Implementation of Delay and Power Reduction in Deep Sub-Micron Buses Using Coding

by

Theodoros K. Konstantakopoulos

Diploma Electrical and Computer Engineering
University of Patras, Greece, 2000

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2002

Author .....
Department of Electrical Engineering and Computer Science
May, 24 2002

Certified by .....
Anantha P. Chandrakasan, Associate Professor
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses
Department of Electrical Engineering and Computer Science

Implementation of Delay and Power in Deep Sub-Micron Buses Using Coding

by

Theodoros K. Konstantakopoulos

Submitted to the Department of
Electrical Engineering and Computer Science
on May 24, 2002
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Delay and Power have become the most important metrics in modern VLSI. Applications are becoming more demanding and the need for reducing both delay and power is emerging. Process scaling is constantly shifting larger portions of delay and power to buses and interconnect networks.

This work focuses on the design of practical circuit implementations, that address these two problems. A coding scheme that eliminates delay-costly transitions is proposed, thus allowing faster clocking on the bus. An increase of 36% in the total throughput is achieved, while there is a trade-off of increased latency. Furthermore, a smart and efficient implementation of charge recycling, which reduces the dynamic power dissipation when driving long buses, is presented. The design circuit can be used for an arbitrary number of bus lines, and for the test cases that were examined energy savings up to 32% are reported.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor

Acknowledgements

I couldn't thank more my parents for everything they have done for me. They have always been supportive and encouraging me to try different things in my life. They have always respected my decisions and helped me make my goals reality. I am very grateful for that.

I would have never achieved any of this, without the support and guidance of my advisor, Prof. A. Chandrakasan. He is a constant source of great ideas and enthusiasm. His supervision literally helped me move many academic steps ahead, helping me build a solid background and introducing me to the "true" meaning of research. Working with him, although some times tough, has been a great intellectual experience.

Special thanks to my friend and colleague, Paul-Peter for helping me out many times when I first came to MIT and throughout my two years of being here. His advice has been very valuable and it has been a great pleasure collaborating with him in research projects.

I would also like to thank all the people at the Digital Integrated Circuits and Systems Group, for making me feel like home, from the very start. They have been the first people to go and bug when I had a question and they have never been reluctant to help me, especially, Frank and Ben. The three of us know, better than anyone what we've been through to have this chip done, so I have to say that it was fun sharing some "area" with them.

I greatly appreciate Frank for proof-reading my thesis. His suggestions and comments made the whole text more readable.

However, there's life (yes there is) outside MIT, and I have been blessed to meet new friends here that we shared very good times together. Especially, whenever things get tough and very unfriendly, I was happy to have people to talk to and hang around.

Special thanks to Anna, Ayla, Christina, Maria, Noelle, Aggelos, George, Christos, Petros and Yiannis. They are the people I feel closest to and like to be with all the time. Also the greek community at MIT and especially the “Syllogos” for giving me a useful academic distraction.

I have to mention especially, Petros’ helpful tips in writing this thesis and his general eagerness to help everyone in need, since he seems to know better than anyone how to “GAMIT”.

Last but not least, I would like to thank all the MIT faculty and the administrative stuff for doing a perfect job in providing excellent standards of education and a very intellectually stimulating environment.

Στην μνήμη του πατέρα μου

(In memory of my father)

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In the Deep Sub-Micron era interconnect wires are responsible for an increasing fraction of the delay and power in Very Large Scale Integrated (VLSI) circuits.

1.1.1 Interconnect Networks Delay

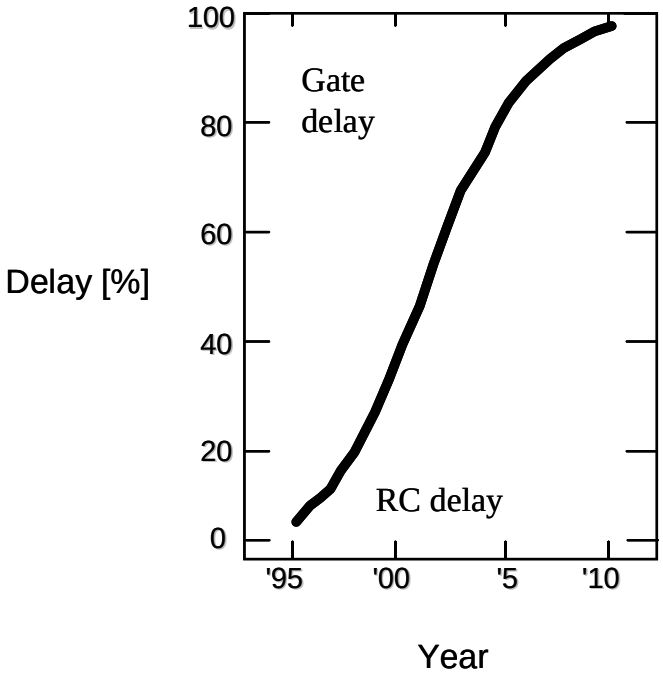

Figure 1.1: Interconnect v. Gate Delays

The graph shown in Figure 1.1 appeared in [1] and describes the trend for delay in modern VLSI. It implies that, in the near future, the component that will determine the operating frequency in integrated circuits will no longer be gate delays. The operating frequency bottleneck will be the interconnect network delay, that is the time that is required for the data to be transmitted from one point of the chip to another.

In order to examine the delay in driving a long line and driving a single gate, we used a test circuit and its schematic diagram is shown in Figure 1.2. The symbol in ***case a***

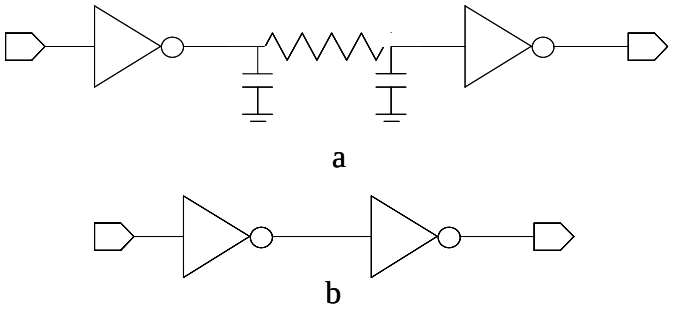

Figure 1.2: Driving long lines v. Driving gates

between the two inverters represents the long line.

For the purpose of the test, we laid out a 1cm straight metal line, with two adjacent metal lines tied to ground, to serve as the interconnect network. The four inverters were identical.

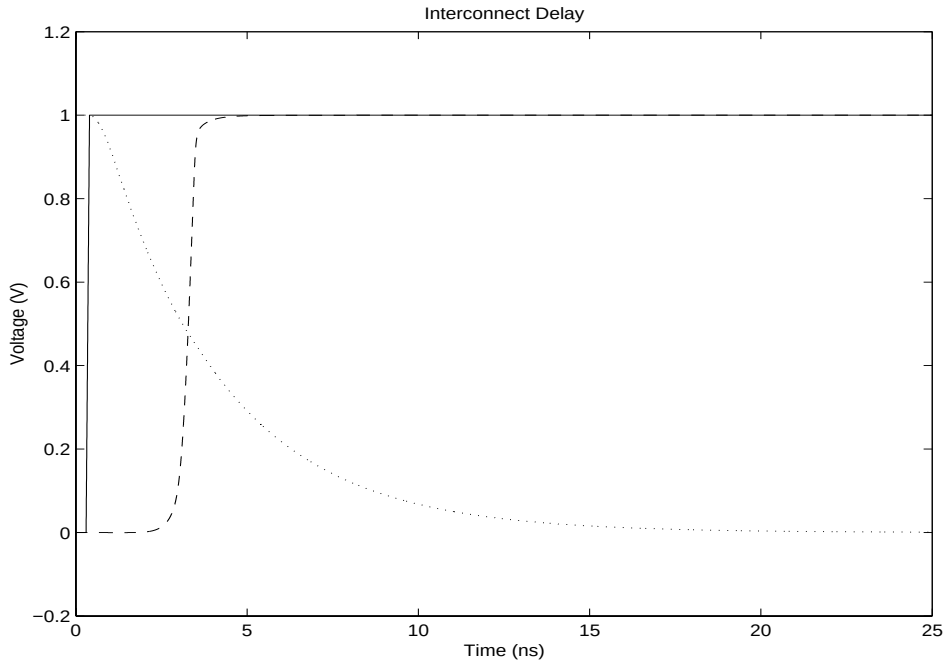

Figure 1.3: Driving a Long Line

The simulation results for case a are plotted in Figure 1.3. The input signal goes from 0 to 1 (solid line). The dotted line is the voltage at the second inverter input gate. The dashed line is the output voltage. The fall time of the line signal is 7.5ns and the transition at the second inverter output follows the first inverter input transition after 2.9ns.

The simulation results for ***case b*** are plotted in Figure 1.4. A signal transitioning from 0 to 1 was applied to the first inverter input (solid line) and the output of the second inverter is represented with the dotted line. The propagation delay for the two cascaded inverters is 70ps.

Therefore, assuming a single gate delay of 35ps, a logic block of 80 gate stages would result in the equal amount of the interconnect delay.

For area concerns, modern buses are designed with the minimum allowed spacing among the lines. This increases as well the total effective line capacitance and makes the charging and discharging of the lines even more challenging.

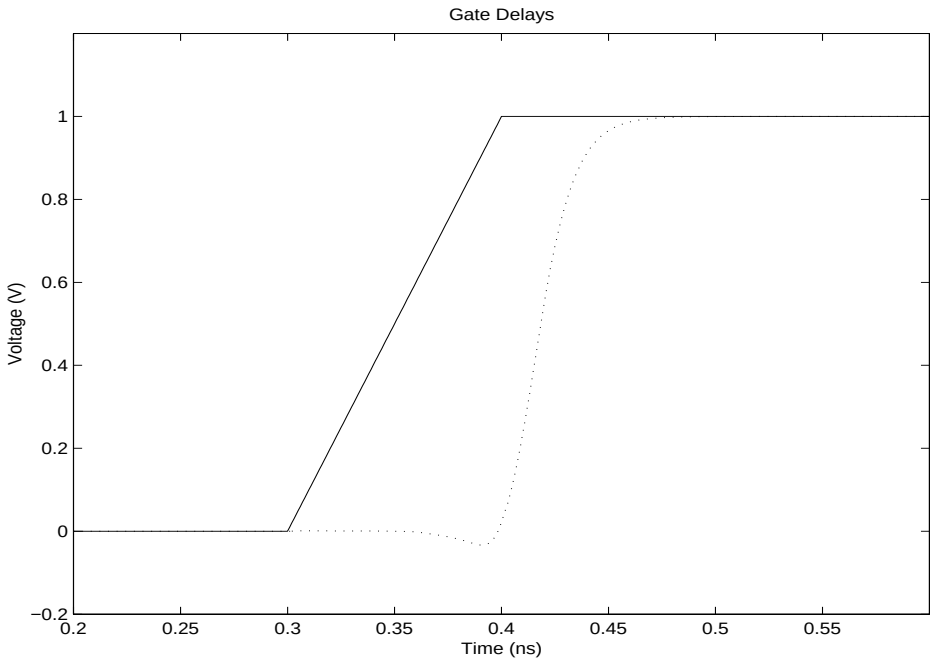

Figure 1.4: Driving Gates

The whole discussion above, proves the emerging need for smart and effective ways to transmit data in long buses, where RC delays are becoming dominant.

1.1.2 Power dissipation in Interconnect Networks

Power consumption has become a major limiting factor in modern integrated circuits. This constraint will become even more serious as the level of integration rises. The portion of the power dissipated on the interconnect is increasing rapidly with technology scaling. Therefore, smart power aware techniques have to be introduced in order to minimize this component.

To make the previous point stronger, for the purpose of this thesis, we did a power breakdown on a Cypress CPLD from the 39K design family.

The Block Diagram of the various components of the CPLD is shown in Figure 1.5.

