdot C_g = 10^{-15}$	$F/\mu m$
$R_w$	$3.15 \cdot x$	$\Omega/\mu m$
$C_w$	$7.203 \cdot x \cdot 10^{-17}$	$F/\mu m$
$FO4$	11.44	$ps$
$L_{wire}$	4000	$\mu m$
$V$	1.2	$V$
$c$	1	$adim.$

**Table 2.1:** Required nanoscale FDSOI process technology parameters.

Where  $L_{eff}$  is effective gate length,  $R_v$  an approximation over different technologies of inverter parameters [15],  $C_g$  gate capacitance (approximated value over different technologies),  $C_d$  drain capacitance,  $FO4$  value for this process technology [16],  $L_{wire}$  wire length,  $V$  supply voltage and  $c$  shortcut currents constant.  $R_w$  and  $C_w$  are better defined in following sections, though they represent the resistance per length and the capacitance per length of the wire, respectively.  $L$  and  $x$  values cannot be revealed due to a NDA.

As mentioned, regarding Equations 2.7 and 2.13, as well as including  $l = \hat{l} \cdot l_{opt}$  and  $w = \hat{w} \cdot w_{opt}$ , the final results can be obtained (Table 2.2).

Parameter	Value	Unit
Optimal $\hat{l}$	1.59	$adim.$
Optimal $\hat{w}$	0.62	$adim.$
$l_{opt}$	78.54	$\mu m$
$w_{opt}$	1.63	$\mu m$
Final $l$	136.67	$\mu m$
Final $w$	1.012	$\mu m$
Final number of inverters	30	$adim.$

**Table 2.2:** Optimal values for a 4mm link (segments length and devices width).

These calculations has been carried out by a *MATLAB* script, shown in Appendix A. From now on, optimized segments of a wire and the most suitable repeaters size have been determined.

### 2.3 Total Wire Model

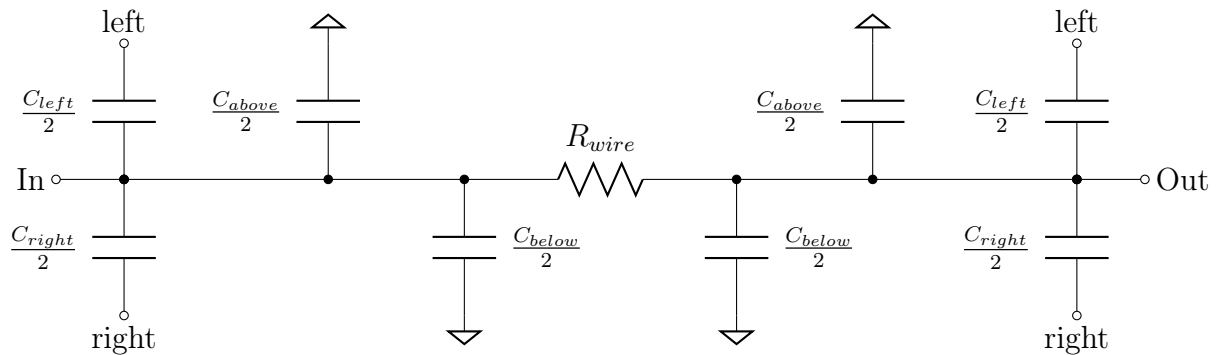
Once wire segments and repeaters have been defined, the *Distributed RC Model* has to be adjusted for a nanoscale FDSOI process technology. As shown in previous crosstalk section (Section 2.1.6), a coupling capacitance between neighboring wires

have been included. *Fringing capacitance* and *parallel-plate (pp)* capacitance can be added to simplify equations (Equations 2.18 and 2.19).

$$C_{below} = C_{below}^{pp} + 2C_{below}^{fringe} \quad (2.18)$$

$$C_{left/right} = C_{left/right}^{pp} + C_{left/right}^{abovefringe} + C_{left/right}^{belowfringe} \quad (2.19)$$

Upper and lower metal layer in the model are assumed to be connected to ground, so that the simulations can be simplified. Therefore, the approximated *Distributed RC* model of wire segment (length of wire between repeaters) with coupling capacitances, is shown in Figure 2.10. This model presented is the *worst case* scenario, as it provides a C-R-C network, instead of using a simpler R-C configuration. In addition, the following analysis, emphasize this *worst case* assumption while choosing technology parameters values (wire width, separation, metal layer, etc.).



**Figure 2.10:** Distributed RC Model of the wire section with coupling capacitances.

With *In* as the input of a wire segment and *Out* as the output. The *left* and *right* ports are meant to be connected to surrounding wires. As shown,  $C_{above}$  and  $C_{below}$  are connected to ground. Equivalently, consecutive higher and lower metal layers, regarding this model, are connected to ground, so that model complexity is reduced.

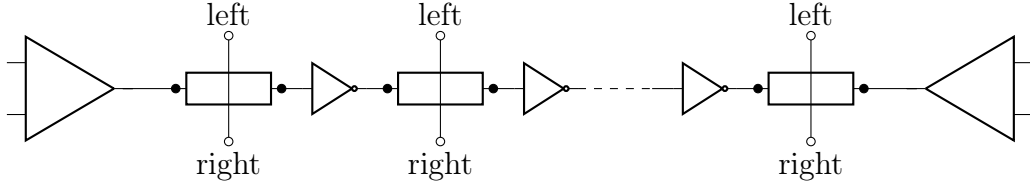
Middle layers in practical designs are usually reserved for routing. Hence, metal 4 layer is suitable for the design of a 4mm link. Values of previous model parameters for a nanoscale FDSOI process technology are shown in Table 2.3.

Parameter	Value	Unit
$C_{above}$	$y$	$fF/\mu m$
$C_{below}$	$y/0.4405$	$fF/\mu m$
$C_{left}$	$y/0.144$	$fF/\mu m$
$C_{right}$	$y/0.144$	$fF/\mu m$
$C_{wire}$	$y/0.0583$	$fF/\mu m$
$R_{wire}$	$y/0.00133$	$\Omega/\mu m$

**Table 2.3:** nanoscale FDSOI process technology parameters (M4 layer, minimum pitch and separation).

These values assume wiring at minimum pitch, at the same level, and with 100% metal coverage above and below. Also, all parameters are assumed to be nominal.  $y$  cannot be revealed due to a NDA.

These results and those corresponding to previous section, set up the total wire model, ready to use in the following simulations. Finally, this model can be used in designs as a *test wire model*, as illustrated in Figure 2.11.



**Figure 2.11:** Total Wire Model, including the encoder-decoder scheme.

Components between inverters represent the *Distributed RC Model* of a wire segment. The *left* and *right* ports have to be shorted to neighboring wires, as those wires in the vicinity include an exact wire model.

Regarding the results obtained in Section 2.2 and Table 2.3, the model is fully defined. Final values are enumerated in Table 2.4.

Parameter	Value	Unit
$C_{above}$	$y$	$fF$
$C_{below}$	$y/0.4405$	$fF$
$C_{left}$	$y/0.144$	$fF$
$C_{right}$	$y/0.144$	$fF$
$C_{segment}$	$y/0.000426$	$fF$
$R_{segment}$	$102502 \cdot y$	$\Omega$
Segment Length	136.67	$\mu m$
Inverter Width	1.012	$\mu m$
Number of stages	30	<i>adim.</i>
$L_{total}$	4	<i>mm</i>

**Table 2.4:** Distributed RC model parameters for a nanoscale FDSOI process technology.

Including the explained model in a schematic simulation tool, a first approximation of the behavior of an interconnection affected by crosstalk can be easily simulated. If the layouts of the actual design are available, parasitics can be easily extracted. However, in order to grossly adjust an encoding-decoding scheme, before going through layouts of an specific design, this characterization is completely suitable.

This procedure provides delay, crosstalk, rise-fall time, and other typical information, for designers who prefer starting by a first approximation of the behavior of a long link. Nevertheless, final design includes 134x inverters and a segment length 51% lower, differing from the results predicted by this model. Final results are shown in Table 2.5.

Parameter	Value	Unit
$C_{above}$	$y$	$fF$
$C_{below}$	$y/0.4405$	$fF$
$C_{left}$	$y/0.144$	$fF$
$C_{right}$	$y/0.144$	$fF$
$C_{segment}$	$y/0.000426$	$fF$
$R_{segment}$	$102502 \cdot y$	$\Omega$
Segment Length	60.8	$\mu m$
Inverter Width	134x	–
Number of stages	30	<i>adim.</i>
$L_{total}$	2.04	<i>mm</i>

**Table 2.5:** Final wire model parameters for a nanoscale FDSOI process technology.

In conclusion, this model may not be appropriate for pulse signaling techniques, where pulses are thin comparing to normal digital operation. This leads to an oversized repeater width, as well as a reduction of the segment length.

# Chapter 3

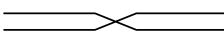

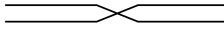
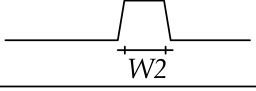
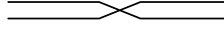
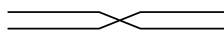
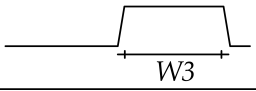
## Crosstalk-aware Pulse Width Modulation Bus Technique

This chapter reviews a *Pulse Width Modulation* (PWM) bus technique. A brief explanation of the concept behind is included, focusing on the modulation type and the crosstalk basics. Then, the whole encoder-decoder set is described, going through simulations. Finally, illustration of results and conclusions are incorporated. All of the work in this chapter represents a nanoscale FDSOI process technology adaptation of the mono-PWM bus design included in [10].

### 3.1 PWM-Based Signaling Concept

As explained in previous section, in order to accommodate a voluminous interconnection density in microprocessor designs, wiring pitch has to be small, leading to high RC values. To address these issues, this approach aims to half the amount of wires in a chip, or at least, in the longest and power-consuming buses.

The concept is based on the conversion of a 2-bit bus into a 1-bit bus, by shortening or lengthening a pulse width depending on the actual combination of the input bits. This is best illustrated in Figure 3.1.

Input	PWM
$D0$ _____ $D1$ _____	_____
$D0$  $D1$ _____	 $W1$
$D0$ _____ $D1$ 	 $W2$
$D0$  $D1$ 	 $W3$

**Figure 3.1:** PWM encoding technique (2-bit bus).  $D0$  and  $D1$  information is encoded into 3 different pulse widths.

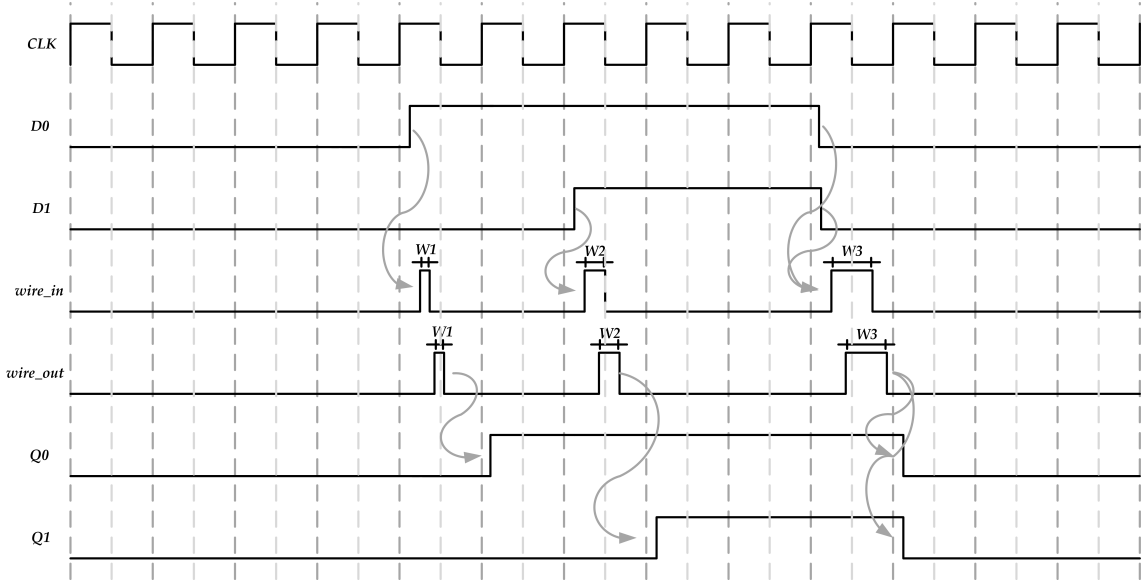
Regarding Figure 3.1, as well as the following enumeration, PWM code is simply elucidated.

1. The switch of  $D0$  is encoded into  $W1$ .



2. The switch of  $D1$  is encoded into  $W2$ .
3. The switch of  $D0$  and  $D1$  is encoded into  $W3$  (with  $W1 < W2 < W3$ ).
4. Without any input switching, there are no changes in the PWM wire.

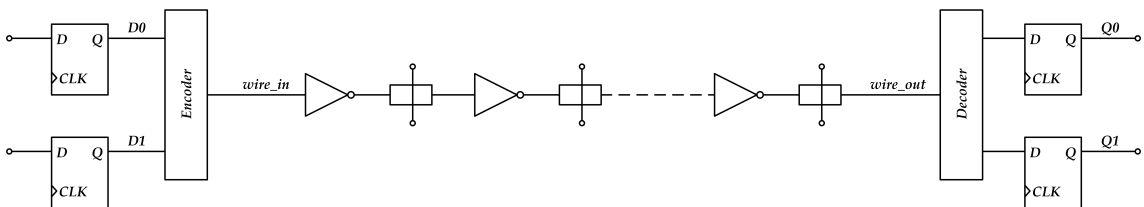
This concept can be further understood regarding a simplified PWM communication timing diagram, shown in Figure 3.2.



**Figure 3.2:** PWM encoding timing diagram (2-bit bus). For instance,  $W2$  pulse width is generated due to first  $D1$  rising edge. Then,  $W2$  is propagated through the wire (from  $wire\_in$  to  $wire\_out$ ) and decoded again. Finally, first  $Q1$  rising edge is generated.

The wire stays unchanged while the input is not switching. The delay between  $wire\_in$  and  $wire\_out$  signals is caused by the wire propagation delay. In conclusion, the information resides in the switches of the inputs, not in their current high or low state. Therefore, every switch is encoded into one of the three different pulse widths ( $W1$ ,  $W2$ ,  $W3$ ).

A general view of the system is represented in Figure 3.3, which also shows the circuit location of previous signals.



**Figure 3.3:** General view of the PWM system.

The Encoder-Decoder illustrated represents combinational circuitry. Naturally, the Encoder-Decoder scheme have an area and delay overhead. However, in following

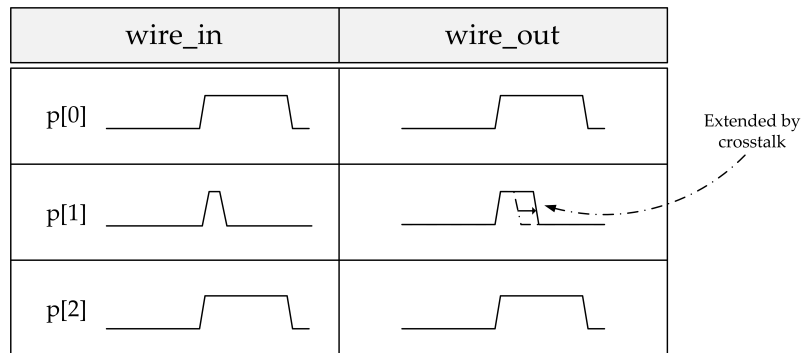
sections, this approach is proven to be more power efficient than the usual digital full-swing signaling scheme, with a higher performance.