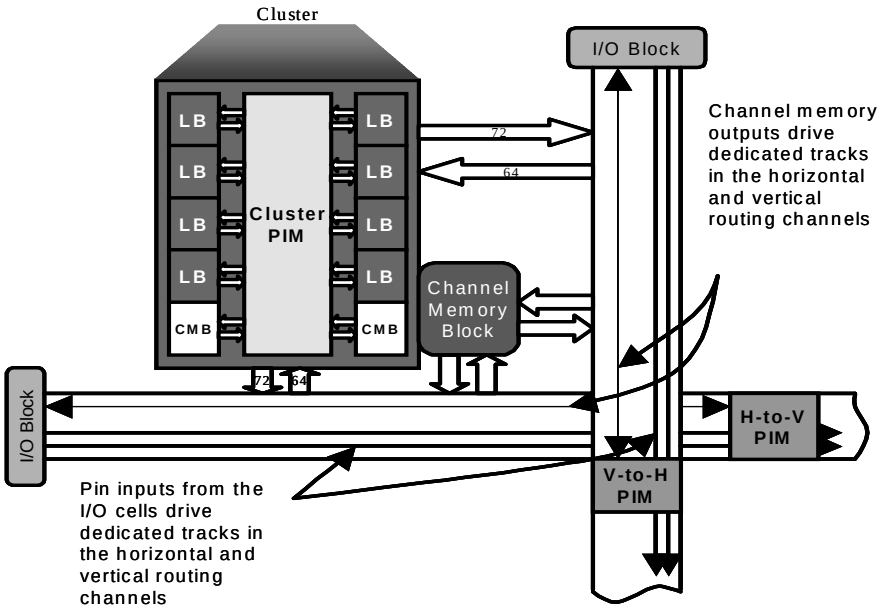

Figure 1.5: Cypress CPLD Components

The different blocks shown, were broken into subcomponents. PowerMill was used for the

simulations and we measure the average current of each component. The data that was collected is summarized in Figure 1.6, where the power breakdown is shown.

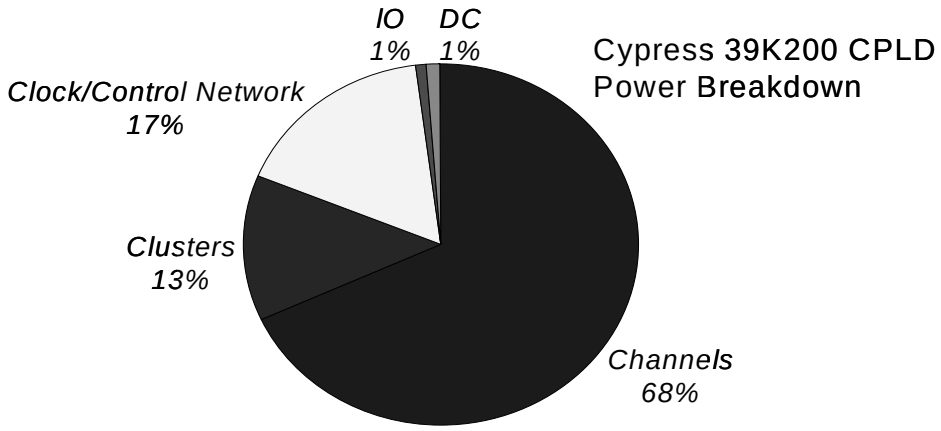

Figure 1.6: 39K200 Power Breakdown

The thing that comes out from the graph, is that the interconnect network (*Channels*) dissipates more than 65% of the total power consumption. What is also interesting is that the logic circuit (*Clusters*) only consumes 13% of the power, and 17% is allocated to the *Clock/Control Network*.

Appendix A gives a detailed description of the CPLD blocks and of the methodology that was used to measure the different power components.

Similar results are also documented in [2]. The analysis that was done on the XILINX XC4003A FPGA shows that the power consumed by the interconnect network was *65%*, on the Clock Network was *21%* and that the logic blocks were only responsible for *5%* of the total energy consumption.

1.2 Approaches

For the problems of delay and power described above, we followed two different approaches in this thesis.

We used coding in order to reduce delays in long buses. It will be obvious in the proceeding chapters that some transitions can be very time consuming. So the delay penalty paid in order to allow all the transitions on long buses is very high.

An efficient way to address this problem is coding. The data is encoded before transmitted through the bus. Coding eliminates the transitions that have high relative delay and allows only those that are not very time consuming. Using a theoretical framework that was already developed in [4], we designed practical encoding and decoding circuits to increase the actual bit rate.

Long buses, with lines strongly coupled to each other also suffer from huge power dissipation. Due to the large total line capacitance, the charge that is needed to charge a line and eventually wasted is very large, too.

In order to address the power that is dissipated on buses, we used a smart practical implementation of a charge recycling technique. Using this technique, charge is re-distributed among lines experiencing a $0 \rightarrow 1$ or $1 \rightarrow 0$ transition. Therefore, the charge that was stored in the bus is not entirely wasted. The resulting partial charge conservation reduces power consumption. Efficient charge recycling drivers were designed, appropriate for modern deep sub-micron technologies.

Chapter 2

Reducing Delay in Deep Sub-Micron Technologies

This chapter gives the necessary theoretical framework for coding. The behavior of buses in deep sub-micron technologies is presented and we give an example coding for a 3-line bus.

2.1 Modeling Buses in Deep Sub-Micron Technologies

As CMOS continues to shrink, interwire parasitic capacitances have become the dominant factor affecting performance in very deep submicron integrated circuit design. To accurately model a metal line in deep sub-micron, we should use a distributed line shown in Figure 2.1 instead of a lumped one, in order to take into account all the line capacitive and inductive parasitics.

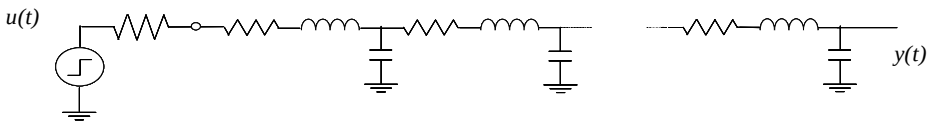

Figure 2.1: Parasitic model for a long metal line

Things get more complicated when dealing with long buses. This is mainly because of technology scaling. As transistors scale, the widths of the lines decrease as well. However, for resistance concerns, the height of the lines doesn't decrease proportionally. This results in narrow and tall lines and large inter-wire capacitance. This produces significant delay increase.

Figure 2.2 shows an appropriate sub-micron bus model. The coupling between the adjacent lines is very strong and is distributed among the whole length of the bus. Consequently a transition in any line affects the transient response of all the bits in the bus.

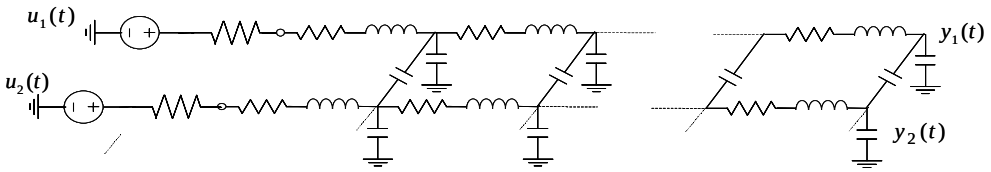

Figure 2.2: Coupled RC model for a long metal bus

The key modeling challenges include modeling of interconnect capacitance, line resistance and inductance along with the physical phenomena that determine these electrical parameters.

2.2 Transitions and their Delays

In order to demonstrate the problems caused by interwire capacitance, we use the simplified bus model shown in Figure 2.3. The effective capacitance that the driver of line l_i sees varies, depending on the transitions of the neighboring lines l_{i-1} and l_{i+1}.

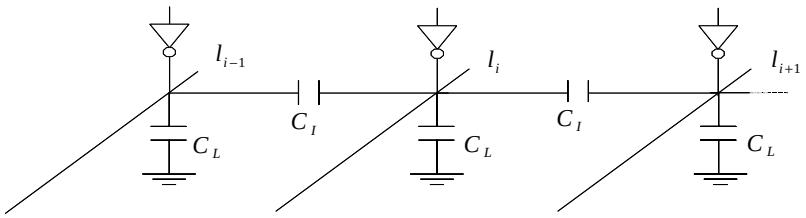

Figure 2.3: Simplified Model - 3 Adjacent Lines

In the model, the lumped capacitances C_L and C_I are used to represent the total line-to-ground and line-to-line capacitances, respectively. If $\lambda \equiv C_I / C_L$ and if we assume that a line is only affected by the transitions on the neighboring lines, then the effective normalized capacitance C_{total}/C_L for line l_i can vary from:

$$C_{total}/C_L = \begin{cases} 1 & \text{When adjacent lines make the same transition} \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 + 4\lambda & \text{When adjacent lines make opposite transitions} \end{cases} \quad (2.1)$$

depending on the transitions on the adjacent lines.

When two adjacent lines make the same transition, assuming that there is no significant relative delay between them, the interwire capacitance C_I has no effect on the charging of the lines. However, if only one line changes state, the driver sees an interwire capacitance in addition to C_L. The interwire capacitance becomes more significant, when the two lines experience opposite transitions. Due to the Miller effect, the effective coupling capacitance doubles to $2C_I$. This explains the worst case for the line l_i, in (2.1). Additionally to the line capacitance C_I there is the $2C_I$ from l_{i-1} and another $2C_I$ from l_{i+1}.

It is obvious that not all the transitions require the same time to complete. Unfortunately, the operating frequency is determined by the worst case relative delay. However, the different transition delays can be grouped into different delay classes [3]. This is described in Table 2.1,

$\hat{T}$ =	**0**	**1**	1 + λ	1 + 2λ	1 + 3λ	1 + 4λ
Transition Delay Classes	X	0 (M0)	1 (M1)	2 (M2)	3 (M3)	4 (M4)

Table 2.1: Transition Delay Classes

where the different Delay Classes are denoted with {X,0,1,2,3,4} corresponding to the total normalized effective capacitance C_{total}/C_L (equation (2.1)) that each time the driver needs to charge.

2.3 Examples of Transition Delays

The plots in Figure 2.4 give us a visual impression of coupling between neighboring bus lines and the effect of different transitions to the transient response of the state of the bus. The circuit used for the simulations, is an eight-line bus, where one inverter drives each line.

The solid line in the graphs is the input signal applied to the inverter that drives line 4. The dashed line is the voltage of bus line 4. Each time the input signal of that line makes an 1 to 0 transition. However, the transitions in the neighboring lines results in transitions for line for that belong to different delay classes. We carefully chose the transitions to produce all the transition delay classes, as shown in the first five graphs. The last graph plots all the previous delay classes together.

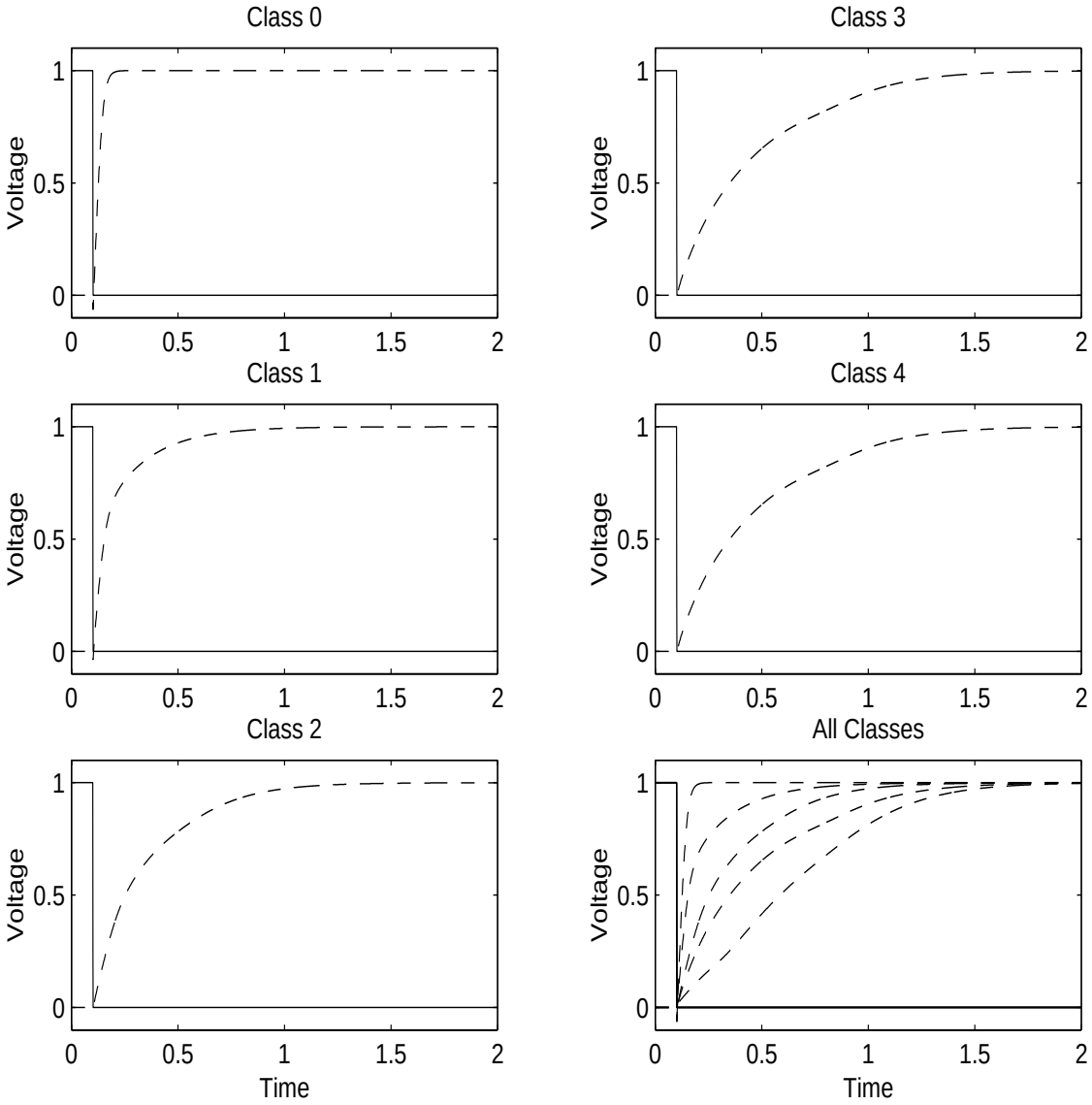

Figure 2.4: Coupling Effect on Different Transitions

For the specific metal line that was layed out and for the specific bus line spacing, λ is approximately seven. The last graph includes the outputs of all the transitions, for every class.

2.4 Grouping of the delay classes

We give, as an example, the delay classes per transition for a 3-line bus in Figure 2.5 The u_1^o,u_2^o,u_3^o correspond to the old state of the bus and the u_1^n,u_2^n,u_3^n correspond to the

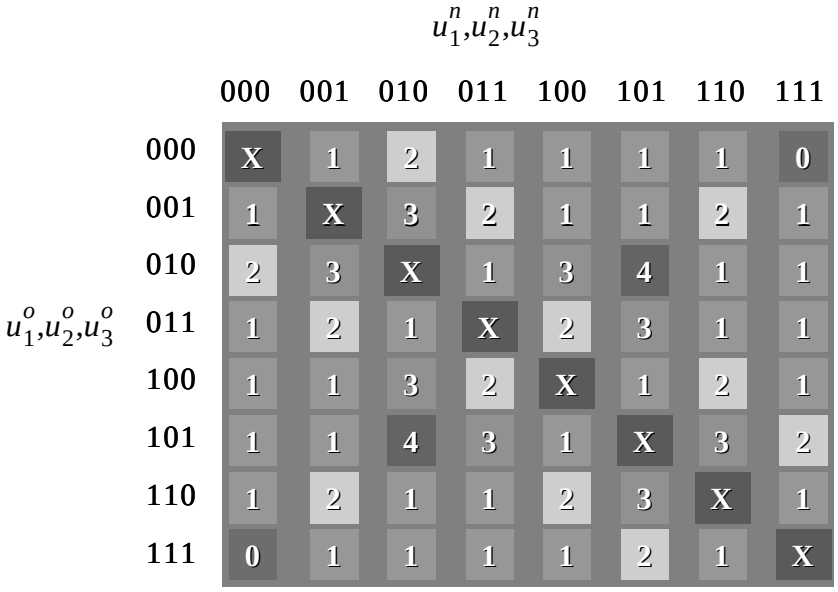

Figure 2.5: Transitions and Delay Classes for a 3-line Bus [3]

new state.

The two most time consuming transitions appear, as expected, when the intermediate line makes the opposite transition to the neighboring lines. This means that these transitions should be eliminated in order to speed up the bus. For even better delay savings, the eight transitions that give delays of the M3 class should also be eliminated.

The figure clearly shows that the distribution of the different delay classes is not even. The class with the worst case delay [4], only appears twice. However, it is this delay that determines the frequency of the whole bus.

2.5 Coding for speed Increase

In the traditional operation of data buses, the operation frequency is adequate so that **all** transitions on the bus are completed. However, we can reduce bus delay by eliminating slow transitions [5], [6].

Instead of sending the actual data to the bus, we encode it. The encoder outputs depend on the input data and on the present state of the bus. The encoder is responsible for allowing only the transitions of specific delay classes. We have to introduce some redundancy on the bus lines, because we have to be able to represent all the values and all the transitions of the original data. This means that the number of extra lines of the extended bus has to be adequate to do so.

On the receiver side, the reverse action occurs and the original data is restored after the decoder. The decoder is fed with both the new and the previous state of the bus and recovers at the outputs the original data.

Chapter 3

Design Implementation

This chapter presents a practical implementation of coding to increase speed. The overall operation is described, and then each component is presented in detail separately. Also the additional peripheral circuit is described in detail, to explain the interface of the core with the I/Os. The last section presents simulation results and comments on the performance of the implementation.

3.1 A practical implementation

This section presents a new practical implementation and Figure 3.1 shows the block diagram.

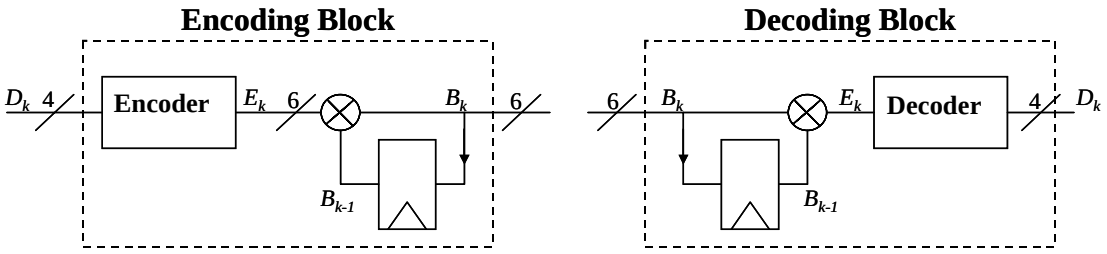

Figure 3.1: A Practical Implementation

For the purpose of this thesis the data on a 4-line bus is encoded to a 6-line bus. D_k is the data that will be encoded and E_k is the 6-bit data that is produced at the ***Encoder*** outputs. B_k represents the current state of the bus and B_{k-1} the previous one. When new data D_k arrives, the produced E_k are XORed with B_{k-1}. The operation will give the new state of the bus B_k.

The ***Encoder*** performs a mapping of the input data to six bits that don't have adjacent bits set to 1 at the same time. If, for example, two neighboring lines of the bus had opposite values, then an XOR operation with two 1's would force the state of both lines to change. So these two lines would be forced to make opposite transitions. Not allowing two adjacent ones at E_k prevents this from happening.

The ***Decoding Block*** is responsible for restoring the data. When the new state of the bus B_k arrives at the Block inputs, then an XOR operation with the previous state of the bus B_k gives E_k. The ***Decoder*** performs the exact opposite mapping of the ***Encoder***, so the original data D_k is restored.

3.2 Encoder Block

A 4-line bus can have 16 different states. All these states have to be mapped as explained above. A bus with five lines doesn't have enough number of states with no adjacent ones, to represent all 16 states. Therefore a six line bus was used, which gives 21 states, that don't have consecutive 1's. There are 5 states that aren't utilized. This gives a very broad number of possible implementations ($\frac{21!}{5!}$).

In order to cope with the different possible implementations, bit *E0* was hardwired to bit *D0*. So the problem was reduced to mapping a three line bus to a five line bus with an additional constraint (*E1* can't be 1 when *E0* is). Designing this encoder, instead of the previous one was a much simpler task, which allowed some experimenting on different possible implementations.

Each time the objective was to have the minimum number of gate stages from the encoder inputs to the encoder outputs.

Table 3.1 shows the mapping that was chosen for the implementation.

D3	D2	D1	D0	E5	E4	E3	E2	E1	E0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1	0
0	0	1	1	0	0	0	1	0	1
0	1	0	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	0	1
0	1	1	0	0	0	1	0	0	0
0	1	1	1	0	1	0	0	0	1
1	0	0	0	1	0	0	0	0	0
1	0	0	1	1	0	0	0	0	1
1	0	1	0	1	0	0	1	0	0
1	0	1	1	1	0	0	1	0	1
1	1	0	0	1	0	1	0	0	0
1	1	0	1	1	0	1	0	0	1
1	1	1	0	1	0	1	0	1	0
1	1	1	1	0	1	0	1	0	1

Table 3.1: Static Mapping Truth Table

The first 4 columns of Table 3.1 (*[D3, D2, D1, D0]*) are the values of the original data, were the last 6 (*[E5, E4, E3, E2, E1, E0]*) correspond to the new encoded ones. As mentioned before, bits *D0* and *E0* have the same values for every state.

The encoder block is shown in Figure 3.2.

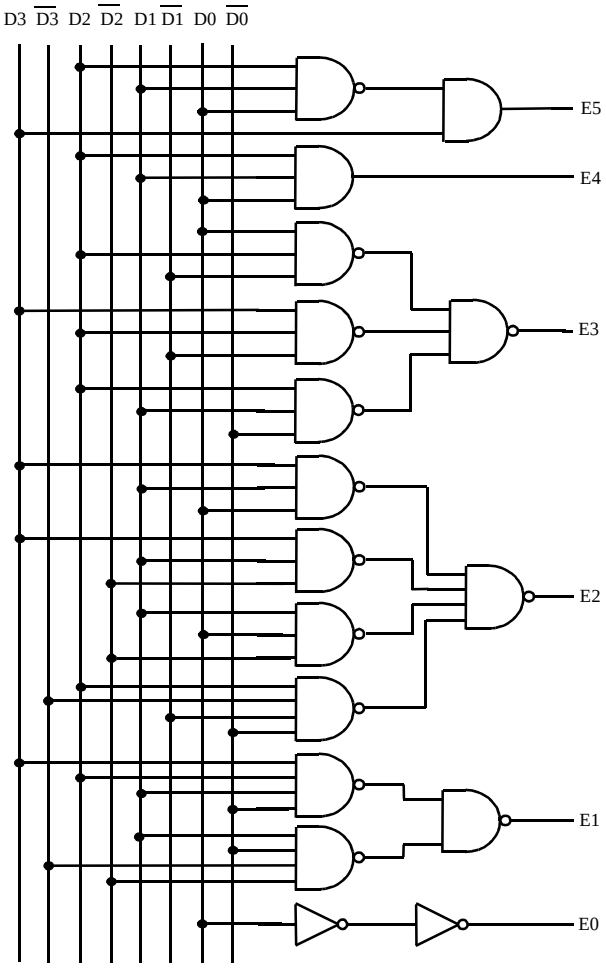

Figure 3.2: Encoder

Each output is a combination of the four input data and their complimentary. The maximum number of gate delays in this case is 4 gate delays and happens when E5 changes value through the NAND and AND gate.

3.3 Decoder Block

The decoder block is analyzed in this section and is shown in Figure 3.3. It performs exactly the inverse mapping operation from the encoder block. So at the outputs the original data are restored and we can get the original 4-bit data from the extended 6-line data

bus.

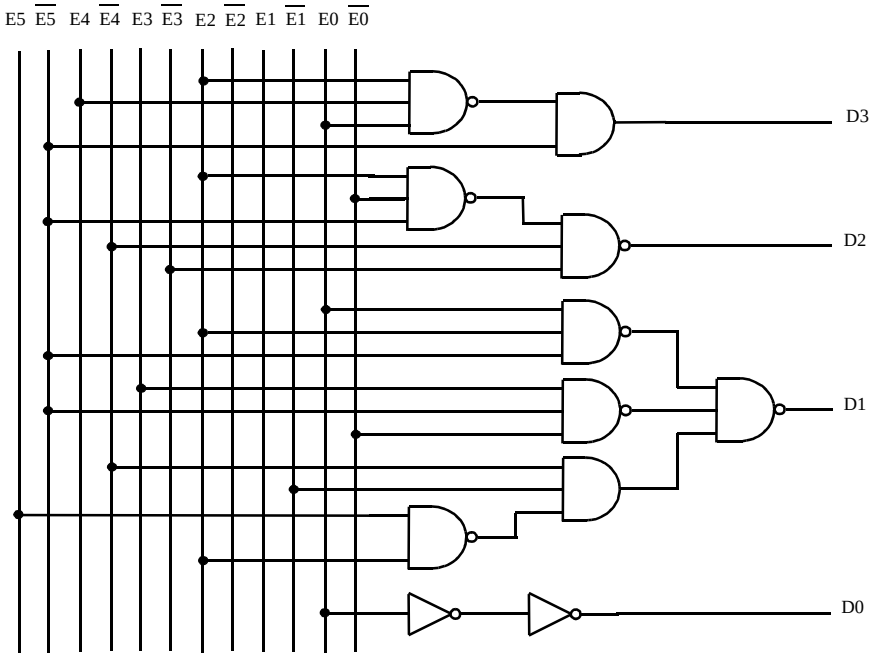

Figure 3.3: Decoder

Again, the maximum number of gate delays is four, as it is on the encoder block and happens when D1 changes value through the two NAND gates and the one AND gate.