### 3.2 Crosstalk Effect in PWM Signals

The pulse width contains information, as shown in the previous section. Hence, any variation on its width, can easily affect to the decoding process. Scaling technologies reduce the width and separation between metal layers. This fact leads to an increase of the interwire coupling capacitances, aggravating the crosstalk phenomenon. As a result, in order to adapt the system to pulse width variation, coupling capacitances and crosstalk effect are taken into account in the design. Crosstalk behavior, in this design, is based on the following considerations.

- Surrounding wires are ideally PWM-based buses.
- Crosstalk can affect the wire, which has activity in the vicinity (rising or falling edges).
- Crosstalk cannot affect when a rising edge is synchronized with the one in neighboring wires.
- PWM encoders are designed to vary pulse widths, but keeping rising edge starting-time equal for all the neighboring wires.
- PWM encoders are designed to vary falling edge time, creating a pulse width modulation effect.

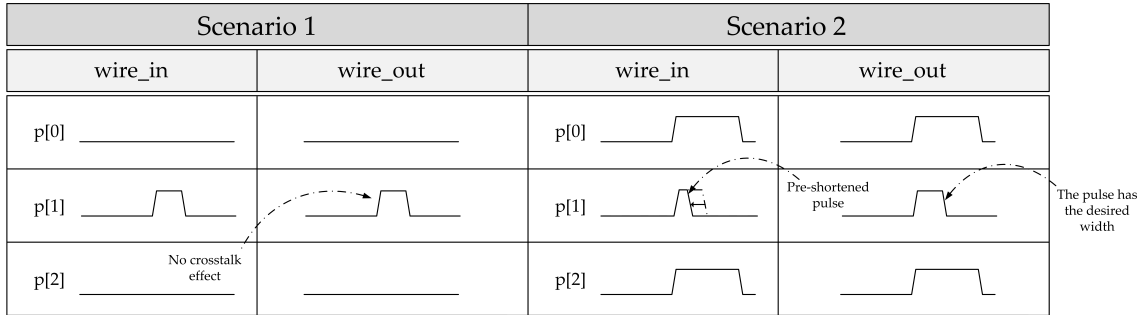
Therefore, in this design, the falling edge is only affected by crosstalk. Shortened or lengthened pulses are consequence of crosstalk, as illustrated in Figure 3.4.



**Figure 3.4:** Crosstalk effect in PWM signals. In this case,  $p[1]$  is lengthened by surrounding wires, which are transmitting wider pulses.

In Figure 3.4,  $p[i]$  represents a pulse generated in  $wire[i]$ . Crosstalk is an important problem in long wires, especially with a PWM encoding, where pulse width is determinant. Therefore, in order to increase the noise margin, crosstalk-aware circuitry is included in the design [10].

In this approach, lengthened or shortened pulses are consequence of crosstalk. Therefore, crosstalk effect can be simply avoided pre-shortening or pre-lengthening a pulse, depending on the neighboring combination. This concept is illustrated in Figure 3.5.



**Figure 3.5:** Crosstalk avoidance concept. In both scenarios,  $p[1]$  remains unchanged at the wire output ( $wire\_out$ ).

A pulse width is shortened in case surrounding wires pulses are wider and vice versa, remaining an unchanged output. This concept is represented in  $p[1]$ , illustrated in Figure 3.5.

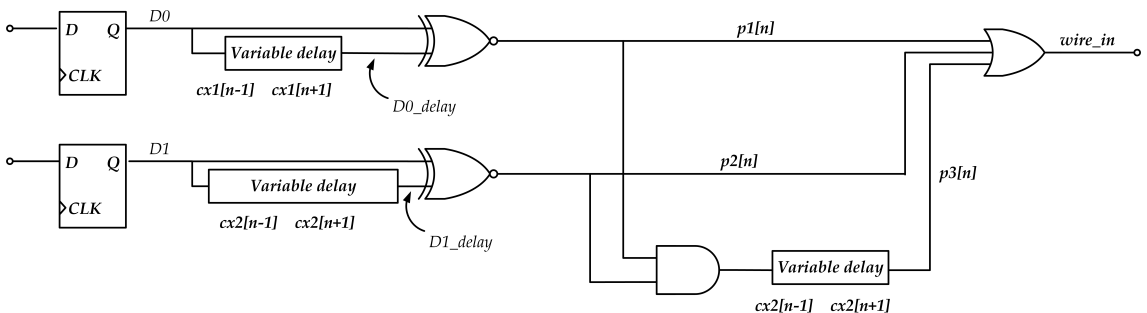
This technique is based on a pre-correction circuitry (a pulse is modified before transmitting it). Design can be further improved with post-correction circuitry [10]. In order to fine adjust crosstalk-aware circuitry, either an extracted simulation or a simulation including the wire model explained in Section 2.3, can be used.

### 3.3 Encoder-Decoder Circuits

This approach is based on three sub-circuits: Encoder, Decoder and Crosstalk-aware Circuitry. In the following section, these three components are described.

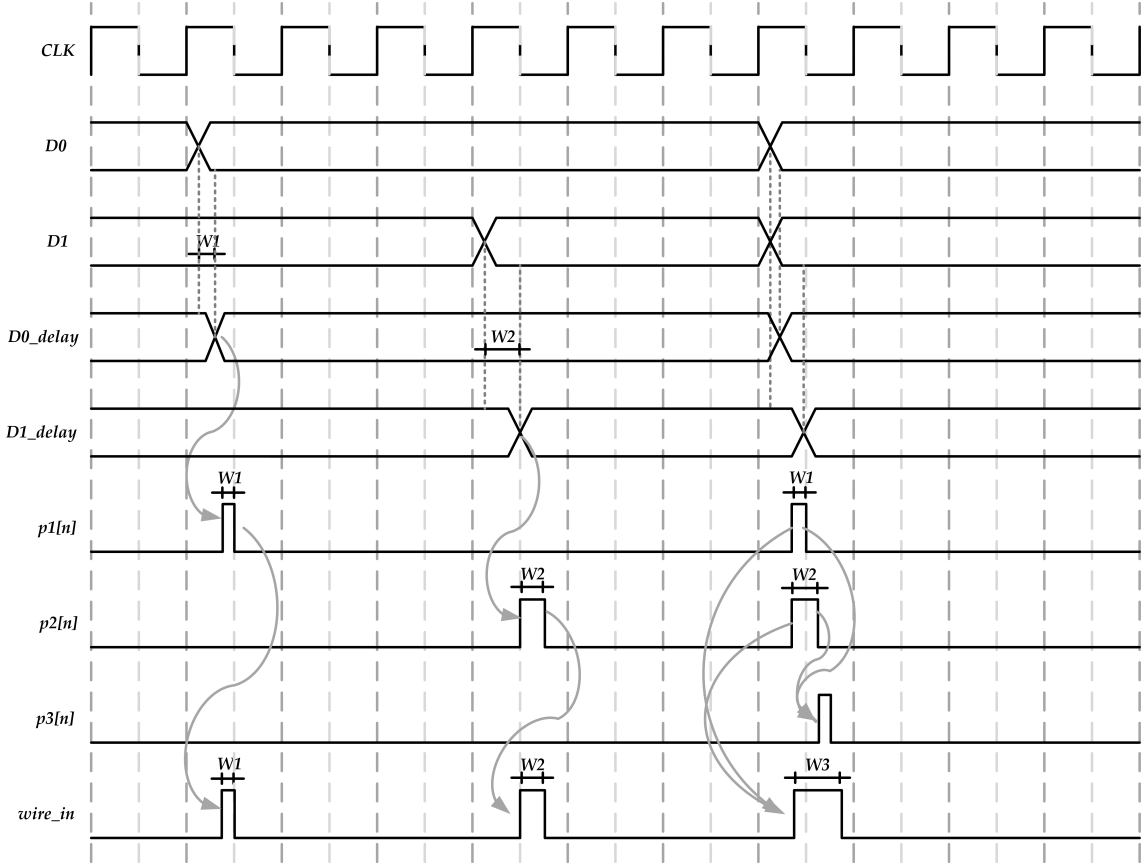
#### 3.3.1 Encoder

First, the encoder scheme, which is in charge of doing the digital-to-PWM conversion, is shown in Figure 3.6.



**Figure 3.6:** PWM encoder scheme.

Encoder functionality is best illustrated in the following figure (Figure 3.7). Three types of encoding pulses are shown. Every plausible input combination is encoded into  $W1$ ,  $W2$  and  $W3$  pulse width. The crosstalk control signals ( $cxi[n \pm 1]$ ) are explained in Section 3.3.3.



**Figure 3.7:** Encoder functionality timing diagram. Delay time among  $D_i$  and  $D_i\_delay$  generates the desired pulse width ( $W1$  and  $W2$ ). An AND-Delay operation between  $p1[n]$  and  $p2[n]$  generates  $W3$  pulse width (signals location in Figure 3.6).

Encoder functionality is summarized in the following enumeration.

- A switch of either  $D0$  or  $D1$  signals generates a  $p1[n]$  or  $p2[n]$  pulse, respectively.
- When  $D0$  is the only one switching,  $wire\_in$  becomes equal to  $p1[n]$  (same situation whether  $D1$  is the only one switching, with  $p2[n]$ ).
- When both  $D0$  and  $D1$  are switching, a third delayed pulse ( $p3[n]$ ) is generated and ORed to the previous  $p1[n]$  and  $p2[n]$  pulses, creating a wider pulse at the output.

The delay value of *variable delay* structures is controlled by  $cxi[n \pm 1]$ , as explained further on. This scheme implements the behavior described in Figure 3.1. Now, any bit pattern at the input is correctly encoded into a PWM signal.

### 3.3.2 Decoder

Decoder scheme is illustrated in following Figure 3.8. The functionality of the decoder is simple: measures the length of incoming pulse, and re-generates the correct  $D0$  and  $D1$  pattern at the output.

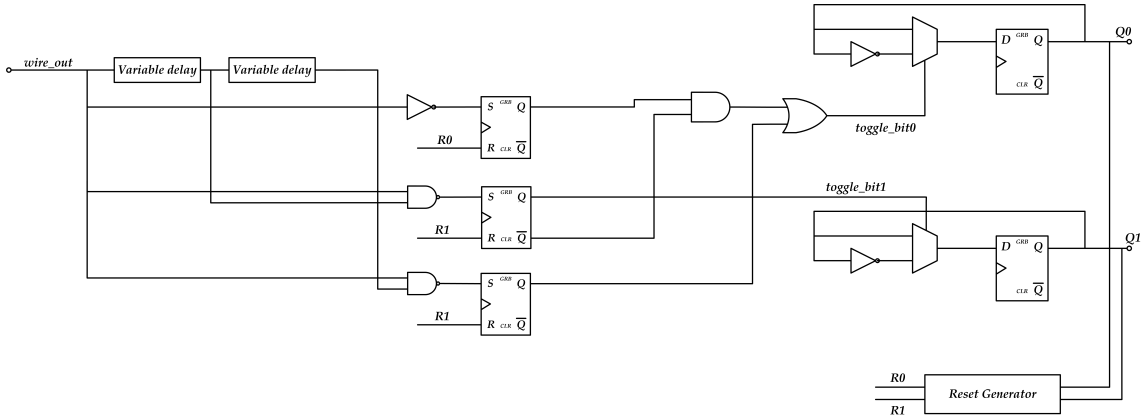
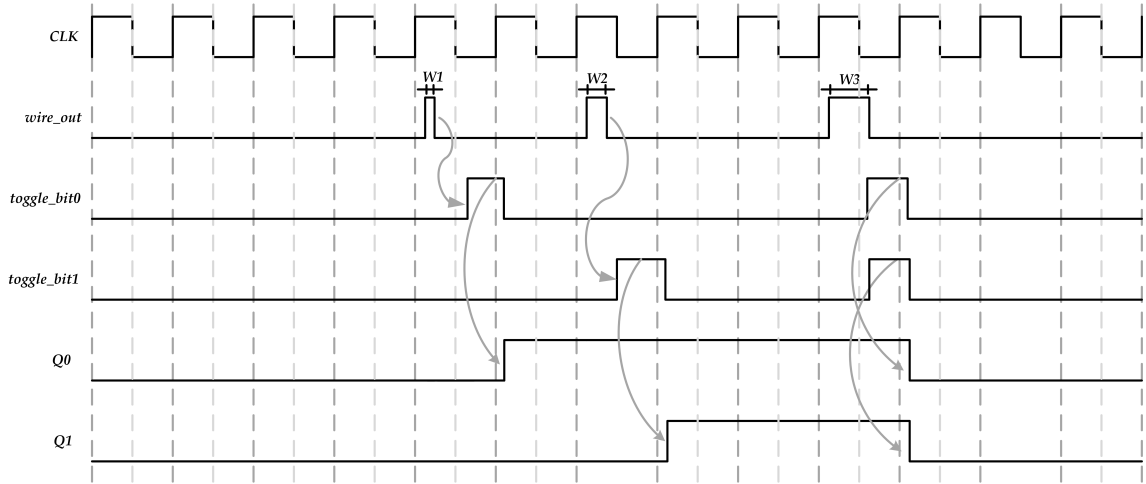


Figure 3.8: PWM decoder scheme.

Decoder functionality is summarized in the following enumeration.

- A pulse signal which comes into *wire\_out* port is delayed twice, by the first *variable delay* and the second *variable delay*.
- The delay time of each variable delay corresponds to  $W1$  and  $W2$  width, based on the encoder design.
- High-state *toggle\_bit*[ $i$ ] makes output  $Q[i]$  to switch from the previous state (from high to low or vice versa).
- When the incoming pulse width is  $W1$ , *toggle\_bit0* changes into high state but not *toggle\_bit1* (and vice versa, with  $W2$ ).
- When the pulse width is  $W3$ , both *toggle\_bit0* and *toggle\_bit1* change into high state.

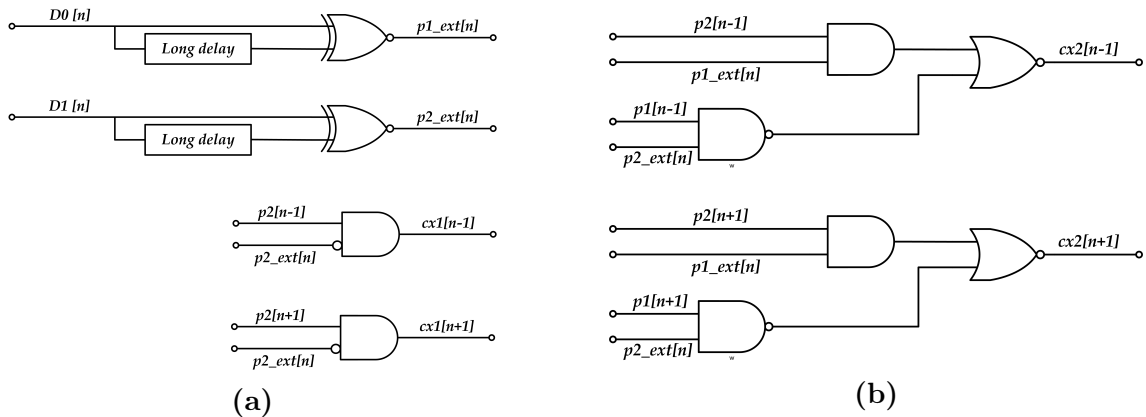
Description above is best represented in the following timing diagram (Figure 3.9). Every possible decoding scenario is represented. Incoming pulses cause switches on the output signals. Therefore, is important to correctly reset the system before transmitting, having it in a known state.