3.4 Driver and Receiver Circuit

The ***Driver Circuit*** includes, besides the driving inverters, logic for resetting the state of the bus, so that the coding mechanism can start. Actually, from the 5 non utilized states mentioned before, the state [010000] was chosen to be the one that resets the bus. This was essential, because the initial state of the bus must be ensured to be acceptable (i.e. when the new data arrives at the XOR inputs, then the operation won't give a transition with delay greater than the ones that belong to the M2 delay class).

Since we want to be able to reset the lines to both V_{DD} and *Ground*, there are two driver blocks with some different logic circuitry.

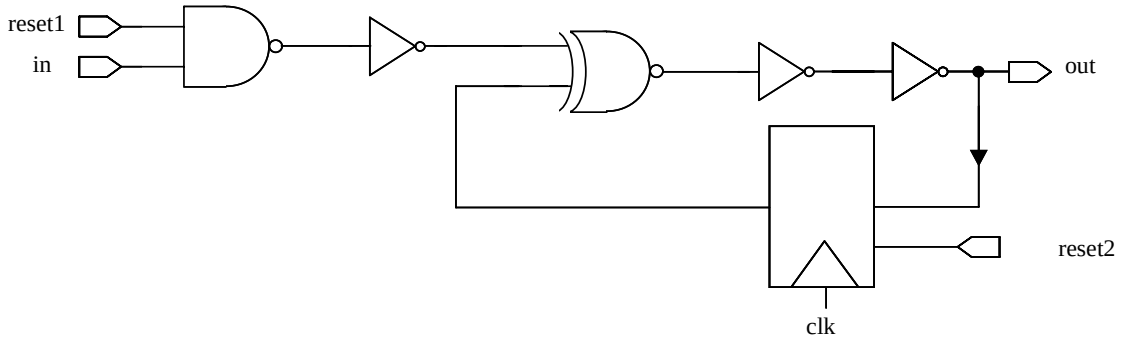

Figure 3.4: Driver Block with Modified Register

The modified register in Figure 3.4, differentiates the driver blocks. The modified register is explained in detail below. Although there are two reset signals shown in the driver block, there is actually only one. Depending on the value, that we want to set the line to, reset2 is either reset1 or inverted reset1.

In any case, when reset1 is high, the XOR operation is performed between the input and the old state of the line.

The two modified registers are shown in Figure 3.5

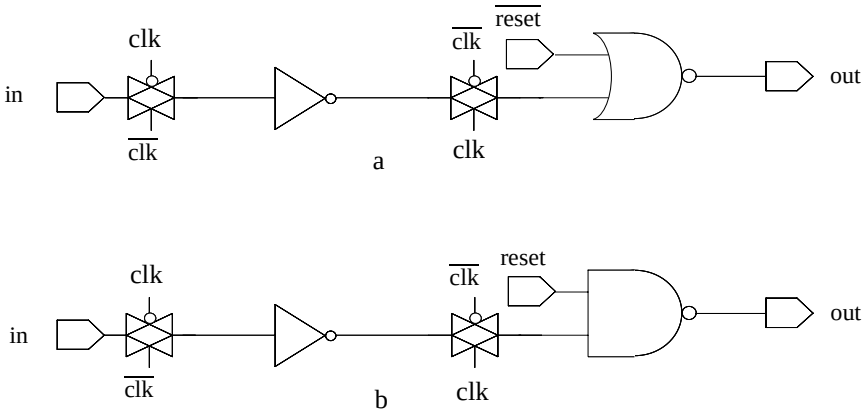

Figure 3.5: Modified Fully Dynamic Registers

When reset is high, the two registers act as normal fully dynamic registers. When reset is low, the register output becomes 0 and 1, for cases *a* and *b*, respectively. The output of

the NAND gate of the driver block (Figure 3.4) is always 1, which means that in order to reset a bus line to 0, a modified register of type a (from Figure 3.5) should be used in the driver block.

Since our extended bus has 6 lines, we need six driver blocks to form the driving circuitry for our implementation. As previously mentioned, the reset state of the bus was chosen to be [010000]. This means that 5 fully dynamic registers of type a and 1 of type b were used.

The ***Receiver Block*** is much simpler. It consists of six blocks shown in Figure 3.6.

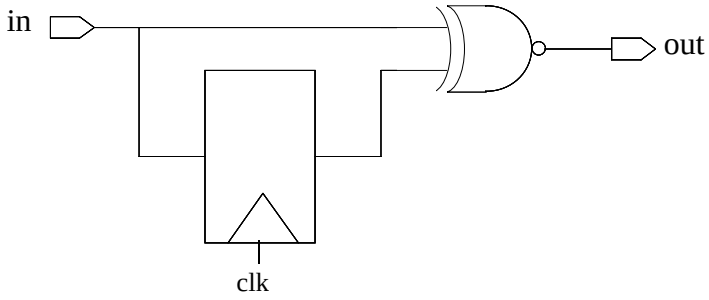

Figure 3.6: Receiver Block

The register shown is a normal fully dynamic register.

An XOR operation is performed between the new and the old state of the bus. So at the output of the XOR, the data produced originally by the encoder block will be restored. It is interesting to note that when the reset signal is asserted, the output of the Receiver Block cannot be determined. However, after the first rising edge of the clock it will be reset to [0,0,0,0,0,0] since the inputs of the XOR will hold the same data. For this reason, a special register is not needed for this block.

3.5 Test Structure

In order to test and measure the performance of this coding scheme the test structure

shown in Figure 3.7 was built. In *a* there are four pipelined stages. The encoder, the driver**,** receiver blocks and the decoder are pipelined with the four registers. The symbol between the Driver and the Receiver Blocks represents the bus.

This is the core circuit and has 9 total inputs and 8 outputs. 4 inputs for the circuit that implements the approach of Figure 3.1, 4 inputs for the uncoded case and 1 input for the Clock that synchronizes the registers. There are, as expected, 8 outputs that correspond to the outputs of the 2 cases that are tested.

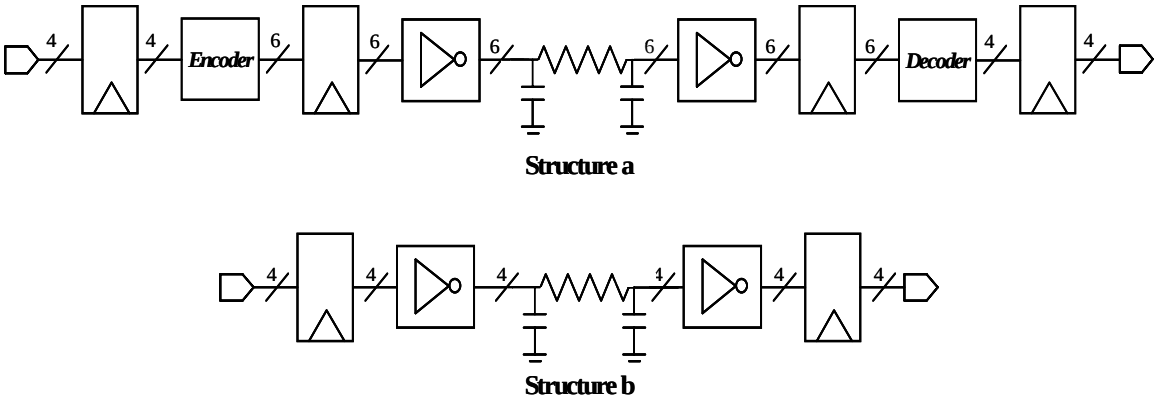

Figure 3.7: Test Structure - Core Circuit

For each structure, the bus is fed with data that will produce transitions that have the maximum delay times.

The inverters that were used to drive the bus for Structure B, are identical with the inverters shown in Figure 3.4 and were used to drive the bus for Structure A.

3.6 Peripheral Circuit

This section describes the blocks that had to be built so that the test structure could interface through the chip input and output pads.

3.6.1 Input Shift Register Block

Due to I/O limitations, the number of I/O pads had to be minimized. As a result the data had to be fetched serially through a single pad. Then, on chip the serial data is parallelized using the circuit shown in Figure 3.8.

The shift register is clocked using a different clock from the rest of the circuit. This modification was done for two reasons. The first one is isolation from the input/output pads. After the shifting of the data is complete, that is, all the input serial data is parallelized, the clocking of the shift register is stopped and won't affect the data of the register. The data is then stored at the register outputs.

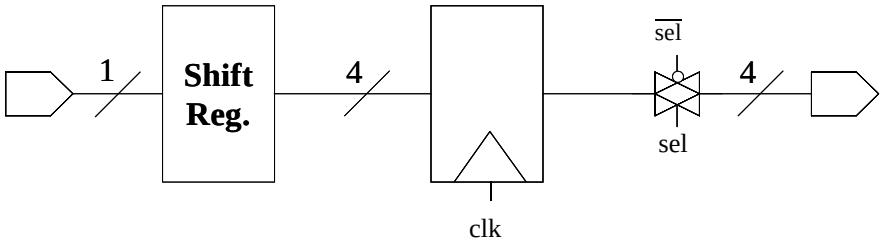

Figure 3.8: Serial to Parallel Circuit

The second reason was clock loading. Using a different clock, the core clock is released from the additional loading of the input shift and output shift (described below) registers.

This cell is used 4 times, because there is a need for four 4-bit latched data to feed into the test structure. Only one pad is needed however, and that is accomplished by having the bit, which is shifted out from one shift register, hard-wired to the input of the next shift register.

The transmission gates at the multiplexer output are used to choose the data from one of the four registers in the block.

3.6.2 Structure Selection Switch

After the data is latched at the outputs of the multiplexer of Figure 3.8, the circuit shown in the next figure follows. It is a switch, consisted of two multiplexers, that directs the

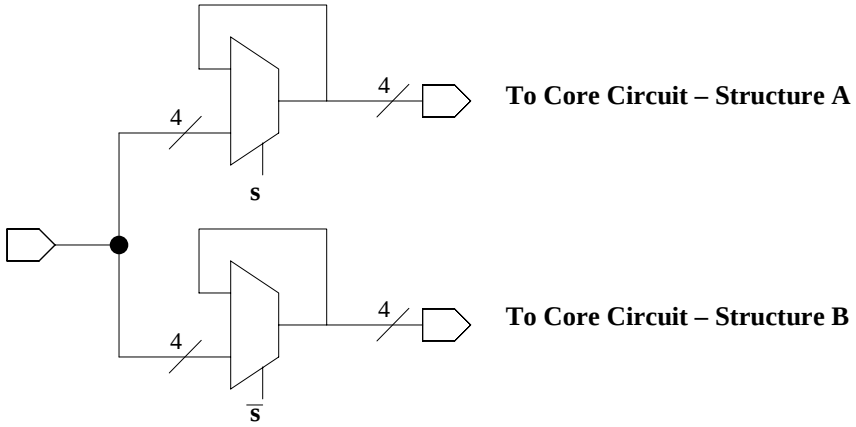

Figure 3.9: Switch

input data (after they are parallelized) to the data input registers (par. 3.6.3 p. 34) in structure a and structure b.

Bits *s* and $\bar{s}$ are responsible for making the selection, and the data is fed to the appropriate data input registers in the two different test structures. This block was designed so that only one input shift register block is needed to feed both structures, and consequently only one input pad.

3.6.3 Data Input Registers

This section describes the ***Data Input Registers*** that follow the structure selection switch and hold the data that will be tested. The outputs of this block is connected to the encoder block for structure a and the inputs of the driver inverters for structure b.

The transmission gates in Figure 3.10 are necessary, because we need to have the latched values from the input shift register block outputs, at the inputs of the four registers.

So we first clock the value for register 4 and then register 4 is isolated from register 3.

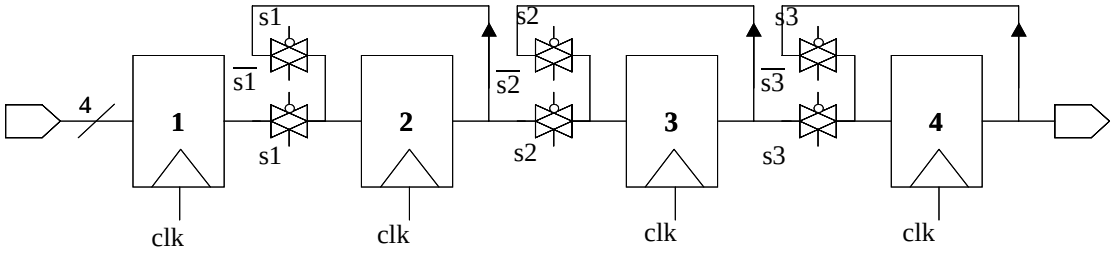

Figure 3.10: Data Input Registers

This mechanism is continued, and each time one more register is isolated, until all the registers are fed with the appropriate values.

This procedure will have no effect on the bus state, because the reset signal is asserted through the whole procedure.

All these blocks were designed to provide the core with the necessary input signals. The two blocks presented next, appear after the core circuit outputs and are developed to serialize the output data, so that only one output pad is required to give us data.

3.6.4 Output Register Block

The ***Output Register Block*** is shown in Figure 3.11. The multiplexer inputs come directly from the core and are connected to the outputs of the two structures. The control signal for the multiplexer is set before the whole procedure starts and is the same selection bit that is

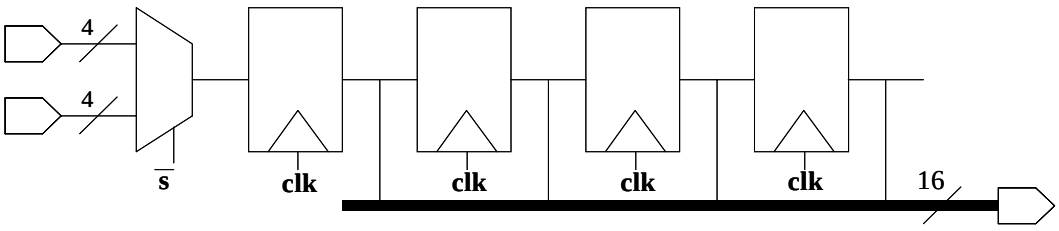

Figure 3.11: Output Register Block

used in the structure selection switch block, Figure 3.9.

When all the register hold valid data, the clock is stopped and the 16 bits are stored at the outputs of the registers.

3.6.5 Output Shift Register

The last block is a shift register that gets the 16 outputs from the previous blocks and shifts the most significant bit on each clock cycle. As expected, the clock signal is the same as the clock of the input shift register.

3.7 Simulation and Results

This section describes in more detail the operation of the core circuit and examines the general performance of the implementation.

The technology that was used is 0.13μ CMOS with dual threshold devices. The criterion for the performance of each of the structures shown in Figure 3.7, is that the output serial data has to be exactly the same with the input serial data. So there is no corruption due to bus delays and that the operating frequency is appropriate for the driver-receiver operation. In order to quantitatively evaluate the performance of the two structures, different clock periods were used to clock the circuit. As we kept shrinking down the clock period, at some point the voltage level at the end of the bus wasn't appropriate to give a valid output signal.

In structure a the bus consists of six lines spaced at the minimum allowed distance. In order to have a "fair" comparison for the two cases, the four lines of the bus in structure b were spaced so that the total spacing area of the buses in each case would be the same.

The input signals transition from state [0000] to state [0010] and back to [0000]. The reset signal goes to 1 with the second rising edge of the clock, right after the new data has arrived. The state of the bus before the second rising edge of the clock is [010000] (reset state) and the new data at the Encoder outputs is [000010]. So after the XOR operation the

new state of the bus will be [010010]. There is a transition from 1 to 0, at the second low order bit of the bus. Since the adjacent lines stay at 0, the transition delay for this line belongs to M2 class, which is the worst case delay.

The graphs in Figure 3.12 show the simulated results for structure a, where the data is encoded and then recovered at the receiver side.

The top and middle waveforms show the transitions of the input and the output respectively, that change value. The pipeline latency for the structure is four clock cycles (Figure 3.7). In all the graphs, the dashed waveform represents the clock.

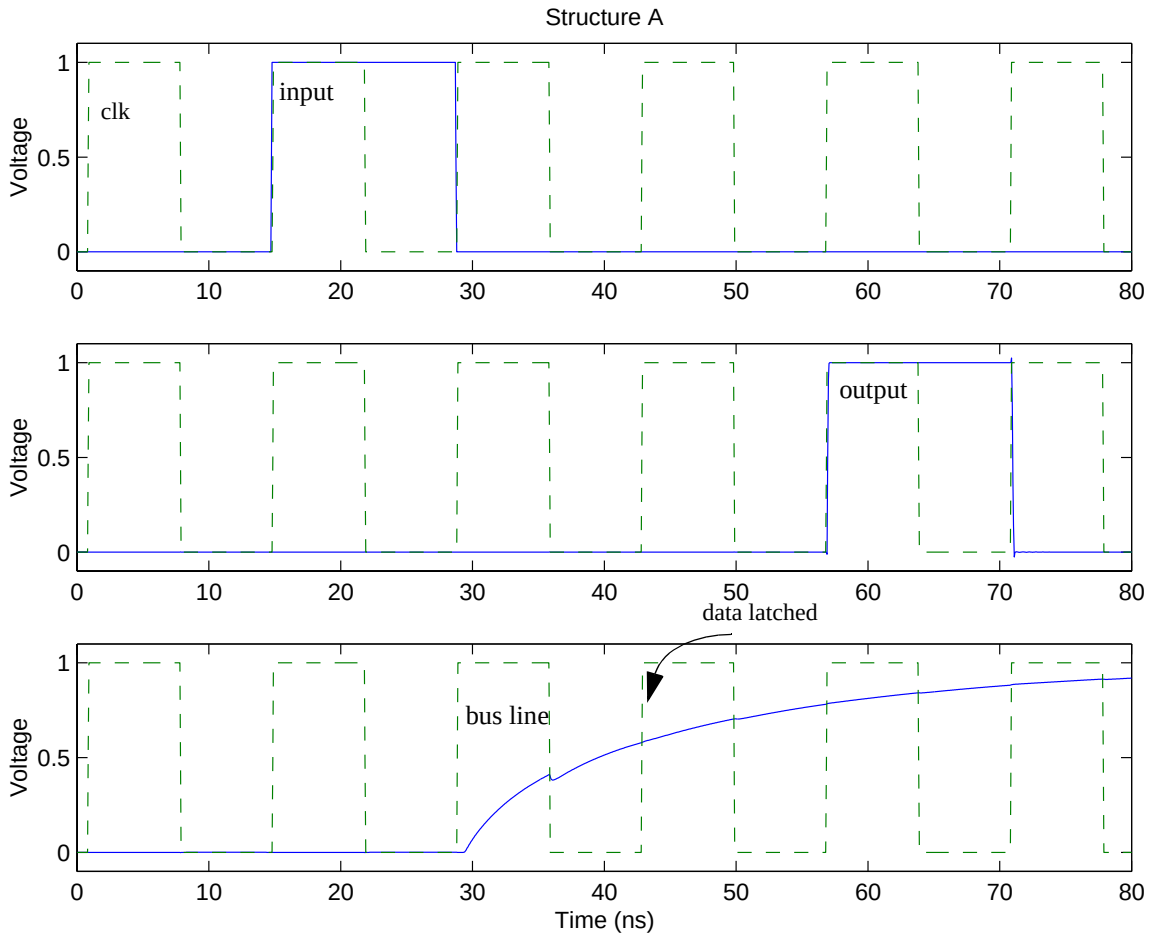

Figure 3.12: Coded Case - Correct Operation

The bottom waveform represents the bus line that makes the transition from 0 to 1. One clock cycle after the new input data has arrived, the bus line starts transitioning. After an additional clock cycle, the new data is latched at the receiver circuit. Although, the input signal returns to 0 after one clock period, the corresponding bus line doesn't follow this transition and continues to rise to 1. At the following rising edge of the clock, the input transition appears at the output of the structure (middle waveform).

If the clock cycle is not sufficient enough to latch the right data after on period, then the input data will be restored at the outputs after an additional clock cycled. Figure 3.13, shows this case where the bus line voltage doesn't reach an appropriate value.

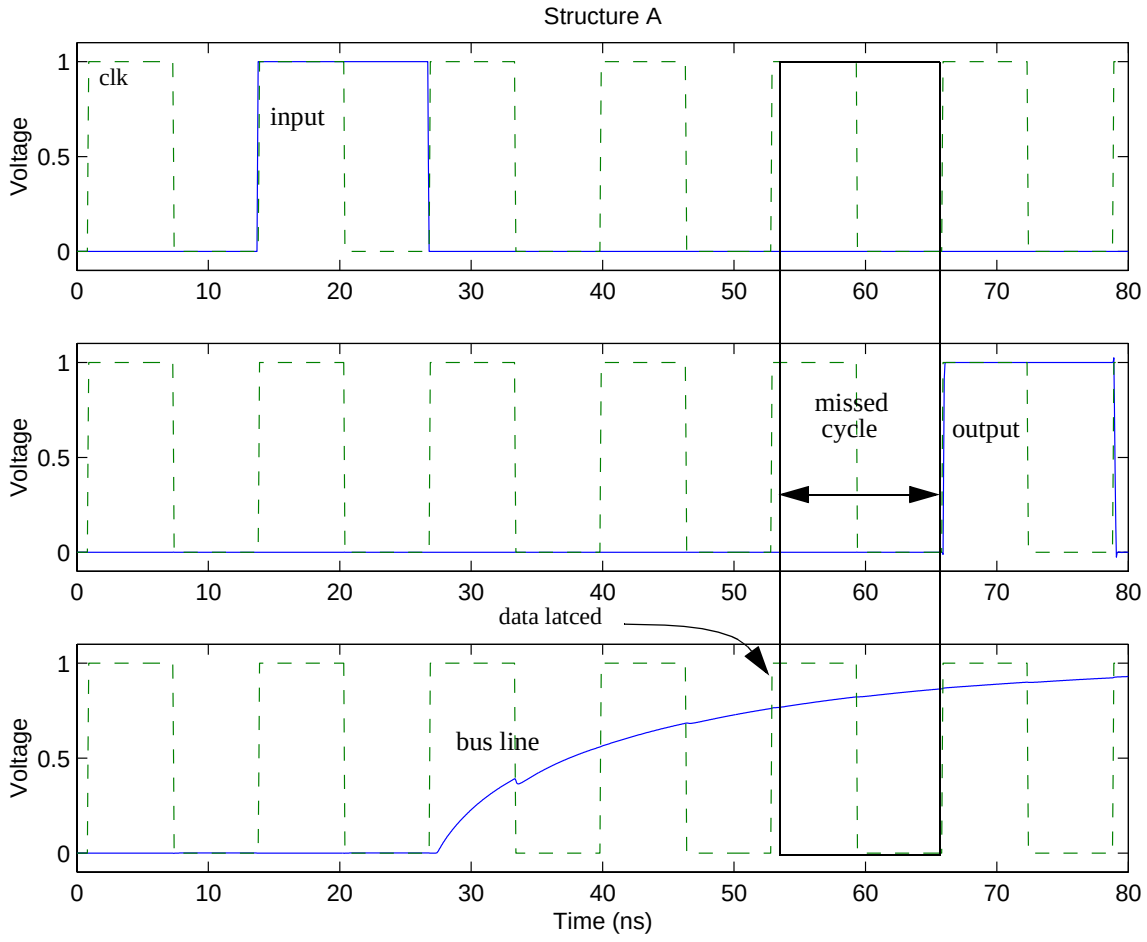

Figure 3.13: Coded Case - Faulty operation due to insufficient clock period

So the bus signal is misinterpreted by the receiver circuit. The output data (second waveform) follows the input data (first waveform) after 5 clock cycles. The missed cycle is shown in Figure 3.13.

The correct data is latched two clock cycles after the bus line starts transitioning.

Structure b uses the input pattern that gives the worst case transition delay. Adjacent input signals on every clock cycle make opposite transitions. The top waveform in Figure 3.14, shows the transitions on the second line of the four line bus and the middle one the corresponding output of structure b. The dashed line in all the graphs represents the clock signal.