**Figure 3.9:** Decoder functionality timing diagram. For instance,  $W1$  pulse width rises  $toggle\_bit0$ , which makes the output  $Q0$  to switch.

### 3.3.3 Crosstalk-Aware Circuitry

Overall bus performance is enhanced by crosstalk-aware circuitry. It is constituted by a set of combinational components, which are described in this subsection. All of the crosstalk-aware circuit units are illustrated in Figure 3.10.



**Figure 3.10:** Crosstalk-aware circuitry.

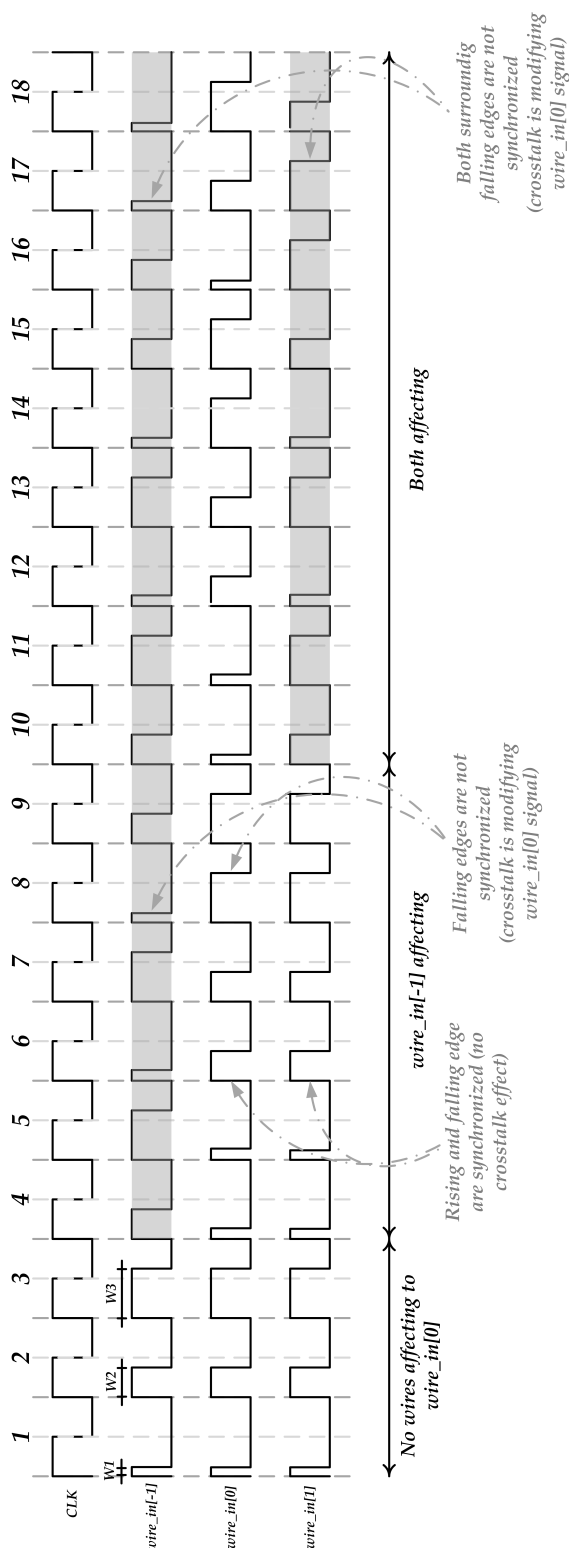
Crosstalk-aware circuit functionality is summarized in the following enumeration.

- $[n]$  index represents the main wire, which is seen as the one affected by the surroundings.
- $[n + 1]$  and  $[n - 1]$  indexes represent the neighboring wires.
- An extended version of  $p1[n]$  and  $p2[n]$  (Subsection 3.3.1) is represented by  $p1\_ext[n]$  and  $p2\_ext[n]$ , respectively.

- Using current states of the the surrounding wires pulses ( $p1[n - 1]$ ,  $p2[n - 1]$ ,  $p1[n + 1]$  and  $p2[n + 1]$ ), the scheme generates  $cx1[n - 1]$ ,  $cx2[n - 1]$ ,  $cx1[n + 1]$  and  $cx2[n + 1]$ .
- $cx1[n - 1]$ ,  $cx2[n - 1]$ ,  $cx1[n + 1]$  and  $cx2[n + 1]$  indicate whether the current pulse is affected by crosstalk.
- Combination of  $cx_i[n \pm 1]$  signal states, differentiate between different crosstalk scenarios.

Extra circuitry is required to drive the input of the variable delays, in order to make it compatible to a specific design. This concept is best illustrated in Figure 3.16.

As explained, crosstalk in this approach can be approximated by a deterministic behavior. In this context, a finite number of scenarios are analyzed. With three wires, affected by crosstalk, the possible scenarios are fixed to 15. This is best illustrated in Figure 3.11.

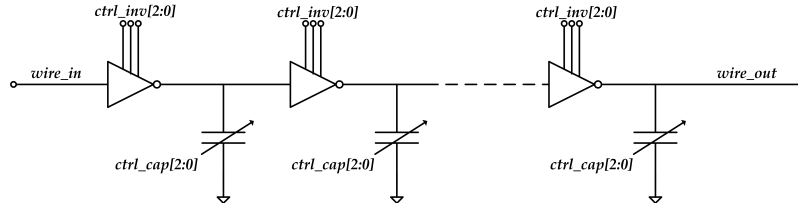


**Figure 3.11:** Possible crosstalk scenarios (3 wires configuration). Victim wire is represented by *wire\_in[0]*. In the three first cases, where the wires propagate the same pulse width, there is no crosstalk effect (scenarios from 1 to 3). Next, where *wire\_in[-1]* propagates different pulse widths (comparing to *wire\_in[0]* and *wire\_in[1]*), crosstalk starts to affect (scenarios from 4 to 9). Finally, crosstalk fully affects in scenarios from 5 to 18, where both *wire\_in[-1]* and *wire\_in[1]* propagates different pulse widths. Differences in pulse widths between a wire and *wire\_in[0]* are highlighted by a grey background.



Hence, simulating these scenarios and measuring how each one affects to the actual wire ( $wire\_out[0]$ ), crosstalk influence can be easily characterized. After, crosstalk-aware circuitry is accurately calibrated.

The *variable delay* scheme is shown in Figure 3.12. The inverters represented in Figure 3.12 have a variable drive strength, controlled by  $ctrl\_inv[2:0]$ . Variable capacitances can vary their capacitance value depending on control signals  $ctrl\_cap[2:0]$ .



**Figure 3.12:** Variable Delay schematic.

Varying the amount of *variable inverter - variable capacitance* pairs, the required nominal delay is obtained. Also, in order to decrease or increase the pulse width (crosstalk pre-correction concept),  $ctrl\_inv$  and  $ctrl\_cap$  are adapted to  $cx1$  and  $cx2$  signals, as explained further on.

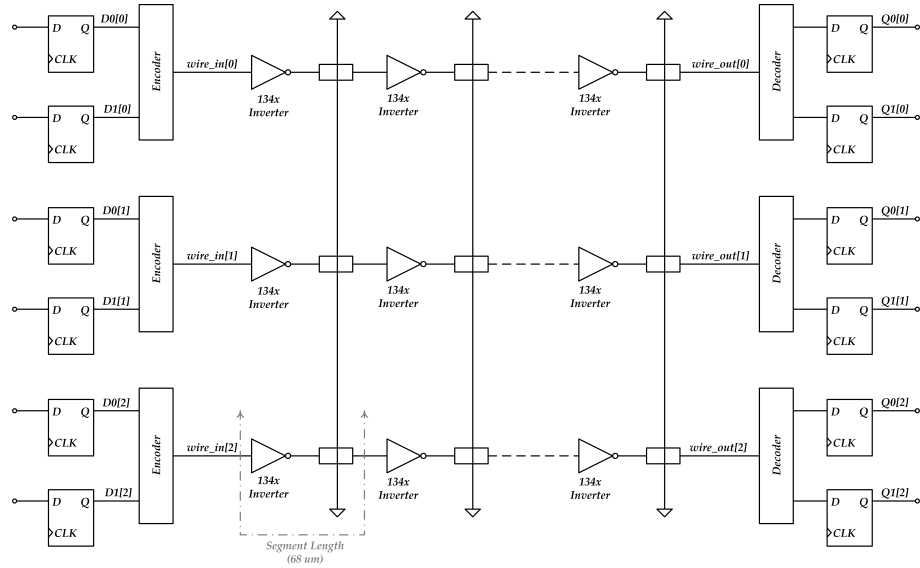
Finally, the implementation of the remaining circuits such as the *variable capacitances*, *weight inverters* and *Reset Generator* is shown in the Appendix M. The complete system schematic can also be found in the Appendix D.

## 3.4 Simulation and Results

The procedure to carry out simulations as well as the results, are presented in this section. All schematic simulations have been realized by *Eldo (Mentor Graphics)* software.

### 3.4.1 Simulation

Since this thesis is comparing two bus design techniques, a common testbench for both approaches is needed, in order to be able to compare them. This test bench consists on a crosstalk wire model (Section 2.3) to which either encoder and decoder are connected. Final configuration is illustrated in Figure 3.13. Crosstalk-aware circuitry remains implicit in the following figure. However, mentioned circuitry is included in the actual design.



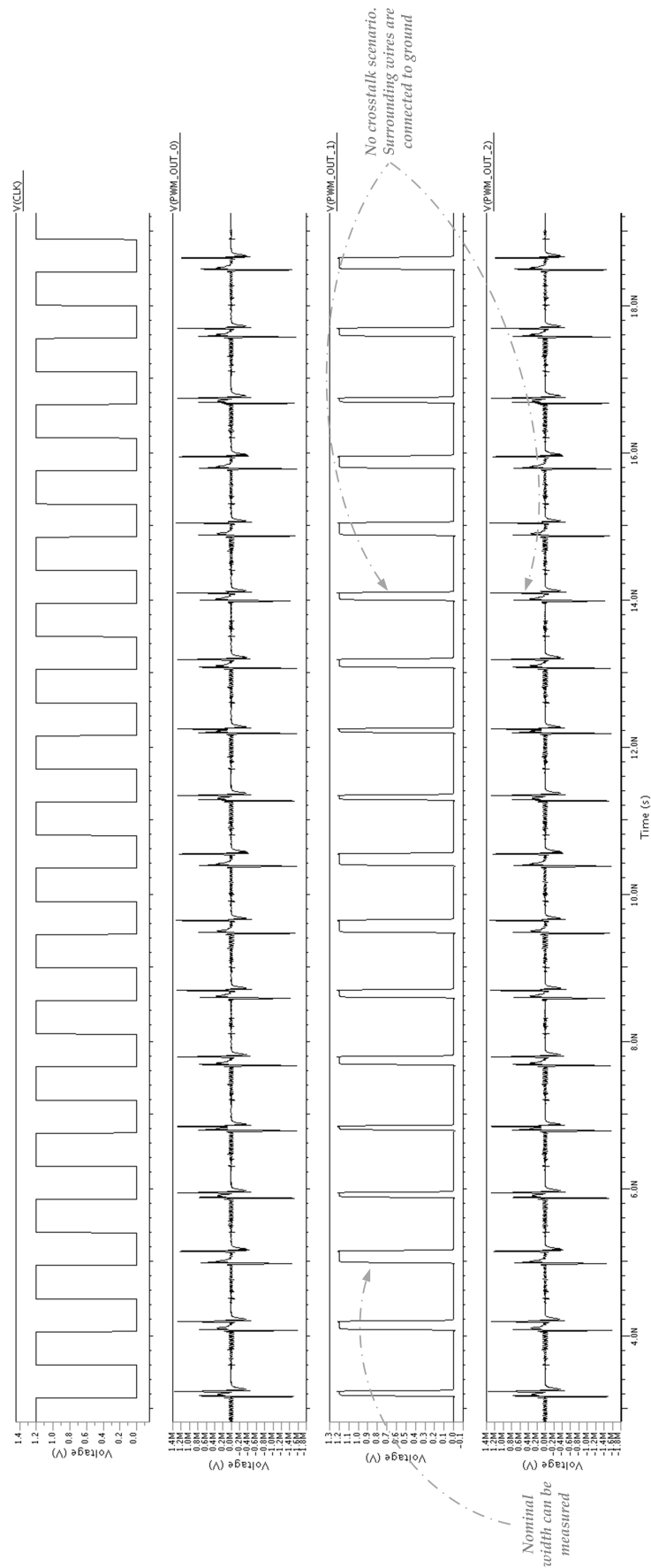
**Figure 3.13:** Complete testbench schematic for the PWM-based bus.

This schematic has been implemented regarding results in Table 2.5. In this schematic,  $wire[0]$  and  $wire[2]$  neighboring wires are also assumed to be connected to ground, in order to simplify simulations. Neighbors of neighbor wires influence can be neglected.

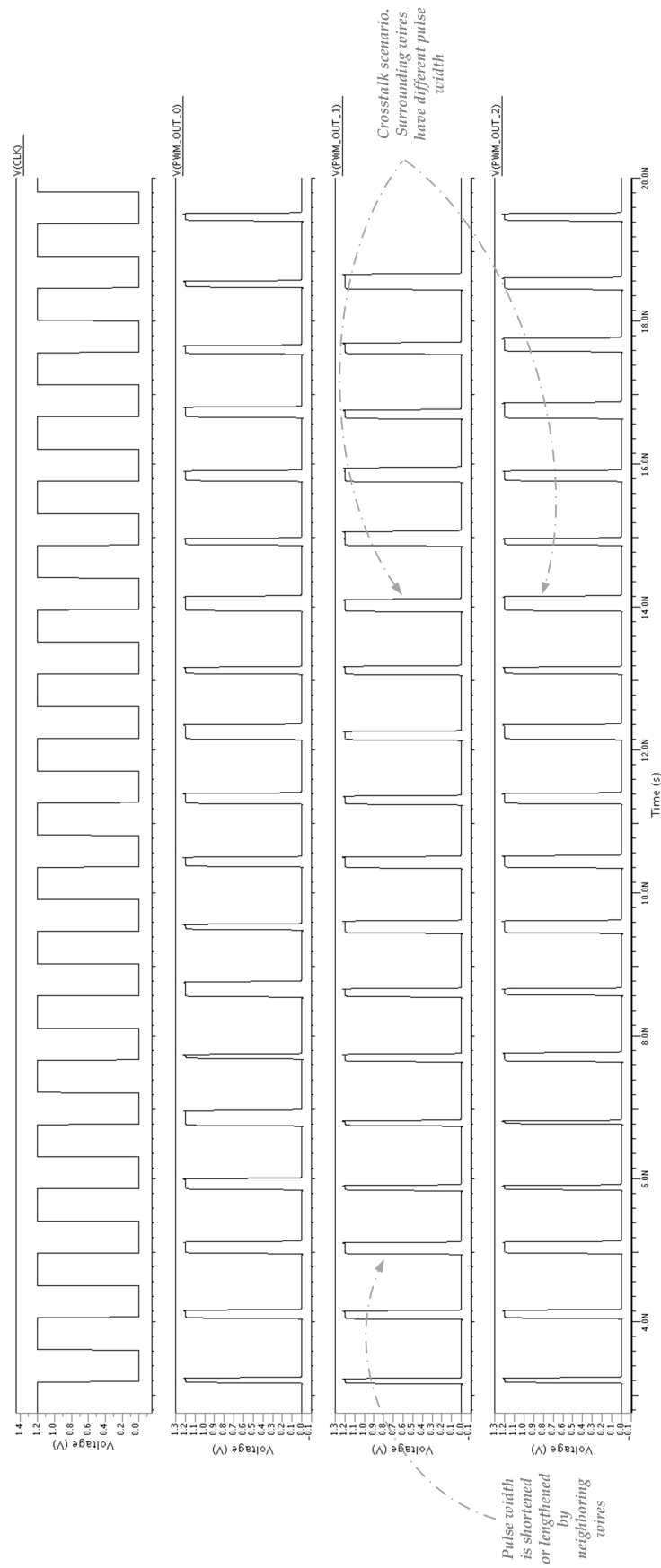
Before proceeding to the bus testing, crosstalk-aware circuitry has to be calibrated, taking advantage of the none-switching neighbors scenario. In this situation,  $wire[1]$  is not affected by crosstalk.