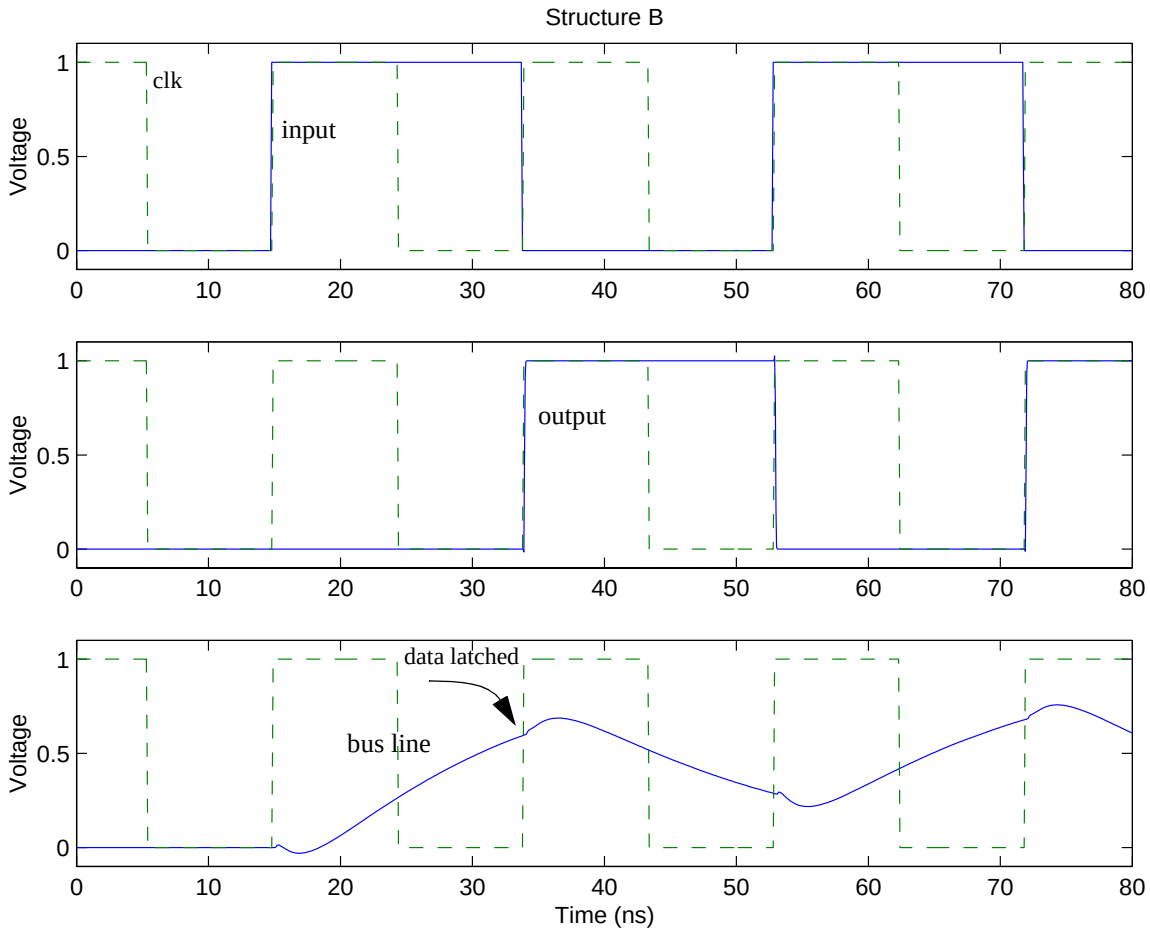

Figure 3.14: Uncoded Case - Correct Operation

The bottom waveform, shows the bus line signal. The voltage level on the bus line, one clock cycle after the input transition, is appropriate to give a valid output (the middle waveform follows the top after one period).

However, this is not true in the case shown in Figure 3.15. We increased the frequency and the input transition didn't appear on the output after one cycle. The clock period was not long enough for the bus line (bottom waveform) to reach an appropriate voltage level.

The missed cycle is shown in the figure, while the right data is latched after this missed clock cycle.

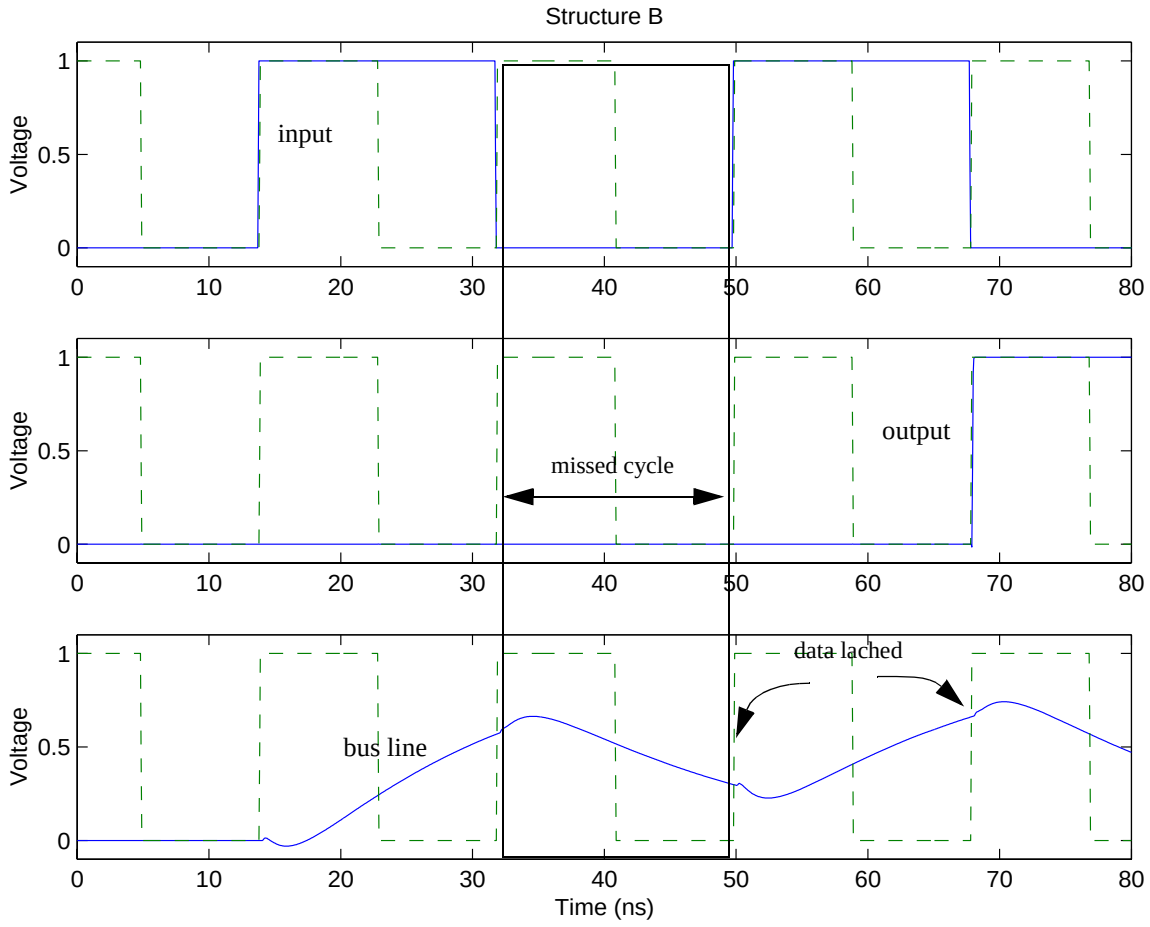

Figure 3.15: Uncoded Case - Faulty operation due to insufficient clock period

The total bus length, for each structure was 2.5cm. The length was chosen to be that big so that off-chip generated clock signals could be used to validate the simulation results. For this particular length, the maximum operation frequency for the coded case was 71.4Mhz and for the uncoded case 52.6Mhz, resulting in a net frequency increase of 36%.

Despite the significant speed increase that was achieved using coding, it is obvious from the discussion above that there is a latency penalty of two clock cycles that we pay, as a trade-off for the increase in throughput.

3.8 Layout

This section presents the layout blocks of the encoder, the decoder, the driver circuit and of the top core cell.

The area of the blocks that are used for the implementation are given at the appropriate figures. The total area penalty for this implementation is 7800 μm^2.

3.8.1 Encoder

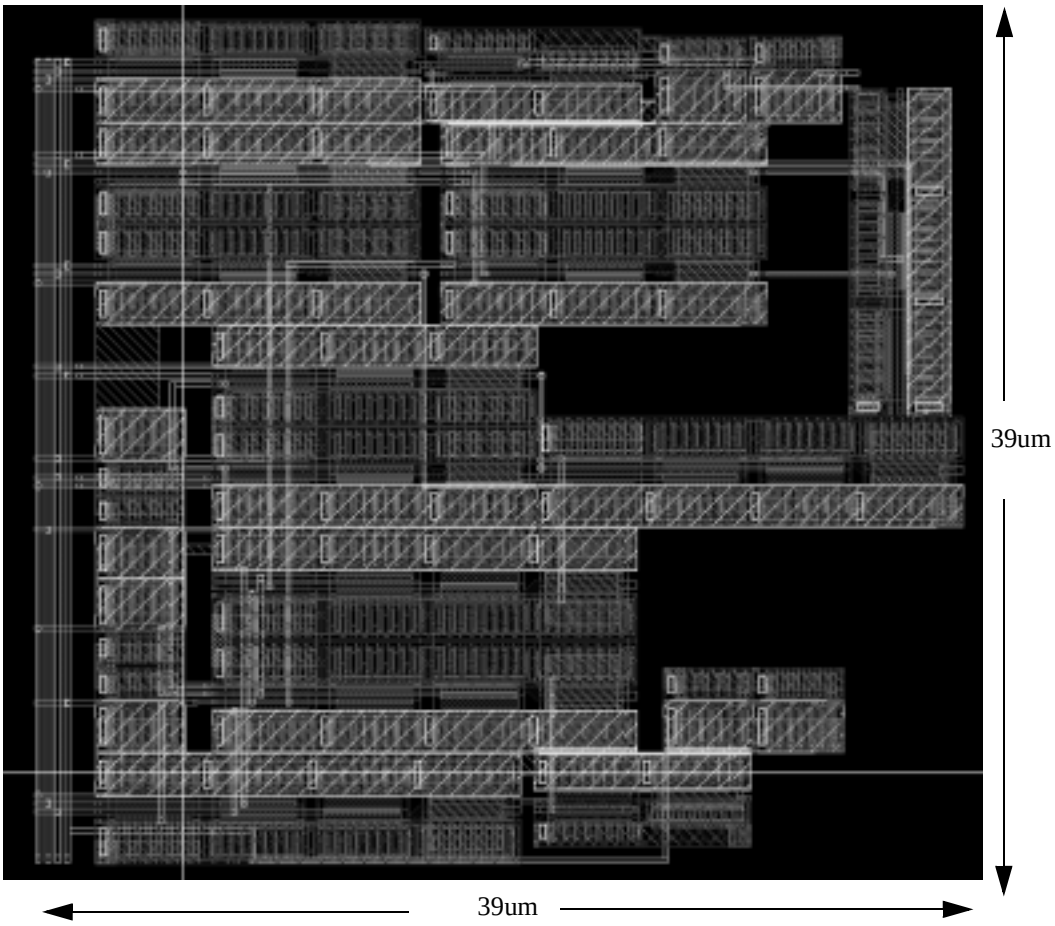

Figure 3.16: Encoder Block

3.8.2 Decoder

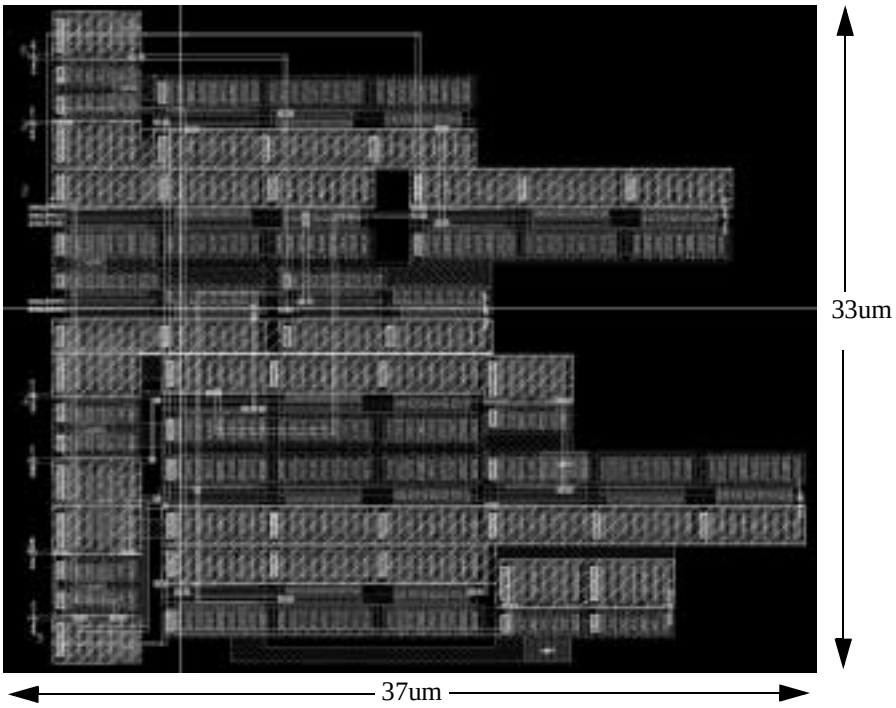

Figure 3.17: Decoder Layout

3.8.3 Driver and Receiver Circuits

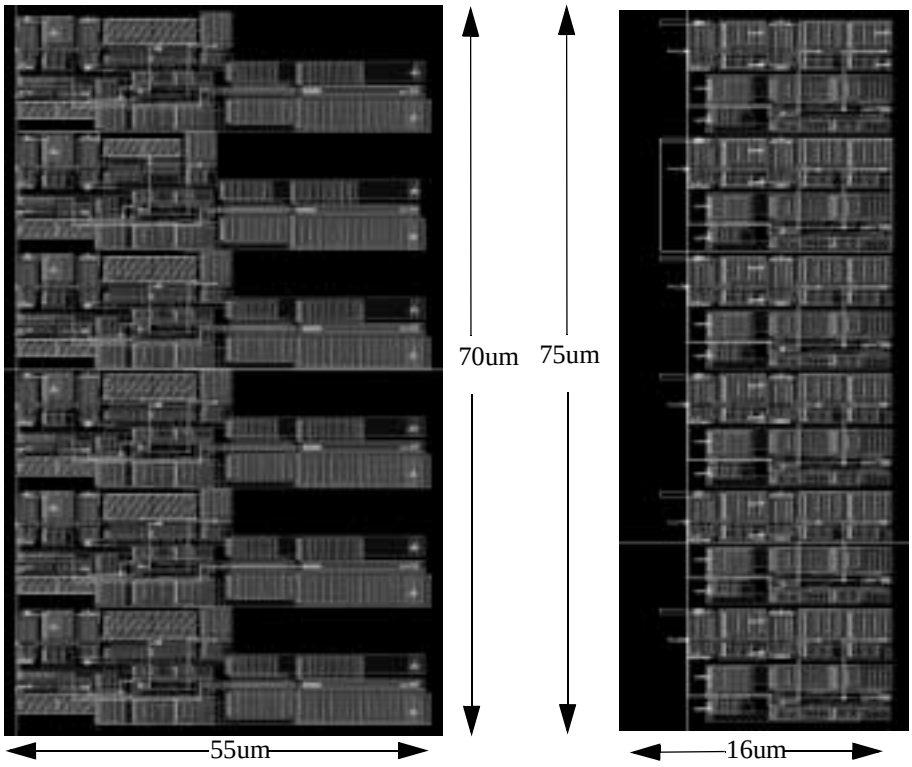

Figure 3.18: Driver and Receiver Circuit

3.8.4 Core Layout

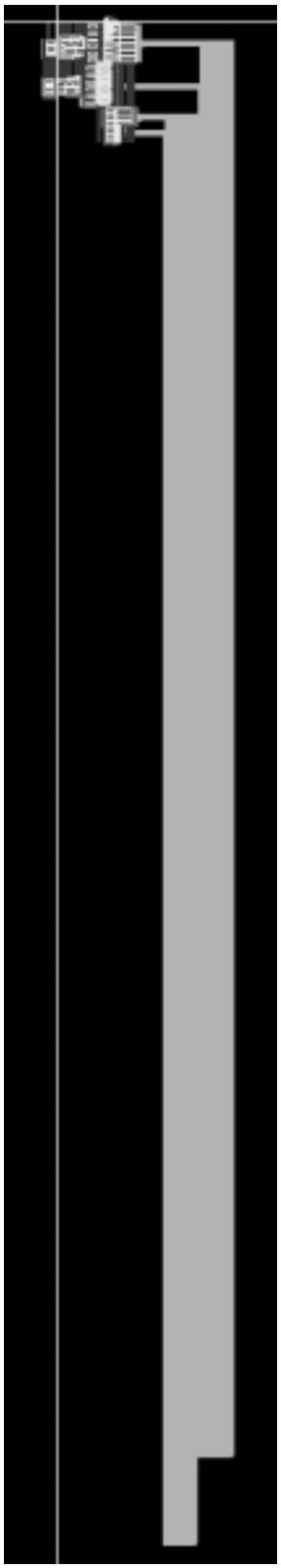

Figure 3.19: Core Layout with Bus

Chapter 4

Power Reduction in Data Buses

The proceeding sections provide a brief overview of power dissipation which will be a helpful background for understanding the motivation behind the approach that is described later.

4.1 Power Consumption Components

There are different sources of power consumption for CMOS integrated circuits that have been analyzed and are explained below: dynamic power, short-circuit power and leakage power.

$$P = Pdynamic + Pshort + Pleakage + PStatic \quad (4.1)$$

4.2 Dynamic Power

The dynamic power component in (4.1) is responsible for the largest portion of power dissipation and comes from the charging and discharging of the capacitance C_L as shown in Figure 4.1

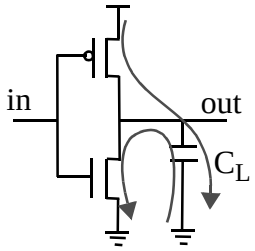

Figure 4.1: Charging and Discharging Capacitances

The dynamic power is given by

$$Pdynamic = a \cdot C_L \cdot V_{DD} \cdot V_{swing} \cdot Freq \quad (4.2)$$

The dynamic power is not a function of the transistor sizes but mainly is dependent on the total output capacitance and the input data (switching activity).

Reducing V_{dd} has a quadratic effect, however it has a negative effect on performance. As seen in (4.2), reducing the switching activity *a* and the operation frequency *Freq* we manage to reduce the dynamic power, however logic and architecture decisions impact these decisions.

4.3 Short-Circuit Power

In practical circuits the slope of the input signal is not infinite. This causes a direct current path between supply and ground for a short period of time, when both NMOS and PMOS devises are conducting. The power dissipation due to short-circuit can be minimized, if the input and output signal slopes are matched. In most cases, the short-circuit power approximately reaches 10% of the dynamic power.

4.4 Leakage Power

There are two main sources of leakage power, namely the reverse bias diode leakage and the sub-threshold conduction. Even if there is no switching at the outputs of a circuit, it will still consume power due to the reverse biased drain and source diodes of the non-ideal MOSFET devices.

4.5 Addressing Energy Issues in Buses

Although leakage power is becoming very significant as technology scales down the operating voltage and the device sizes, still dynamic power remains the most serious power consuming component in VLSI.

As explained in (par. 4.2 p. 45), the driving capacitance has a linear effect on the total dynamic power. Taking into account that there is an increasing need for driving large and long buses, then there is no surprise that over the past several years, significant emphasis has been placed on reducing the energy dissipation associated with on chip communication. Numerous

schemes have been presented for reducing energy associated with driving wires including low swing signaling [7],[8], charge re-cycling [9],[10],[11],[12],[13] and data coding [14],[15],[16].

This section summarizes the results that were presented on the above schemes, before the implementation, that we developed, is presented in the next chapter.

In (4.2) is shown that, if the bus is not charged and discharged with full swing, we can achieve dynamic power savings. Zhang *et. al.* [7] compares different low swing techniques, used for reducing dynamic power in integrated circuits. A thorough examination of those shows that some of the schemes have energy savings of a factor of seven using a benchmark interconnect circuit.

Khoo *et. al.*[10] has shown a theoretical energy savings of 47% and 59% for a 32 bit databus using charge recycling, using one and four charge transfers per bus access, respectively. Also, it is shown that the theoretical maximum energy savings on a bus of 32 bits wide is 72%, where random data is assumed.

A design for a charge recovery databus was also presented by Bishop *et. al.* [11], [13] and Lyuboslavsky *et. al.* [12]. Initial results showed an average power savings of 20% for an 8-bit charge recovery bus, based on realistic benchmarks, with the overhead of the control circuit estimated at 3.6% of the total power consumption. Extension on the work presented in [11], increases the energy savings to 28%, using 15 benchmarks and four high level coding schemes.

Stan *et. al.* [14] focuses on low-power techniques for communication in CMOS VLSI using *data encoding* methods. The encodings that are presented can decrease the power consumed for transmitting information over heavily loaded buses.

Using *Transition Pattern Coding* (Sotiriadis *et. al.* [15]) an energy savings of 50% is possible in submicron technologies. The coding strategy modifies the transition profiles to

reduce the switching energy. In the case where the coupling capacitance is significant, favoring specific transitions is a much more effective technique than trying to minimize the activity of the bus.

The coding schemes that were designed and described in [16], achieve a reduction in switching activity of a factor of 32%. For typical values of bus capacitances, this reduces the total power consumption by a factor of 36%.

4.6 The Charge Recycling Technique

A new practical Charge Recycling Technique (CRT) is introduced, appropriate for sub-micron technology buses. Its performance is verified by both mathematical analysis and circuit implementation. In sub-micron technology the strong capacitive coupling between the lines must be taken into account since it dramatically changes the energy consumption during bus transitions (with or without the CRT). For this purpose a sub-micron bus energy equivalent model is used, which is presented in [15] and shown in Figure 4.2.

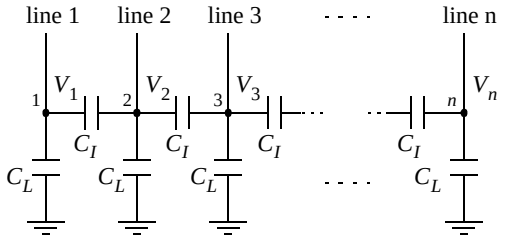

Figure 4.2: Sub-Micron energy-equivalent Bus-Model

In this section the two steps of CRT are presented. Suppose the bus has *n* lines and let T be the clock cycle period. New data is transmitted through the bus every T seconds. The time interval *[0,T]* is divided into the subintervals Int_1 and Int_2. The two steps of CRT are timely related as in Figure 4.3.

Suppose that during the clock cycle the bus transitions from its current values, $x=[x_1,x_2,...,x_n]^T$, to its new values $y=[y_1,y_2,...,y_n]^T$, (x_i,y_i correspond to line *i*). Normalizing by $V_{dd} = 1$, then all x_i and y_i belong to {0,1}. The voltages of the lines as functions of time $t \in [0, T]$

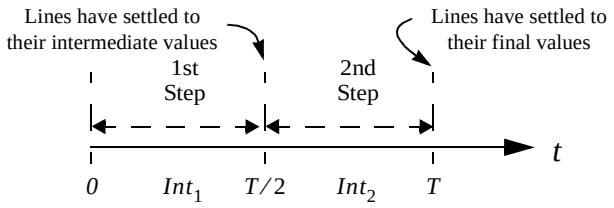

Figure 4.3: Timing of CRT

are denoted by $V=[V_1,V_2,...,V_n]^T$. At $t = 0$ and $t = T$ it is $V(0) = x$ and $V(T) = y$ respectively. The CRT is presented in Figure 4.4.with the modified driving circuit. We agree that switch w_i has value 0 if node *i* is connected to the output of driver *i*, and value 1 if node *i* is connected to the common node *q*. During Int_1 the lines that change logical values (during the transition $x \rightarrow y$) are connected to node *q* and not to their drivers. The lines retaining their logical values remain connected to their drivers. (This is a major difference to the strategy in [11]. If there is coupling between the lines it makes a difference if the non-changing lines remain connected to their drivers or not during the charge redistribution). During Int_2 all lines are connected to their drivers.