Then, current width of each pulse is measured ( $wire[1]$ ), corresponding to the non-crosstalk width results (nominal pulse width). This is best represented in the following simulation diagram (Figure 3.14). Where  $PWM\_OUT[i]$  is equal to  $wire\_out[i]$  of Figure 3.13 and  $CLK$  represents the clock signal.

In addition, a similar simulation is carried out, with activity in neighboring buses. Crosstalk effect on this design is quantitatively determined, regarding the following simulation diagram (Figure 3.15).



**Figure 3.14:** Simulation of the wire outputs. No Crosstalk is affecting  $PWM\_OUT[1]$ . Noise represented in  $PWM\_OUT[0]$  and  $PWM\_OUT[2]$  corresponds to a 0V value (unchanged wire). The nominal value of the pulse widths is measured.

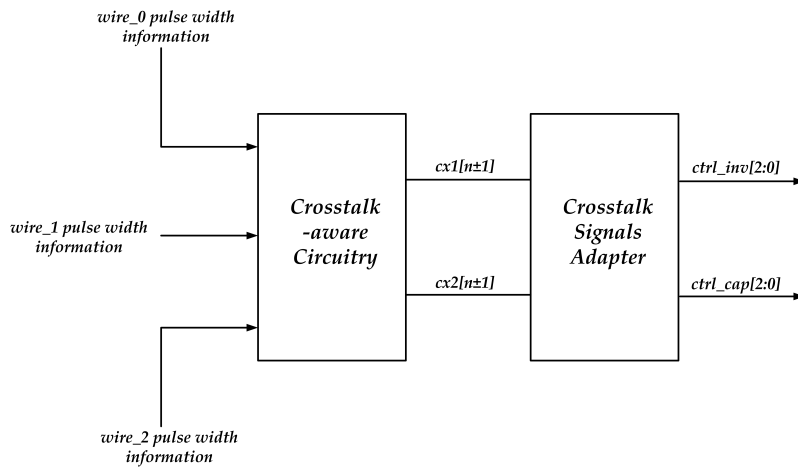


**Figure 3.15:** Simulation of the wire outputs. Crosstalk is affecting  $PWM\_OUT[1]$ . This simulation represents all of the possible scenarios illustrated in Figure 3.11. In the scenarios where crosstalk is affecting  $PWM\_OUT[1]$ , the time that a pulse is shortened or lengthened is measured. Naturally, there is no crosstalk-aware circuitry active.

In Figure 3.15, pulse widths in  $PWM\_OUT[1]$  differ from those shown in Figure 3.14. Differences in  $PWM\_OUT[1]$  between Figure 3.14 and Figure 3.15 are measured, in order to obtain quantitative crosstalk data. Therefore, regarding the pulse width variation among crosstalk and non-crosstalk scenarios, crosstalk effect is quantitatively determined and crosstalk-aware circuitry can be calibrated.

As an example, if a pulse is  $Nps$  lengthened by crosstalk, then crosstalk-aware circuitry must pre-shorten it (before driving the wire) the same  $Nps$  amount, and vice versa. Consequently, 15 possible scenarios (Figure 3.11), in terms of pulse width variation, are taken into consideration by crosstalk-aware circuitry.

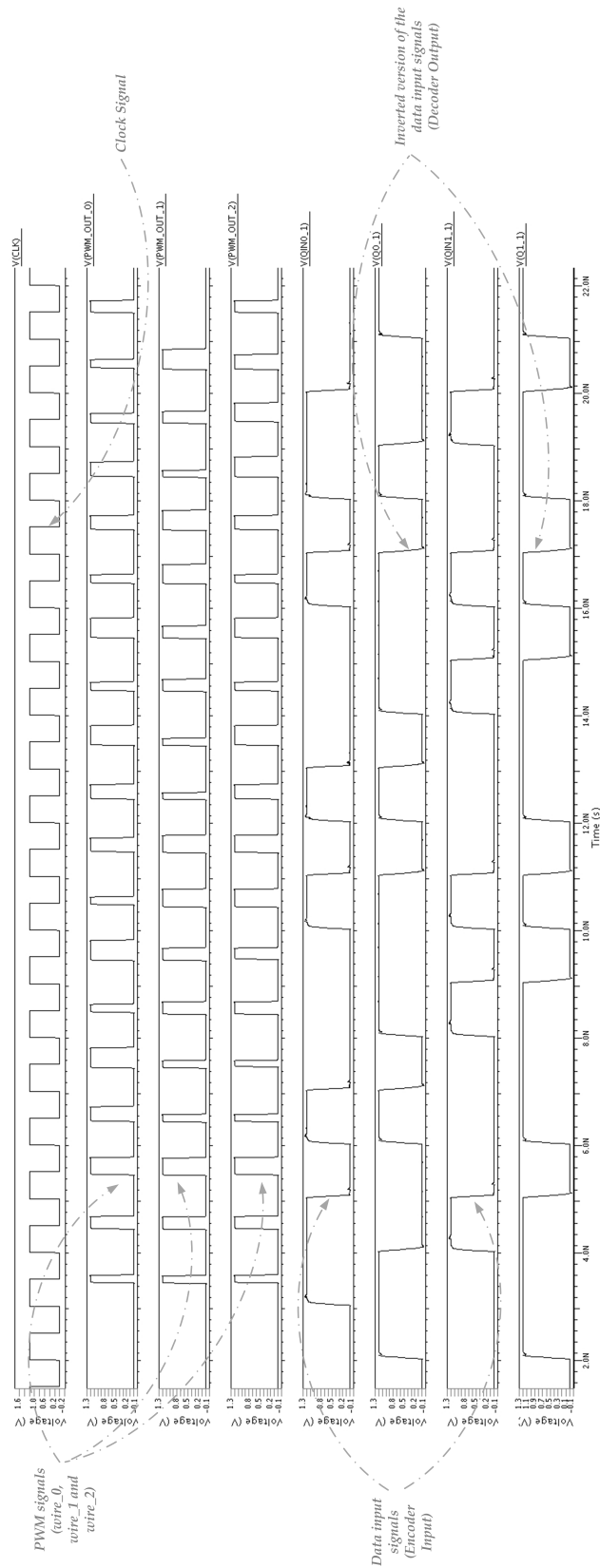
The signals  $cx1[n]$  and  $cx2[n]$  are required to provide previous functionality. Using these signals, as well as previous measured crosstalk information, circuitry which adapts  $cx1[n]-cx2[n]$  to  $ctrl\_inv[2:0]-ctrl\_cap[2:0]$  (Figure 3.12) can be implemented. This is best illustrated in Figure 3.16.



**Figure 3.16:** Block diagram of crosstalk signals adaptation circuitry.

The delay of the complete crosstalk signaling is determinant while improving overall system performance. Crosstalk signals adapter is a technology and design dependent circuit which varies depending on the *variable delay* implementation. This adapter is composed by a simple set of combinational logic. Therefore, its fully description is avoided in this work. However, its existence and necessity is consequently highlighted.

Finally, crosstalk pre-correction mechanism is fully implemented and calibrated. Final simulation of the full system (*TT corner*) at  $27^\circ$  degrees is shown in Figure 3.17.



**Figure 3.17:** Simulation of the Decoder output. Crosstalk is affecting every wire. This figure shows the correct functionality of the Encoder-Decoder scheme.

As shown, both  $Q0[1]$  and  $Q1[1]$  bus outputs are an inverted version of  $D0[1]$  and  $D1[1]$  inputs. System has to be preset, in order to avoid this 'inverter behavior'. Also,  $PWM\_OUT[0]$ ,  $PWM\_OUT[1]$  and  $PWM\_OUT[2]$  represent the PWM signal at the wire output.

### 3.4.2 Results

Several simulations have been carried out, in order to determine minimum pulse width, as well as maximum working frequency (including total wire model). It is important to highlight the approximated nature of these results, where the wire model is not as precise as an extracted simulation. However, for a designer who prefers to pre-test a digital system before going through the layouts, this methodology is completely suitable. Optimum results of this PWM-based technique, are shown in Table 3.1.

PWM-based Simulation Results			
Results	Value	Unit	Simulation Comments
Short Pulse Width ( $W1$ )	130	$pS$	Active crosstalk-aware circuitry
Medium Pulse Width ( $W2$ )	230	$pS$	Active crosstalk-aware circuitry
Wide Pulse Width ( $W3$ )	330	$pS$	Active crosstalk-aware circuitry
Wire Length	2.04	$mm$	--
Max. Clock Frequency	1.0	$GHz$	Active crosstalk-aware circuitry / 2.04 mm link / 1.2 V / TT-FF-SF-FS corners / 80°C
Min. Voltage Supply	1.1	$V$	Active crosstalk-aware circuitry / 6-bit bus / 18-bit pattern transmission / 18 ns simulation time / 1GHz / 2.04 mm link / TT corner / 27°C
Energy/Bit ( $E_b$ )	1.269	$pJ/b$	Active crosstalk-aware circuitry / 6-bit bus / 18-bit pattern transmission / 18 ns simulation time / 1GHz@1.2V / 2.04 mm link / TT corner / 27°C
Average Total Power Consumption ( $P_{tot}$ )	7.6161	$mW$	Active crosstalk-aware circuitry / 6-bit bus / 18-bit pattern transmission / 18 ns simulation time / 1GHz@1.2V / 2.04 mm link / TT corner / 27°C

**Table 3.1:** Simulation results table. Global optimized parameters of the PWM-based approach.

$W1$ ,  $W2$  and  $W3$  are result of several simulations (*corners and Monte Carlo*). Crosstalk-aware circuitry is determinant while reducing pulse width in the global system, since pre-shortening a pulse requires time between *variable delay* control signals are generated and those signals have a real effect on the pulse. As a consequence, in order to minimize  $W[i]$ , an optimization of the *variable delay* and/or *Crosstalk-aware circuitry* is required, as a first approximation. Other timing optimizations can be achieved in both Encoder or Decoder schemes.

It is possible to optimize maximum clock frequency by decreasing pulse widths, improving simultaneously crosstalk-aware circuitry performance. This improvement would reduce the encoder-decoder delay overhead and/or optimize wire energy-delay. However, a global parameters optimization leads to fine adjustments, which requires a deep knowledge of this technique.

As shown, this system is highly sensitive to voltage drops. Any variation on the supply voltage incurs in slopes alteration. As a result, the pulse width changes and overlaps other pulse widths, leading to system errors. To solve these issues, other modules can be added to re-calibrate the system after certain time periods. A self-calibration module is included in the original version of this approach [10]. The approach described in this thesis does not cover any self-calibration module, since it is a system level module, not a bus level technique.

Both  $E_b$  and  $P_{tot}$  can be optimized while optimizing pulse widths. Equation 4.2 highlights the three main components of the total power consumption.

$$P_{tot} = P_{encoder} + P_{wire} + P_{decoder} \quad (3.1)$$

Encoder-decoder power contribution can be neglected in long wires, where the overall system becomes more efficient. In addition, it is important to emphasize that all of these results are completely wire-length dependent. Wire increases the delay, power consumption (repeaters) and distorts the signal. Consequently, variations in the wire length and/or width and re-optimizations of the wire segment and the number of repeaters, can lead to improvements in performance.

Finally, in order to provide a global vision of this technique, following chapters include a benchmark. Also, the benefits and disadvantages of this approach is reviewed in conclusions.



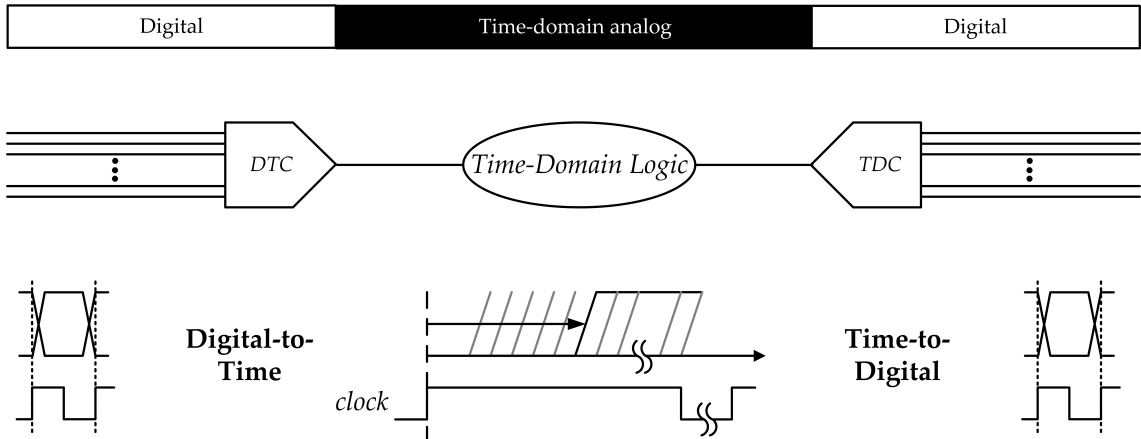
# Chapter 4

## Time-Domain Analog Conversion Bus Technique

This chapter presents the second low-power and high-fan-out bus design technique included in this thesis. The *Time-Domain Conversion* concept is introduced, as well as encoding-decoding schemes, explaining their functionality. Finally, performed simulations and results are shown, ending with conclusions and future work. All of the work in this chapter represents a nanoscale FDSOI process technology adaptation of the Time-Domain analog and digital mixed-signal processing (TD-AMS) bus design included in [13].

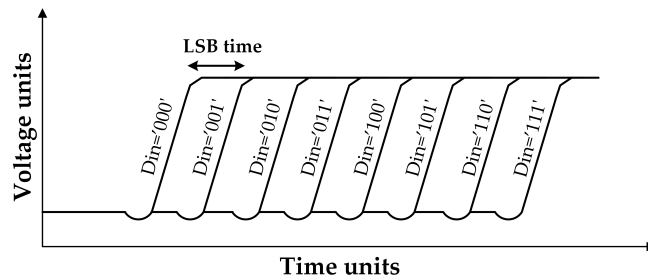
### 4.1 Time-Domain Signaling Concept

*Time-Domain Conversion* is based on a variable delay introduced to a square signal in a line. Figure 4.1 describes this time conversion technique.



**Figure 4.1:** TDC encoding technique (N-bit bus).

The rising edge is delayed, depending on  $Din[n]$  bus input. The available delay range covers one clock cycle. Thus, fixing the time step between different delay values, as well as the  $Din[n]$  bus input width, minimum clock cycle is consequently fixed. Figure 4.2 shows this concept.



**Figure 4.2:** TDC encoding technique (3-bit bus).

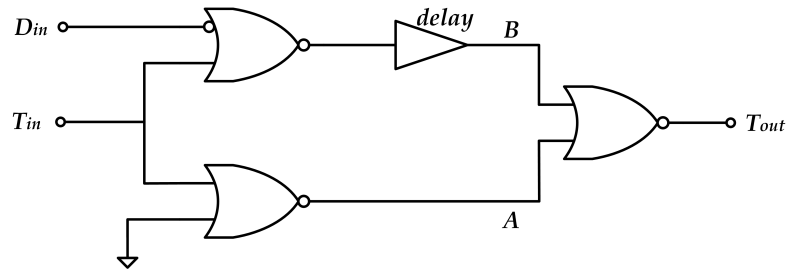
In Figure 4.2, each bit combination is encoded into a delay value. Higher binary values correspond to a higher delay value. Also, each consecutive bus state is encoded with a difference of a LSB time step, determined in design time. As can be inferred, this approach reduce drastically the number of wires in an interconnect, being perfectly suitable for a high-fan-out interconnection. However, a number of wires reduction also decreases overall frequency, since the number of steps needed increases exponentially. These issues are further explained in following sections.

## 4.2 Encoder-Decoder Circuits

This technique requires both Encoder-Decoder schemes, just as the previous PWM-based approach.

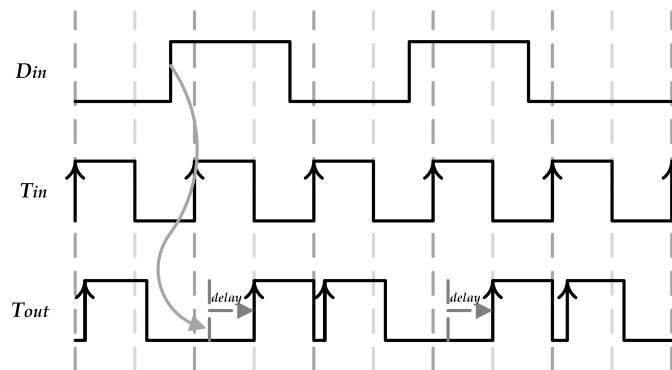
### 4.2.1 Encoder

First, *Digital-to-Time Converter (DTC)* is illustrated in Figure 4.3, which composes the encoder basic cell.



**Figure 4.3:** DTC basic cell (encoder component).

The *DTC* module is combinational, with  $D_{in}$  and  $T_{in}$  as the inputs and  $T_{out}$  as the output.  $T_{in}$  represents an incoming rising edge and  $D_{in}$  is the 1-bit data input. *DTC* behavior is fully described in Figure 4.4.

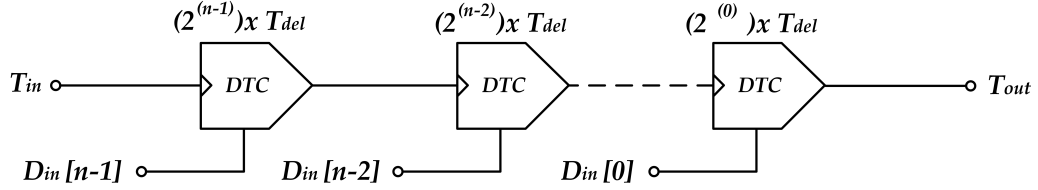


**Figure 4.4:** DTC timing diagram.

In Figure 4.4,  $T_{out}$  is delayed '*delay*' seconds, depending on whether  $D_{in}$  is in a high state. When  $D_{in}$  is low,  $T_{in}$  propagates with the minimum combinational delay

of the structure. Using this module, the delay of a rising edge is easily controlled by  $D_{in}$  signal.

Therefore, a chain of  $DTCs$  is made, in order to control a wider range of delay possibilities, creating the encoder scheme. Figure 4.5 illustrates a generic Time-Domain Encoder scheme.

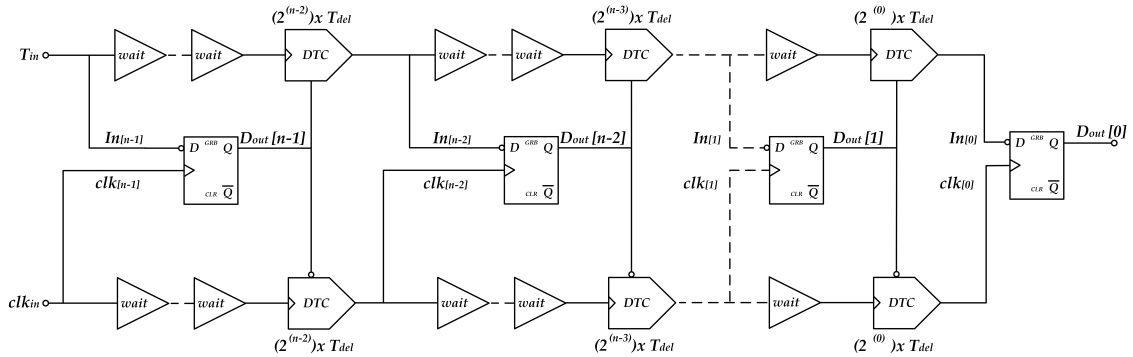


**Figure 4.5:** Time-domain Analog Encoder circuit diagram.

Where  $T_{del}$  represents the LSB time step value. Implementing an  $n$ -bit bus, a chain of  $DTCs$  is needed, with an exponentially decreasing  $DTC$  delay value ( $T_{del}$ ), in order to generate a similar pattern to previous Figure 4.2. Introducing a square signal into the  $T_{in}$  input port,  $D_{in}[n-1:0]$  is encoded to time-domain ( $T_{out}$ ).

#### 4.2.2 Decoder

The decoder scheme has a higher level of complexity, resulting in a sequential structure. Figure 4.6 shows a circuit diagram of the binary-search *Time-to-Digital Converter* ( $TDC$ ).



**Figure 4.6:** Time-domain Analog Decoder circuit diagram.

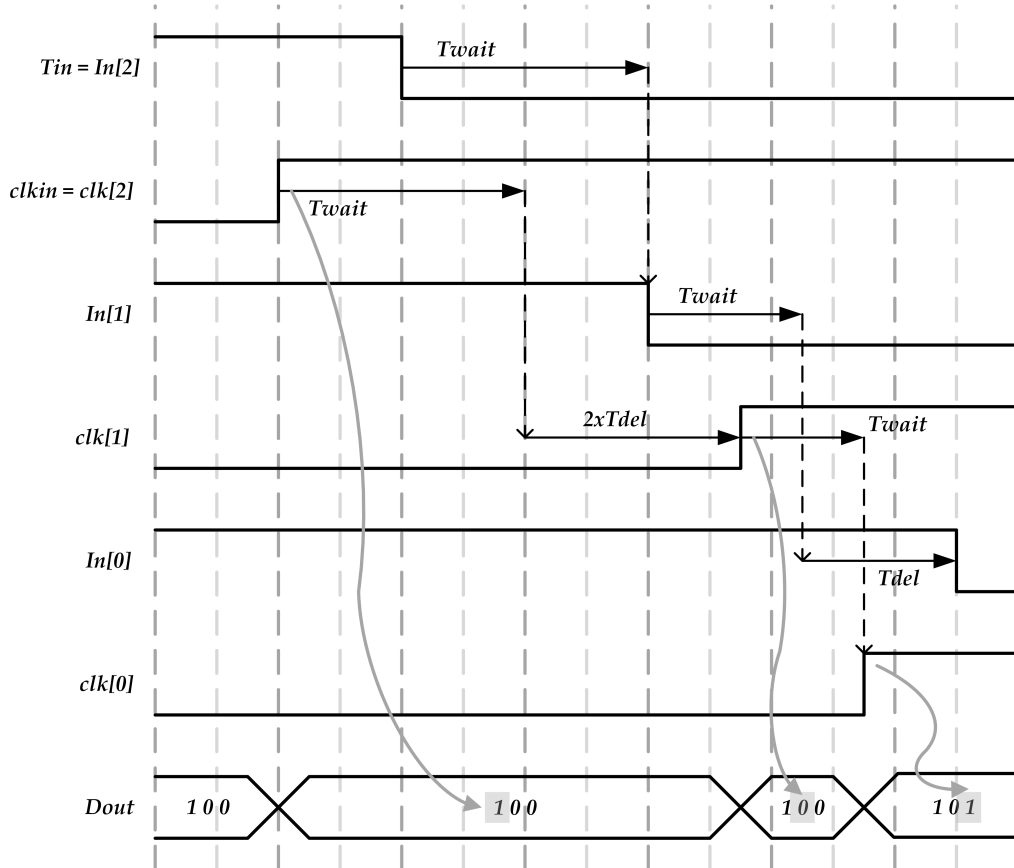
In Figure 4.6, shown scheme implements a binary-search algorithm, decoding each bit separately. Binary-search algorithm is further understood regarding the timing diagram represented in Figure 4.7.

Regarding the following enumeration, mentioned diagram is elucidated.

- $clk[i]$  signal samples  $In[i]$  signal.
- Encoded time value makes  $In[i]$  falling edge pass  $clk[i]$  rising edge, which results in a '1' sample.

- The opposite situation can happen, when  $In[i]$  falling edge occurs before  $clk[i]$  rising edge, which results in a '0' sample.

This design shows a robust behavior, since the relative time-distance between  $clk[i]$  and  $In[i]$  is dynamically adjusted by the binary-search algorithm. As shown in the mentioned timing diagram, each stage of the circuit diagram (Figure 4.6) implements one stage of the binary-search algorithm, decoding its corresponding bit.



**Figure 4.7:** Binary-Search Algorithm timing diagram (2-bit TDC). The  $clk[i]$  signal samples  $in[i]$ . The decoder scheme has 3 stages, where each bit is decoded separately.

Although this thesis is targeting basic understanding and characterization of these bus approaches, re-combining some components can provide more functionality, suitable for certain applications. As an example, in HNNs the time-domain conversion technique is completely convenient, since the number of crossed interconnections is voluminous. Furthermore, this TDC approach provides additional advantages. For instance, some operations, as minimum calculation, constant coefficient multiplication or summation, are carried out faster in the time-domain with less power consumption [13]. In addition, high reduction of the number of wires provides a better scalability, as well as an area diminution.

### 4.3 Synthesizable Verilog Description

TDC-based approach is fully digital, despite of being based on a time-domain conversion. This fact implies that all the functionality is implemented by digital structures.

As a consequence, this design is compatible with the standard *EDA* flow. This TDC-based technique can be included and optimized by place-and-route tools as well as be modeled using a *High Level Hardware Description Language*, such as *Verilog* or *VHDL*.

This technique is highly scalable, any bus width can be implemented. This TDC approach was implemented in *Verilog*. Thus, setting values for Bus Width ( $N$ ) and Time Step ( $k$ ), a completely functional TDC-based bus can be synthesized.

$$LSB \text{ Time Step} = k \cdot 50 \text{ ps} \quad (4.1)$$

The minimum LSB time step is fixed to 50 *ps*. This value can be reduced in future optimizations. However, in order to hold the system completely functional a moderate value was chosen.

In conclusion, without going through the real design, a designer interested in including this bus design approach can easily synthesize, place and route an  $N$ -bit TDC-based bus. Full TDC-based *Verilog* description code is shown in Appendix H, I, J, K and L.

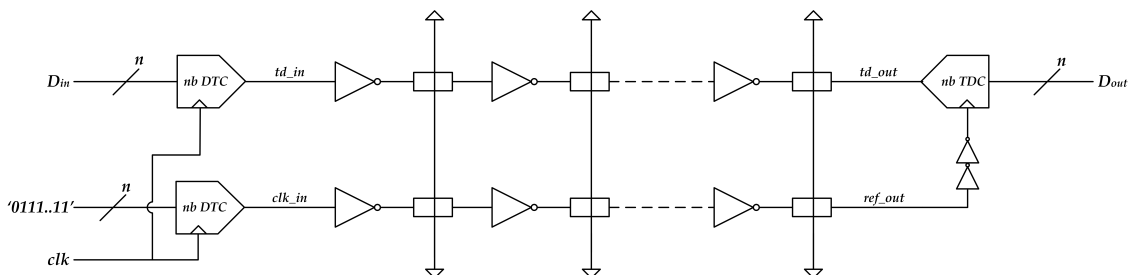
It is necessary to emphasize that previous PWM-based approach is completely digital as well. However, its *Verilog* and/or *VHDL* description is substantially more complex. Also, optimizations in some basic structures of this PWM-based technique, have to be carried out before trying to make a generic definition of the approach. Therefore, a PWM *Verilog* description remained out of target in this thesis.

### 4.4 Simulation and Results

In this section, the procedure to carry out simulations is presented, as well as results. All schematic simulations have been realized by *Eldo* (*Mentor Graphics*) software.

#### 4.4.1 Simulation

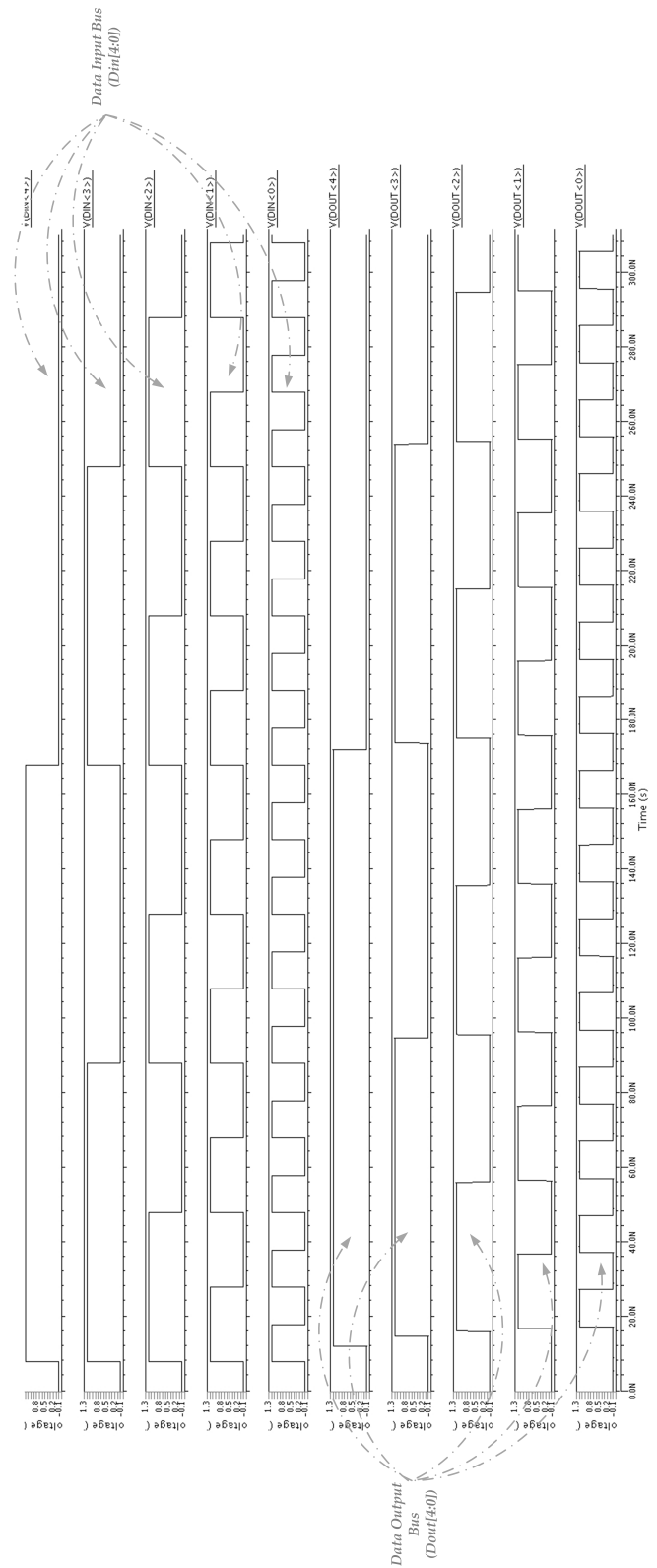
Again, since this thesis compares two bus design techniques, a common test bench for both approaches is needed, in order to be able to compare them.



**Figure 4.8:** Complete testbench schematic for the TDC-based bus.

The testbench, represented in Figure 4.8, consists on a crosstalk wire model (Section 2.3) to which encoder and decoder are connected. The testbench is composed by two Encoders, a Crosstalk Wire Model and a Decoder. The clock signal has to be propagated beside the time-domain data signal, in order to minimize *skew* and *jitter*. Also, in order to center the reference, approximately in the middle of the clock cycle, the  $n - bit$  *DTC* data input is fixed to the half of the bus size. For instance, to encode a 3-bit bus, the reference input is '011', which corresponds to the decimal 3 (half of 7). In addition, in order to meet register setup-time constraints, two inverters are included at the end of the wire model, with an overall delay of approximately  $T_{del}/2$ .

Signals diagram of a 5-bit bus encoding-decoding simulation (100MHz,  $1.2V_{supply}$ ,  $27^{\circ}\text{C@TT}$  corner,  $T_{del} = 100 ps$ ) are shown in Figure 4.9.



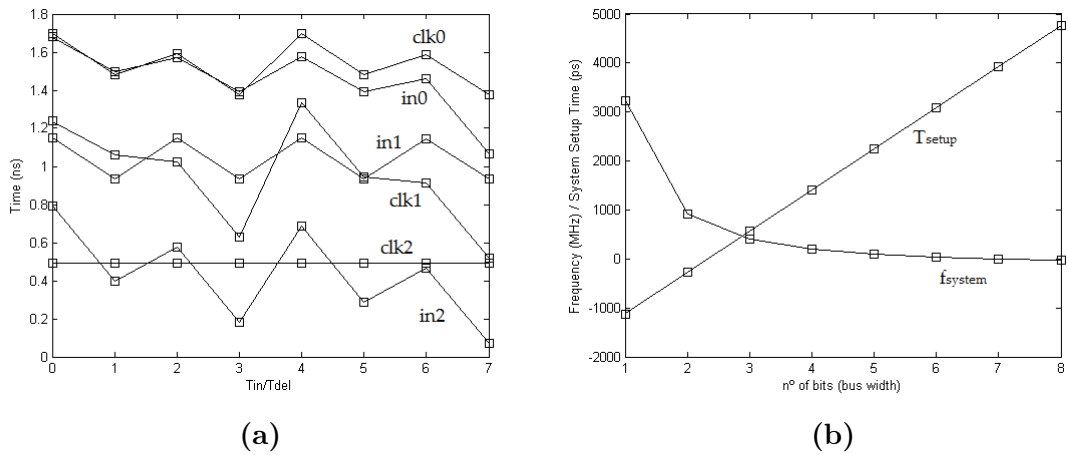
**Figure 4.9:** 5-bit bus encoding-decoding simulation (data input and data output are shown). This figure shows the correct functionality of the synthesized TDC bus. 5-bit input combination ( $DIN[i]$ ) is encoded and decoded, providing the same pattern at the output ( $DOUT[i]$ ).

Figure 4.9 illustrates the correct functionality of this TDC-based approach. Thus,  $D_{out}[4 : 0]$  is a delayed version of  $D_{in}[4 : 0]$ . Simulation shown in Figure 4.9 covers all possible cases in a 5-bit transmission.

Also, simulated sequence of bits represents worst case switching ('10000' -> '01111', '01000' -> '00111', etc.), since time distance between clock reference and time-domain input data is the shortest. This fact leads to a reduced noise margin, making the system more likely to have a bit error.

As explained previously, internal nodes of the TDC decoder circuit are independent for each bit. Each  $clk[i]$  samples each incoming time-domain data  $in[i]$ . After each sample, either  $clk[i]$  or  $in[i]$  are delayed multiples of  $T_{del}$ , depending on the sampled value (Figure 4.6).

As a consequence, according to the incoming data pattern,  $clk[i]$  and  $in[i]$  edges pass each other several times in each transmission. This is best illustrated in Figure 4.10a. In some extreme cases (*corners* and *Monte Carlo* scenarios), time-distance between them is not long enough, incurring in setup time issues and causing a system bit error. Nevertheless, a desirable design has sufficient  $T_{del}$ , as well as setup time restrictions, to avoid this inadequate behavior.



**Figure 4.10:** Figure 4.10a Simulated edge positions of the internal nodes of the 3b TDC. The time position of  $clk[i]$  and  $in[i]$  is represented for each stage of the 3-bit encoder. These 3 stages are represented for every possible scenario in a 3-bit communication (from 0 to 7). Figure 4.10b Bus frequency vs. no. of bits (bus width) and Setup Time vs. no. of bits (bus width). Frequency and Setup Time shares the magnitude order in the same axis. This fact is due to the sensitivity of this approach to Setup Time.

This approach is strongly sensitive to bus size. More precisely, an increase of 1 bit in the bus size, causes an exponential decrease in system frequency (Figure 4.10b). At the same time, system setup time (encoder setup time) increases its value exponentially, becoming more restrictive (Figure 4.10b). This loss in performance is due to the fact that each bit of the bus size increased, adds another *DTC* extra component, which includes as the minimum delay time, the next factor in a exponential series of  $T_{del}$ . Previous graph shows two fitted exponential curves to 3b, 4b and 5b bus data information (Figure 4.10b).



As a result, minimum clock period is  $> N \cdot T_{del}$ , with  $N = \text{maximum binary value of the bus}$ . In addition, minimum setup time is also higher than  $N$  times  $T_{del}$ , decreasing even more the system frequency. In conclusion, in order to optimize the system performance  $T_{del}$  value has to be minimized.

#### 4.4.2 Results

Several simulations have been carried out, in order to determine the minimum time step, as well as the maximum working frequency (including the total wire model). It is important to highlight the approximated nature of these results, where the wire model is not as precise as an extracted simulation. However, this is completely adequate for pre-layout simulations. Best results (after *corners and Monte Carlo simulations*) of this TDC-based technique, are shown in Table 4.1.

TDC-Based Simulation Results			
Results	Value	Unit	Simulation Comments
Min. Time Step $T_{del}$	100	$pS$	<i>Corners and Monte Carlo</i> simulations / 400MHz@1.2V / 5-bit bus / 2.04 mm link
Min. Setup Time $T_{setup}$	560	$pS$	3-bit bus (min. simulated bus width) / 400MHz@1.2V / TT corner / 27°C
Wide Bus Width	5	$bits$	Min. simulated bus width
Wire Length	2.04	$mm$	–
Max. Clock Frequency	400	$MHz$	3-bit bus / 10-bit pattern transmission / 1.2V / TT corner / 27°C
Min. Voltage Supply	0.6	$V$	3-bit bus / 10-bit pattern transmission / 400MHz / TT corner / 27°C
Energy/Bit ( $E_b$ )	4.47	$pJ/b$	4-bit bus / 10-bit pattern transmission / 49.5 ns simulation time / 200MHz@1.2V / 2.04 mm link / TT corner / 27°C
Average Total Power Consumption ( $P_{tot}$ )	3.6193	$mW$	4-bit bus / 10-bit pattern transmission / 49.5 ns simulation time / 200MHz@1.2V / TT corner / 27°C
Average Total Power Consumption ( $P_{tot}$ )	5.6258	$mW$	4-bit bus / 10-bit pattern transmission / 49.895 ns simulation time / 200MHz@1.2V / 2.04 mm link / TT corner / 27°C

**Table 4.1:** Best global parameters of the TDC-based approach.

The minimum time step can be further reduced. However, this design has not been fabricated, such that  $T_{del}$  remained a moderate value. This system is considerably sensitive to setup time constraints. However, setup time can be decreased by reducing time step.

The simulation time depends exponentially on the bus width. Therefore, 5-bit bus remained as the maximum simulated bus width. Nevertheless, wider bus designs would be similar to the 5-bit bus.

As shown, this design is substantially sensitive to bus width variation, reducing overall frequency. Therefore, maximum clock frequency is achieved with a 3-bit bus. Again, overall frequency can be increased while optimizing minimum time step.

Voltage drops does not pose an important issue, regarding this design. Minimum voltage supply remained 0.6 V, holding a normal system function. Energy per bit has been measured including a total wire model.

Both  $E_b$  and  $P_{tot}$  can be optimized by optimizing the time step. In Equation 4.2, it is highlighted the three main components of the total power consumption.

$$P_{tot} = P_{encoder} + P_{wire} + P_{decoder} \quad (4.2)$$

Encoder-decoder power contribution can be neglected in long wires, where the overall system becomes more efficient. In addition, it is important to emphasize that all of these results are completely wire-length dependent. Wire increases the delay, power consumption (repeaters) and distorts the signal. Consequently, variations in the wire length and/or width and re-optimizations of the wire segment and the number of repeaters, can lead to improvements in performance.

Finally, in order to provide a global vision of this technique, following chapters include a benchmark. Also, benefits and disadvantages of this approach is reviewed in conclusions.

# Chapter 5

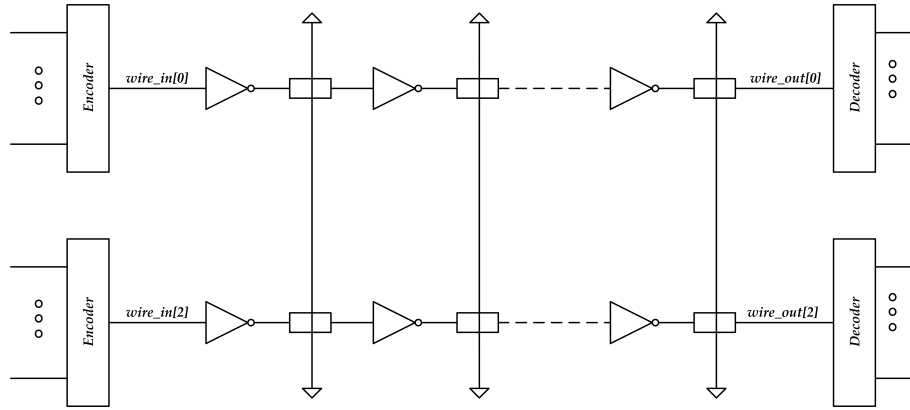
## Bus Design Techniques Benchmark

The differences between *PWM-based* and *TDC-based* 4-bit bus design techniques are reviewed in this chapter. This is first illustrated in Table 5.1

Parameter	Bus Design Techniques Benchmark	
	4-bit PWM-based Bus	4-bit TDC-based Bus
Max. Simulated Bus Width	6-bit bus	5-bit bus
Wire Length	2.04 mm	2.04 mm
Max. Clock Frequency	1 GHz	400 MHz
Min. Supply Voltage	1.1 V	0.6 V
Encoder+Decoder Energy/bit	0.336 pJ/b	4.47 pJ/b
Wire Energy/bit	0.498 pJ/b (2x2.04 mm)	2.6 pJ/b (2x2.04 mm)
Total Energy/bit (with Wire)	0.835 pJ/b	7.07 pJ/b
Full-Swing/Low-Swing	Full-Swing	Full-Swing
Crosstalk	Crosstalk-aware circuitry	No crosstalk-aware circuitry
Clock Overhead	No	Yes
Implementation Complexity	High	Medium-Low
Technology Node	nanoscale FDSOI	nanoscale FDSOI
Field of Application	Low-power buses in parallel architectures, NoCs	Low-power buses in parallel architectures, NoCs, HNNs
Extra Benefits	–	Time-domain fast and low-power calculations

**Table 5.1:** Best global parameters benchmark between a PWM-based and a TDC-based bus approach.

Illustrated results have been obtained from a common simulation structure for both bus design approaches. A 4-bit bus width has been fixed, as well as an identical wire length. A generic structure is represented in Figure 5.1.



**Figure 5.1:** Common testbench for both approaches.

In Table 5.1, main comparative results are shown. Bus width capabilities are presented, with 6-bit for the PWM-based bus and 4-bit for the TDC-based bus, as a maximum simulated value. However, PWM-based bus width can be extended to any multiple of 2, as well as TDC-based bus width which can be extended to any value, with less power consumption than previous one (lowering clock frequency).

Both techniques have been tested with a 2.04 mm wire link. Nevertheless, both of them are suitable for longer links, where the encoder/decoder overhead becomes depreciated.

In terms of clock frequency, PWM-based bus design reveals a superior performance, with a higher maximum clock frequency as well as less power consumption. A dependence between maximum frequency and wire length is needed to be highlighted, since propagation delay becomes significant in a long wire transmission.

A higher voltage sensitivity is shown by PWM-based bus design technique, since a variation in supply voltage leads to a pulse width alteration, unbalancing the system. Likewise, TDC-based bus approach becomes more robust against voltage supply drops, as it is still functional at 0.6 V.

A considerable difference in terms of energy per bit dissipation between PWM-based and TDC-based bus techniques is revealed. Encoder/decoder energy dissipation overhead of PWM-based bus, is approximately 13x lower than TDC-based bus design approach. Internally, TDC-based structure includes gates which switch every clock cycle, since its input is driven by clock signal. This every-clock-cycle switching of the TDC-based Encoder/Decoder bus technique can lead to a higher power consumption. Each wire energy dissipation has been obtained taking into consideration the structure represented in Figure 5.1, where two parallel 2.04 mm links are included.

Both techniques are based on full-swing signaling. This fact has been outlined in order to emphasize that despite of being a pseudo-analog encoding/decoding technique, both of them are fully digital approaches, with all the associated benefits. Synthesis capability from a High Level Hardware Description Language (e.g. *Verilog*) is one of the main advantages. Also, these approaches are fully compatible with the standard EDA flow.

The TDC-based approach requires a highly restrictive demand of time synchrono-

nization. In order to achieve this level of time resolution, clock has to be propagated beside the data signal, through the same distance. Regarding the PWM-based technique, clock signal propagation is not a necessity, since all the information is encoded into the pulse width.

As described in previous sections, in order to successfully implement a PWM-based bus, an encoder, decoder and crosstalk-aware circuitry are needed. Furthermore, once the encoder/decoder structure is implemented, crosstalk-aware circuitry has to be calibrated realizing several simulations. However, a full n-bit bus synthesis is available for the TDC-based bus approach, without any further adjustments. This fact leads to a "high" and a "medium-low" implementation complexity for the TDC-based and the PWM-based bus design technique, as a qualitative description. Both techniques have been implemented using a nanoscale FDSOI process technology.

Possible application areas are similar for both approaches. However, regarding these results, PWM-based technique aims to reduce power consumption in a higher amount, providing a *state-of-the-art* bus performance. Long wires power reduction and NoCs interconnect wiring are two appropriate applications for this approach. Likewise, TDC-based approach targets on decreasing interconnection complexity and area reduction of high-fan-out buses, since its encoding capabilities into the same wire are higher. Therefore, low-power and high fan-out long buses, NoCs interconnect wiring and HNNs wiring (with fast time-domain calculations) are the most suitable application domains for this technique.

Time-domain fast and low-power calculations remained out of target of this thesis. However, some calculations can be more efficiently carried out in time-domain, with less power consumption. Some examples of these operations are multiplication with constant coefficient, summation or minimum calculation.

# Chapter 6

## Conclusions and Future Work

This thesis first explained the basics of wiring theory, describing the three most common parameters used to model wires. Resistance, inductance and capacitance were detailed. Crosstalk approximated model was presented and included in a *Distributed RC Model*, through which all simulations were carried out. Then, an optimization procedure for the size and amount of repeaters in a wire link was presented.

Crosstalk-aware pulse width modulation bus technique was fully described. The Encoder-Decoder scheme, as well as crosstalk-aware circuitry were characterized. Simulated results were shown, emphasizing benefits and disadvantages. Likewise, time-domain analog conversion bus technique was elucidated. Encoder/Decoder schemes were described as well. Simulation and results were showed and interpreted. Both of them were implemented in a nanoscale FDSOI process technology.

A comparison between these two bus approaches has been synthesized in a benchmark table. Performance parameters, applications areas and other technical aspects have been highlighted and compared.

Total wire model provided enough accuracy, leaving extracted simulations apart. However, results of the optimization procedure for the size and amount of repeaters were not suitable for these type of pulsed signaling techniques. Then, optimized results were modified, in order to fit timing requirements, leading to oversized repeaters and undersized segment lengths.

Both bus designs met the performance and power consumption requirements. PWM-based bus technique remained as a design approach perfectly suitable for long and high-power consuming wires. Furthermore, TDC-based technique revealed its benefits while designing high-fan-out buses, as well as HNNs wiring interconnect.

Although the goal of this thesis was attained, additional work is recommended. First, optimization procedure for the size and amount of repeaters requires a deep analysis, in order to adjust the model to the latest process technology nodes.

In addition, additional work for the PWM-based bus design approach is needed. An encoder/decoder transistor-level optimization can be achieved, taking advantage of the unique design capabilities that FDSOI provides. A delay optimization of crosstalk-aware circuitry and an increase in the *variable delay* (Figure 3.12) resolution is feasible. Moreover, crosstalk-aware circuitry needs versatility, so that crosstalk avoidance circuits can be dynamically adjusted. Additionally, an extracted simulation would help to improve accuracy in a real design implementations. An automated calibration model can be added as future work, as well as an HDL system model and complete bus technique layouts.

Similarly, TDC-based approach requires more *corners* and/or *Monte Carlo* simulations of various bus-size synthesized systems. Setup-time requirements can be enhanced, leading to a frequency improvement. Finally, implementing the layout and system prototyping can reveal additional results and remaining work.

## References

- [1] Mark Buckler, Wayne Burleson, and Greg Sadowski, “Low-power networks-on-chip: Progress and remaining challenges,” *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, pp. 132–134, 2013.
- [2] Janardan Misra and Indranil Saha, “Artificial neural networks in hardware: A survey of two decades of progress,” *Neurocomputing*, pp. 239–255, 2010.
- [3] Karl F. Goser, “Implementation of artificial neural networks into hardware: Concepts and limitations,” *Mathematics and Computing in Simulation*, pp. 161–171, 1996.
- [4] B. Victor and K. Keutzer, “Bus encoding to prevent crosstalk delay,” *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, pp. 57–63, 2001.
- [5] J. Seo, D. Sylvester, D. Blaauw, H. Kaul, and R. Krishnamurthy, “A robust edge encoding technique for energy-efficient multi-cycle interconnect,” *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, pp. 68–73, 2007.
- [6] M. Khellah, J. Tschanz, Y. Ye, S. Narendra, and V. De, “Static pulsed bus for on-chip interconnects,” *Symp. VLSI Circuits Dig. Tech. Papers*, pp. 78–79, 2002.
- [7] H. Deogun, R. Senger, D. Sylvester, R. Brown, and K. Nowka, “A dual-V<sub>dd</sub> boosted pulsed bus technique for low power and low leakage operation,” *Proc. Int. Symp. Low Power and Electronics and Design (ISLPED)*, pp. 73–78, 2006.
- [8] A. B. Kahng, S. Muddu, and E. Sarto, “Interconnect optimization strategies for high-performance VLSI designs,” *Proc. Int. Conf. VLSI Design*, pp. 464–469, 1999.
- [9] C. J. Akl and M. A. Bayoumi, “Reducing interconnect delay uncertainty via hybrid polarity repeater insertion,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, pp. 1230–1239, 2008.
- [10] Jae-sun Seo, David Blaauw, and Dennis Sylvester, “Crosstalk-aware PWM-based on-chip links with self-calibration in 65 nm CMOS,” *IEEE Journal Of Solid State Circuits*, pp. 2041–2052, 2011.
- [11] Eisse Mensink, Daniel Schinkel, Eric A. M. Klumperink, Ed van Tuijl, and Bram Nauta, “Power efficient gigabit communication over capacitively driven RC-limited on-chip interconnects,” *IEEE Journal Of Solid State Circuits*, pp. 447–457, 2010.
- [12] Yi Liu, Gang Liu, Yintang Yang, and Zijin Li, “A novel low-swing transceiver for interconnection between NoC routers,” *Digital Content, Multimedia Technology and its Applications (IDCTA)*, pp. 39–44, 2011.

- [13] Daisuke Miyashita, Ryo Yamaki, Kazunori Hashiyoshi, Hiroyuki Kobayashi, Shouhei Kousai, Yukihiro Oowaki, and Yasuo Uekawa, “An LDPC decoder with time-domain analog and digital mixed-signal processing,” *IEEE Journal Of Solid State Circuits*, pp. 73–83, 2014.
- [14] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic, *Digital Integrated Circuits: A Design Perspective*. Faculty Publications.
- [15] Ron Ho, *On-Chip Wires: Scaling and Efficiency*. PhD thesis, Department Of Electrical Engineering, Stanford University, 2003.
- [16] David Harris, Ron Ho, Gu-Yeon Wei, and Mark Horowitz, “The fanout-of-4 inverter delay metric,” [Online]. Available: <http://www3.hmc.edu/harris/research/FO4.pdf>, unpublished.



## Appendix

```

%% Optimal Number of Repeaters in a nanoscale FDSOI CMOS Bus
% Author: Alejandro Rodríguez Ramos
% Email: alejandro.dosr@gmail.com
% 28th January, 2014

% Optimizing Energy.Delay Product
%clear
% Constants
Leff=...; % Gate length (um)
Rv=25000*Leff; % Device parameter (Ohm*um)?
Cg=2e-15; % Gate capacitance (aproximation over different
technologies) (F/um)
Cd=0.5*Cg; % Drain capacitance (F/um)
Rw=...; % Ohm/um
Cw=...; % F/um
dl=[0.4:0.01:2.2]; % Deviation from optimal segment length
(adimensional)
dw=[0.4:0.005:1.4-0.005]; % Deviation from optimal device
width (adimensional)
FO4=11.44e-12; % Another way to obtain it? (s)
L=4000; % Total bus length (um)
V=1.2; % Supply voltage (V)
c=1; % Crowbar currents (adimensional)

% Equations
lopt=3*sqrt((Rv*Cg)/(Rw*Cw)); % Optimum segment length (um)
wopt=(1/sqrt(3))*sqrt((Rv*Cw)/(Rw*Cg)); % Optimum
transistor width (inverter) (um)

for i = 1:length(dw)
    for j = 1:length(dl)
        Delay(i,j)=((0.34*(dl(j)+(1./dl(j))))+(0.4*(dw(i)
+ (1./dw(i)))))*sqrt(FO4*Rw*Cw); % Delay
        function (s)
        Etot(i,j)=Cw*L*(V^2)*(1+((4.5*c*dw(i))./(
3*sqrt(3)*dl(j)))); % Energy
        function (J)
        ED(i,j)=Delay(i,j)*Etot(i,j);
    end
end
end

% Plotting

```

```

%surface(dl, dw, ED)
xlabel('dl')
ylabel('dw')
zlabel('Energy.Delay*1e23')
title('Energy.Delay*1e23')
surfc(dl, dw, (Delay-min(min(Delay)))*100/min(min(Delay)))
%[C,H]=contour(dl, dw, 1e23*ED, 50);
colormap gray
xlabel('dl')
ylabel('dw')
zlabel('Delay penalty from optimal (%)')
%zlabel('Energy-Delay*1e23')
%title('Energy-Delay*1e23')

% Console
clc
disp(sprintf('The wopt (delay) is: %s um',wopt));
disp(sprintf('The lopt (delay) is: %s um', lopt));

minED=min(min(ED));
[r,c]=find(ED==min(min(ED)));
disp(sprintf('The Energy*Delay minimum is: %s', minED));
minD=Delay(r(ceil(length(r)/2)),c(ceil(length(c)/2)));
disp(sprintf('The Delay at this point is: %s', minD));
minE=Etot(r(ceil(length(r)/2)),c(ceil(length(c)/2)));
disp(sprintf('The Energy at this point is: %s', minE));

dlopt=dl(c(ceil(length(c)/2)));
disp(sprintf('The dl_opt (EnergyDelay): %s', dlopt));

dwopt=dw(r(ceil(length(r)/2)));
disp(sprintf('The dw_opt (EnergyDelay): %s', dwopt));

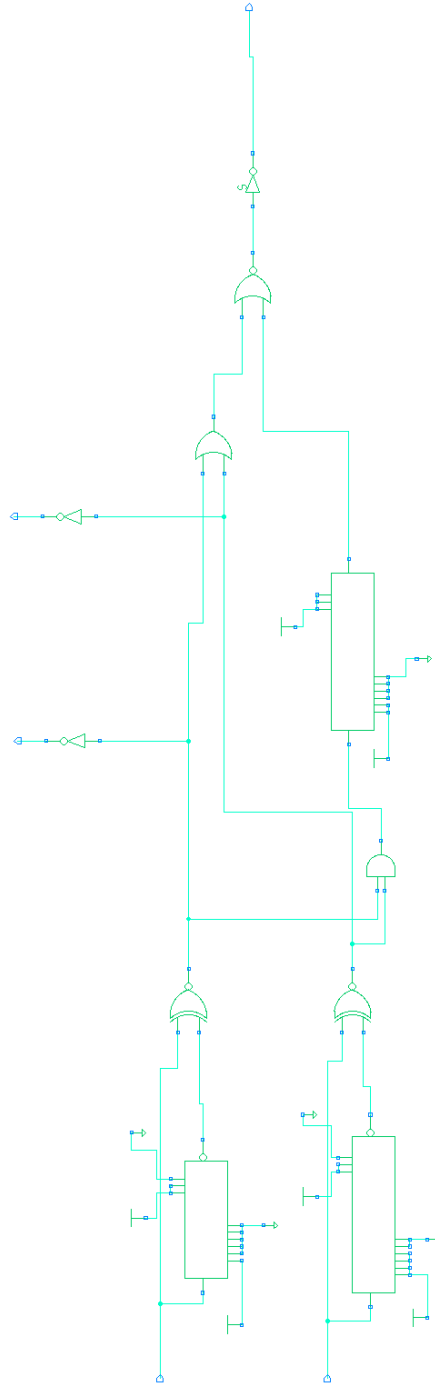
tw=dwopt*wopt;
disp(sprintf('Final transistor width at optimum point: %s um', tw));

sl=dlopt*lopt;
disp(sprintf('Final segment length at optimum point: %s um', sl));

numRep=(L/sl);
disp(sprintf('Final number of repeaters at optimum point: %s', numRep));

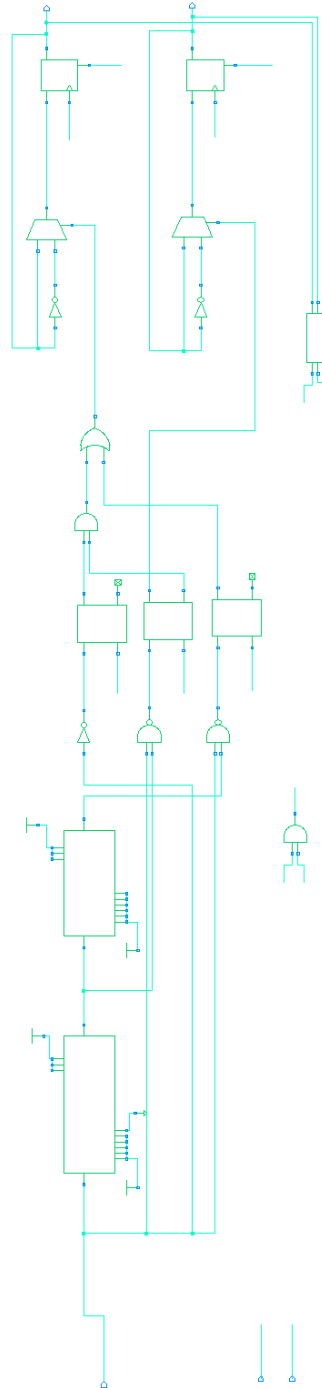
```

## Appendix

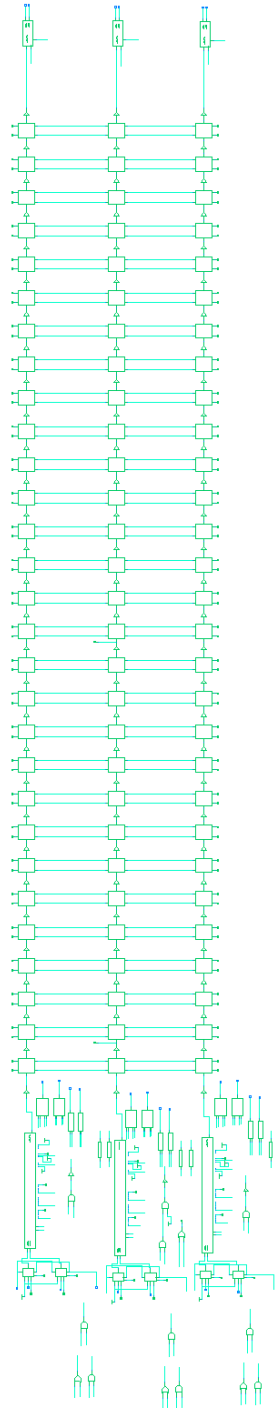


**Figure B1:** Schematic of the PWM encoder

## Appendix

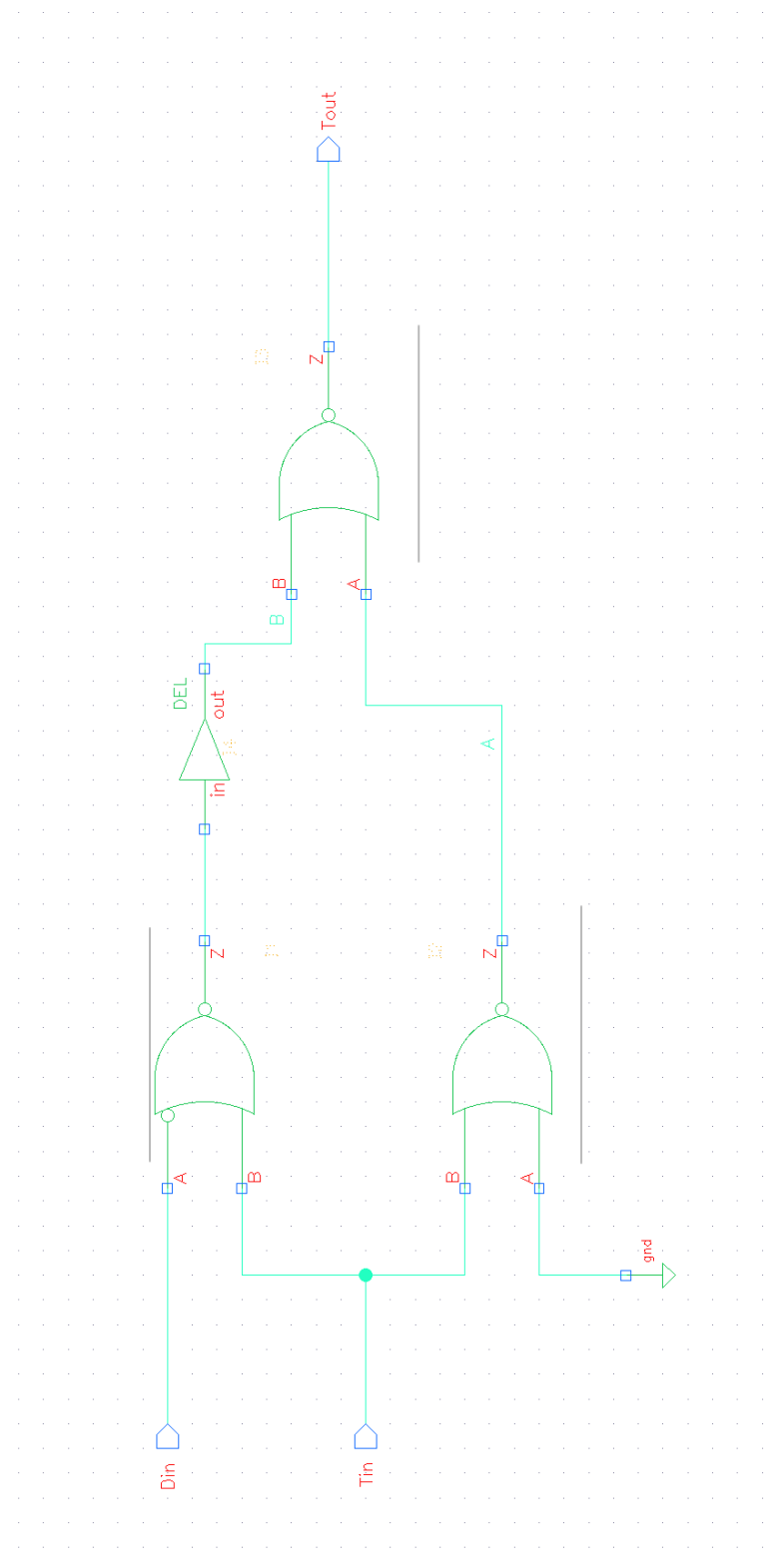
**Figure C1:** Schematic of the PWM decoder

## Appendix



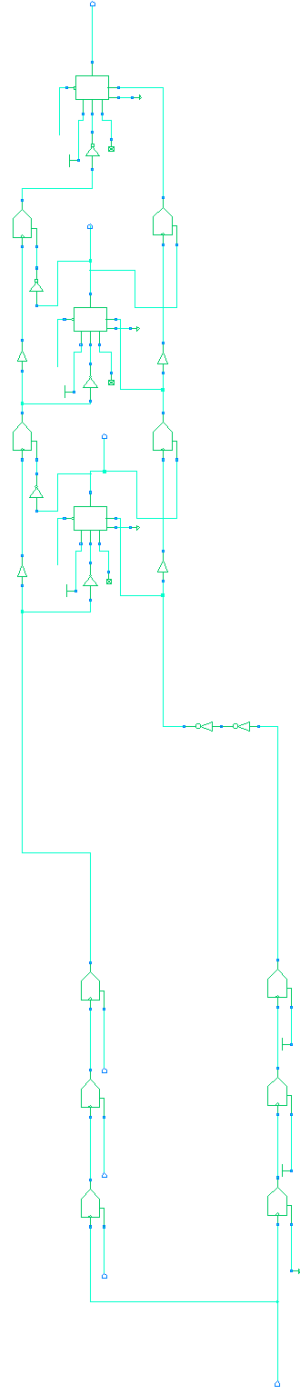
**Figure D1:** Schematic of the complete PWM system (3 wires affected by crosstalk).

# Appendix



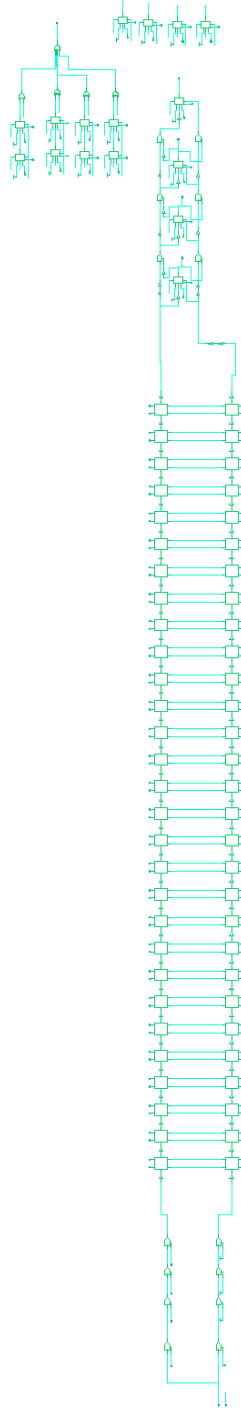
**Figure E1:** Schematic of the basic DTC cell

## Appendix



**Figure E1:** Schematic of the complete 3-bit TDC system.

# Appendix



**Figure E1:** Schematic of the complete 4-bit TDC system, with the total wire model.



## Appendix

```

//-----
//
//-----
//
//          Verilog model of an inverters chain
//
//
//-----
//
//Date          : Tue Feb 25th, 2014
//Description   : A Time-to-Digital converter...
//State        : Already synthesized and simulated (working).
//
//-----

module Del_Half_B (
    in , //input
    out ); // output

    parameter Ndel=19; //Number of inverters

    input in;
    output out;

    wire [Ndel-2:0] ints;

    genvar i;
    generate
        for (i = 0; i < Ndel; i = i + 1) begin : INV
            if (i==0) begin
                INV(.Z(ints[i]), .A(in));
            end
            else if (i==(Ndel-1)) begin
                INV(.Z(out), .A(ints[i-1]));
            end
            else begin
                INV(.Z(ints[i]), .A(ints[i-1]));
            end
        end endgenerate
endmodule

```

## Appendix

```
//-----
//
//          Author: Alejandro Rodriguez Ramos
//
//          Email: alejandro.dosr@gmail.com
//
//-----
//
//-----
//
//          Verilog model of a Digital-to-Time Converter
//                          (DTC)
//
//
//-----
//
//Date          : Mon Feb 24th, 2014
//Description    : A Digital-to-Time converter...
//State         : Already synthesized and simulated (working).
//
//-----
module DTC_nXDEL (
    Din    ,// Data input
    Tin    ,// Rising edge input
    Tout   );// Rising edge output

    parameter Ndel=32; //Has to be an even natural
    input Din, Tin;
    output Tout;

    wire [Ndel-2:0] ints;
    wire ZB;
    wire A;
    wire B;

    NOR NOR1(.Z(ZB), .A(Din), .B(Tin));
    NOR NOR2(.Z(A), .A(1'b0), .B(Tin));
    NOR NOR3(.Z(Tout), .A(A), .B(B));

    genvar i;
    generate
        for (i = 0; i < Ndel; i = i + 1) begin : DTC_nXDEL
            n// Variable instantiation of the inverters chain
```

```
    if (i==0) begin
        IV INV(.Z(ints[i]), .A(ZB));
    end
    else if (i==(Ndel-1)) begin
        IV INV(.Z(B), .A(ints[i-1]));
    end
    else begin
        IV INV(.Z(ints[i]), .A(ints[i-1]));
    end
end endgenerate

endmodule
```

## Appendix

```
//-----
//
//          Author: Alejandro Rodriguez Ramos
//
//          Email: alejandro.dosr@gmail.com
//
//-----
//
//-----
//
//          Verilog model of a n bits Digital-to-Time Converter
//                          (DTC)
//
//
//-----
//
//Date          : Tue Feb 25th, 2014
//Description   : A Digital-to-Time converter...
//State        : Already synthesized and simulated (working).
//
//-----
`celldefine
`timescale 1ns/1ps

module DTC_nb_nX50 (
    clk    ,// Clock input
    Din    ,// N bits data input
    refin  ,// Reference control input
    ref    ,// Clock Reference output
    td    );// Time-Domain output

    parameter n=3, // Number of input bits
               k=2; // kX the lowest time separation between bits
               (1X == 50 ps)

    input clk;
    input [n-1:0] Din;
    input [n-1:0] refin;
    output ref, td;

    wire [n-2:0] ints;
    wire [n-2:0] ints2;
    reg NOTIFIER;
```

```

genvar i;
generate
  for (i = 0; i < n; i = i + 1) begin : DTC
    if (i==0) begin
      DTC_nXDEL #(.Ndel(k*16)) DTC (.Din(Din[i]),
        .Tin(ints[i]), .Tout(td));
      DTC_nXDEL #(.Ndel(k*16)) DTC2 (.Din(refin[i]),
        .Tin(ints2[i]), .Tout(ref));
    end
    else if (i==(n-1)) begin
      DTC_nXDEL #(.Ndel((2**i)*k*16)) DTC (.Din(Din[i]),
        .Tin(clk), .Tout(ints[i-1]));
      DTC_nXDEL #(.Ndel((2**i)*k*16)) DTC2
        (.Din(refin[i]), .Tin(clk),
        .Tout(ints2[i-1]));
    end
    else begin
      DTC_nXDEL #(.Ndel((2**i)*k*16)) DTC (.Din(Din[i]),
        .Tin(ints[i]), .Tout(ints[i-1]));
      DTC_nXDEL #(.Ndel((2**i)*k*16)) DTC2
        (.Din(refin[i]), .Tin(ints2[i]),
        .Tout(ints2[i-1]));
    end
  end endgenerate

specify
  specparam
    //Timing parameters
    tsetup$Din$clk = k*0.05,
    thold$Din$clk = k*0.05;

    //Timing checks
    $setphold(posedge clk, posedge Din, (2**(n-1))*tsetup
      $Din$clk*1.4,(2**(n-1))*thold$Din$clk*1.4,
      NOTIFIER);
    $setphold(posedge clk, negedge Din, (2**(n-1))*tsetup
      $Din$clk*1.4,(2**(n-1))*thold$Din$clk*1.4,
      NOTIFIER);
endspecify

endmodule

```

## Appendix

```

//-----
//
//          Author: Alejandro Rodriguez Ramos
//
//          Email: alejandro.dosr@gmail.com
//
//-----
//
//-----
//
//          Verilog model of an inverters chain
//
//
//
//-----
//
//Date          : Tue Feb 25th, 2014
//Description   : A Time-to-Digital converter...
//State        : Already synthesized and simulated (working).
//
//-----
out ); // output

parameter n=122; //Number of inverters
input in;
output out;
wire [n-2:0] ints;

genvar i;
generate
  for (i = 0; i < n; i = i + 1) begin : WAIT
    if (i==0) begin
      IV INV_WAIT(.Z(ints[i]), .A(in));
    end
    else if (i==(n-1)) begin
      IV INV_WAIT(.Z(out), .A(ints[i-1]));
    end
    else begin
      IV INV_WAIT(.Z(ints[i]), .A(ints[i-1]));
    end
  end
end generate
endmodule

```

## Appendix

```
//-----
//
//          Author: Alejandro Rodriguez Ramos
//
//          Email: alejandro.dosr@gmail.com
//
//-----
//
//-----
//
//          Verilog model of a n bits Time-to-Digital Converter
//                          (TDC)
//
//
//-----
//
//Date          : Tue Feb 25th, 2014
//Description    : A Time-to-Digital converter...
//State         : Already synthesized and simulated (working).
//
//-----

module TDC_nb_nX50 (
    td    ,// Time-Domain data input
    ref   ,// Clock reference input
    r_n   ,// Reset input
    e     ,// Enable input (FF control signal)
    ti    ,// Ti input (FF control signal)
    te    , // Te input (FF control signal)
    Dout  );// Pararell data output

    parameter n=3, // Number of output bits
               k=2; // kX the lowest time separation between bits (1X
                   == 50 ps)

    input td, ref, r_n;
    input [n-1:0] e, ti, te;
    output [n-1:0] Dout;

    wire [n-1:0] in, in_wait, Dout_n, in_n, clk, clk_wait;
    wire net, ref_hold;

    genvar i;
```

```

generate
  for (i = 0; i < n; i = i + 1) begin : TDC
    if (i==0) begin : TDC
      LLHF REG (.Q(Dout[i]),
        .D(in_n[i]), .E(e[i]), .CP(clk[i]), .RN(r_n),
        .TI(ti[i]), .TE(te[i]));
      IV INV (.Z(in_n[i]), .A(in[i]));
    end
    else if (i==(n-1)) begin : TDC
      DTC_nXDEL #(.Ndel((2**(i-1))*k*16)) DTC
        (.Din(Dout_n[i]), .Tin(in_wait[i]),
        .Tout(in[i-1]));
      DTC_nXDEL #(.Ndel((2**(i-1))*k*16)) DTC2
        (.Din(Dout[i]), .Tin(clk_wait[i]),
        .Tout(clk[i-1]));
      IV INV2 (.Z(Dout_n[i]), .A(Dout[i]));
      LLHF REG (.Q(Dout[i]),
        .D(in_n[i]), .E(e[i]), .CP(ref_hold), .RN(r_n),
        .TI(ti[i]), .TE(te[i]));
      IV INV (.Z(in_n[i]), .A(td));
      if (i > 2) begin : WAIT_TDC
        WAIT #(.n((k-1)*(2**(i-2))*122)) WAIT1 (.in(td),
        .out(in_wait[i]));
        WAIT #(.n((k-1)*(2**(i-2))*122)) WAIT2
        (.in(ref_hold), .out(clk_wait[i]));
      end
    else begin : WAIT_TDC
      WAIT #(.n((k-1)*122)) WAIT1 (.in(td),
        .out(in_wait[i]));
      WAIT #(.n((k-1)*122)) WAIT2 (.in(ref_hold),
        .out(clk_wait[i]));
    end
  end
  else begin : TDC
    DTC_nXDEL #(.Ndel((2**(i-1))*k*16)) DTC
      (.Din(Dout_n[i]), .Tin(in_wait[i]),
      .Tout(in[i-1]));
    DTC_nXDEL #(.Ndel((2**(i-1))*k*16)) DTC2
      (.Din(Dout[i]), .Tin(clk_wait[i]),
      .Tout(clk[i-1]));
    IV INV2 (.Z(Dout_n[i]), .A(Dout[i]));
    LLHF REG (.Q(Dout[i]),
      .D(in_n[i]), .E(e[i]), .CP(clk[i]), .RN(r_n),
      .TI(ti[i]), .TE(te[i]));
    IV INV (.Z(in_n[i]), .A(in[i]));
  end
end

```



```
if (i > 2) begin : WAIT_TDC
    WAIT #(.n((k-1)*(2**(i-2))*122)) WAIT1
        (.in(in[i]), .out(in_wait[i]));
    WAIT #(.n((k-1)*(2**(i-2))*122)) WAIT2
        (.in(clk[i]), .out(clk_wait[i]));
end
else begin : WAIT_TDC
    WAIT #(.n((k-1)*122)) WAIT1 (.in(in[i]),
        .out(in_wait[i]));
    WAIT #(.n((k-1)*122)) WAIT2 (.in(clk[i]),
        .out(clk_wait[i]));
end
end
end endgenerate

Del_Half_B #(.Ndel(19)) DEL_HALF1 (.in(ref), .out(net));
Del_Half_B #(.Ndel(19)) DEL_HALF2 (.in(net), .out(ref_hold));

endmodule
```