4.7 First Step (Int 1)

For every line $i = 1, \ldots, n$ we set $d_i = x_i \oplus y_i$. We also use the vector $d = x \oplus y$ where

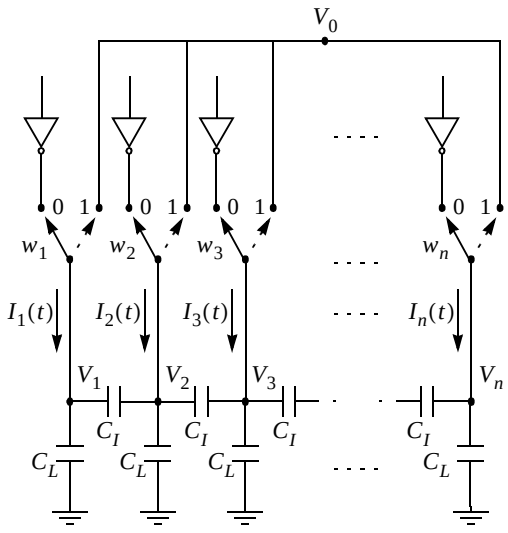

Figure 4.4: CRT - Network Connections

$d = [d_1, d_2, \ldots, d_n]^T$ and the diagonal matrix,

$$D = diag(d_1, d_2, \ldots, d_n) \tag{4.3}$$

During the transition, line i changes value if and only if $d_i = 1$. According to CRT, during the time interval $Int_1 = (0, T/2]$ the lines with changing values are connected to node q. The network is configured respectively. For example let $n = 4$, $x = [1, 0, 0, 1]^T$ and $y = [1, 1, 0, 0]^T$. Then $d = [0, 1, 0, 1]^T$ and during Int_1 the network is configured as in Figure 4.5.

During Int_1 the dynamics of the network satisfies the set of differential equations (see Figure 4.4),

$$\begin{aligned} I_1 &= C_L \cdot \dot{V}_1 + C_I \cdot (\dot{V}_1 - \dot{V}_2) \\ I_k &= C_I \cdot (\dot{V}_k - \dot{V}_{k-1}) + C_L \cdot \dot{V}_1 + C_I \cdot (\dot{V}_k - \dot{V}_{k+1}), \quad 1 < k < n \\ I_n &= C_L \cdot \dot{V}_n + C_I \cdot (\dot{V}_n - \dot{V}_{n-1}) \end{aligned} \tag{4.4}$$

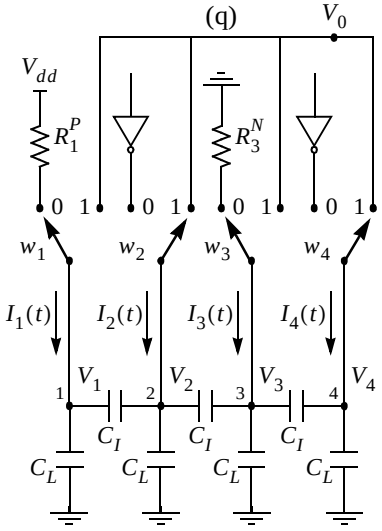

Figure 4.5: Step 1:Example (*n=4*)

The currents vector $I = [I_1, I_2, \ldots, I_n]^T$ is defined and the $n \times n$ capacitance conductance matrix [15] of the network in Figure 4.4,

$$C_T = \begin{bmatrix} 1+\lambda & -\lambda & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & & 0 \\ 0 & -\lambda & & & \\ & & & 1+2\lambda & -\lambda \\ 0 & 0 & \dots & -\lambda & 1+\lambda \end{bmatrix} \cdot C_L \tag{4.5}$$

$V(0) = x = [x_1, x_2, \dots, x_n]^T$ are the initial conditions of the lines and $V\left(\frac{T}{2}\right) = \left[V_1\left(\frac{T}{2}\right), V_2\left(\frac{T}{2}\right), \dots, V_n\left(\frac{T}{2}\right)\right]^T$ are the intermediate ones. Here we assume that the time length $T/2$ is sufficient for the voltages of the network to settle. This assumption is reasonable for the current technology and is always used in charge redistribution (and adiabatic) techniques. So for $i = 1, \dots, n$ the voltage $V_i\left(\frac{T}{2}\right)$ is either x_i if $d_i = 0$ or $z \equiv V_q\left(\frac{T}{2}\right)$ if $d_i = 1$. The value *z* is of course the same for all lines that change logical value. Algebraically we have that $V_i\left(\frac{T}{2}\right) = (1 - d_i) \cdot x_i + d_i \cdot z$ or in vector form that,

$$V\left(\frac{T}{2}\right) = (I - D) \cdot x + z \cdot d \tag{4.6}$$

where *I* is the $n \times n$ identity matrix. The matrix *D* and the vector *d* are as defined before. Since, $V(0) = x$ we have,

$$(d^T \cdot C_T \cdot d) \cdot z = d^T \cdot C_T \cdot D \cdot x \tag{4.7}$$

and

$$z = V_o\left(\frac{T}{2}\right) = \frac{d^T \cdot C_T \cdot D \cdot x}{d^T \cdot C_T \cdot d} \tag{4.8}$$

4.8 Energy Dissipation on Step 1

The energy **drawn from V_{dd}** on the first step of CRT is given in here. The current $I_{V_{dd}}(t)$ drawn from V_{dd} during (Int_1) is the sum of the currents drawn by the lines that do *not* change logical value during the transition *and* remain connected to V_{dd} through

their drivers, i.e the lines $i = 1, \ldots, n$ for which $x_i = y_i = 1$.

So, $I_{V_{dd}}(t) = \sum_{\substack{i:\ x_i = 1 \\ \text{and } y_i = 1}} I_i(t) = \sum_{i=1}^{n} x_i \cdot (1 - d_i) \cdot I_i(t)$ which can be written in matrix form as,

$$I_{V_{dd}}(t) = x^T \cdot (I - D) \cdot I(t) \tag{4.9}$$

(Symbol I is used for both the current vector and the identity matrix. Is should be clear what I represents each time).

The energy that is drawn is given by

$$E_1 = x^T \cdot (I - D) \cdot C_T \cdot (z \cdot d - D \cdot x) \tag{4.10}$$

And by replacing z from (4.8) into we have,

$$E_1(x, y) = x^T \cdot (I - D) \cdot C_T \cdot \left\{ \left(\frac{d^T \cdot C_T \cdot D \cdot x}{d^T \cdot C_T \cdot d} \right) \cdot d - D \cdot x \right\} \tag{4.11}$$

if $d \neq 0$ and $E_1 = 0$ if $d = 0$.

4.9 Second Step (Int 2)

During the second step of the CRT, the time interval $Int_2 = (T/2, T]$, every line is connected to its driver (with the new value y_i). So for all $i = 1, \ldots, n$ it is $w_i = 0$. For the example with

$n = 4$, $x = [1, 0, 0, 1]^T$ and $y = [1, 1, 0, 0]^T$, the network is configured as in Figure 4.6.

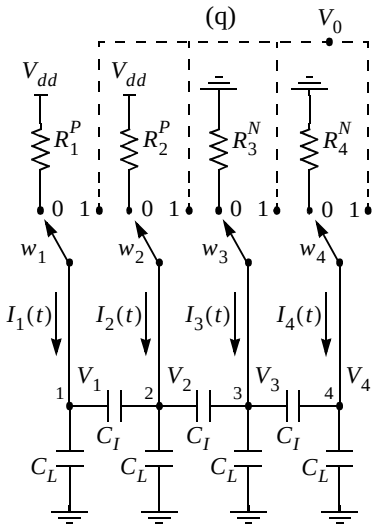

Figure 4.6: Step 2: Example *(n=4)*

4.10 Energy Dissipation on Step 2

During Int_2 the current drawn from V_{dd} equals the sum of the currents $I_i(t)$ of the lines connected to V_{dd} (through their drivers). So $I_{V_{dd}}(t) = \sum_{i\,:\,y_i = 1} I_i(t) = \sum_{i=1}^{n} y_i \cdot I_i(t)$ or in vector form,

$$I_{V_{dd}}(t) = y^T \cdot I(t) \tag{4.12}$$

and

$$I_{V_{dd}}(t) = y^T \cdot C_T \cdot \dot{V}_T \tag{4.13}$$

The energy drawn from V_{dd} on step 2 is: $E_2 = \int_{T/2} I_{V_{dd}}(t)dt$.

And the integral gives,

$$E_2 = y^T \cdot C_T \cdot \left(V(T) - V\left(\frac{T}{2}\right)\right) \tag{4.14}$$

Finally, $V(T) = y$, so we get,

$$E_2(x,y) = y^T \cdot C_T \cdot \left\{ y - (I-D)\cdot x - \left(\frac{d^T \cdot C_T \cdot D \cdot x}{d^T \cdot C_T \cdot d} \right) \cdot d \right\} \tag{4.15}$$

Complete derivations for the energy equations that were presented in the above sections can be found in [19]

4.11 Energy Properties of CRT

The total energy $E(x,y)$ drawn from V_{dd} during the transition $x \to y$ is of course $E(x,y) = E_1(x,y) + E_2(x,y)$. Using the identity $(I-D)\cdot x = (I-D)\cdot y$ and expressions (4.11) and (4.15) we get,

□ with CRT ▒ without CRT

		(y_1, y_2, y_3)							
		000	001	010	011	100	101	110	111
(x_1, x_2, x_3)	000	0	6	11	7	6	12	7	3
		0	6	11	7	6	12	7	3
	001	0	0	10	6	3	6	6.3	2
		0	0	16	6	6	6	12	2
	010	0	5.1	0	1	5.1	11	1	2
		0	11	0	1	11	22	1	2
	011	0	5	5	0	5.3	10	3	1
		0	5	5	0	11	16	6	1
	100	0	3	10	6.3	0	6	6	2
		0	6	16	12	0	6	6	2
	101	0	0	10	5.1	0	0	5.1	1
		0	0	21	11	0	0	11	1
	110	0	5.3	5	3	5	10	0	1
		0	11	5	6	5	16	0	1
	111	0	5	10	5	5	10	5	0
		0	5	10	5	5	10	5	0

Table 4.1: Transition energies with and without the CRT

$$E(x, y) = y^T \cdot C_T \cdot (y - x) + y^T \cdot D \cdot C_T \cdot (D \cdot x - z \cdot d) \quad (4.16)$$

where z is given by (4.8). The first term of the right part of (4.16) equals the energy drawn from V_{dd} by the bus during the transition $x \rightarrow y$ when no charge recycling is applied [15]. The other terms correspond to the energy difference (savings) due to CRT.

For a better intuition on how CRT influences the bus energy transition patterns, Table 4.1 presents the case of a three line bus $n = 3$ when $\lambda = 5$. Five is a representative value of λ for the case of 0.18μ technologies (with minimal distance between the wires). For simplicity we set $C_L = V_{dd} = 1$

For each transition $(x_1, x_2, x_3) \rightarrow (y_1, y_2, y_3)$ the shadowed value (below) is the energy cost *without* CRT, equal to $y^T \cdot C_T \cdot (y - x)$. The numbers on the white background (above) are the energies *with* CRT, i.e. to the values given by (4.16). The energy *with* CRT is always smaller. Also, the highest percentage of energy reduction occurs in the most expensive transitions $010 \rightarrow 101$ and $101 \rightarrow 010$ where adjacent lines transit in the opposite direction and the interline capacitances are charged by $2 \times V_{dd}$.

4.12 Energy Reduction

The result for the transition energy, equation (4.16), allows us to estimate numerically the expected energy drawn by the bus when the CRT is used. We do this for the case of uniformly distributed i.i.d. data. In Figure 4.7 we see the expected energy using CRT as a percentage of the expected energy without CRT for the cases of $n =$ *2,4,8,16,32,64,128,256* and $\lambda = 0, 5, 10$. The figure suggests that for the number of lines $n = 32, 64, 128, 256$ the energy drawn from V_{dd} can be reduced to one half using CRT. Also, the results are independent of the capacitance to ground C_L and they slightly improve when λ increases. In general λ tends to increase with technology scaling.

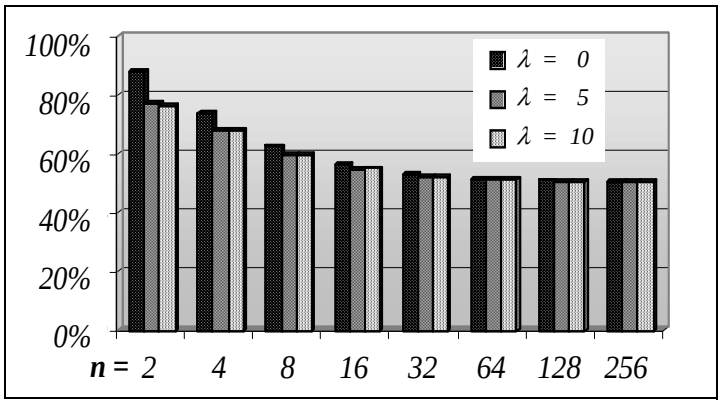

Figure 4.7: Energy with CRT / Energy

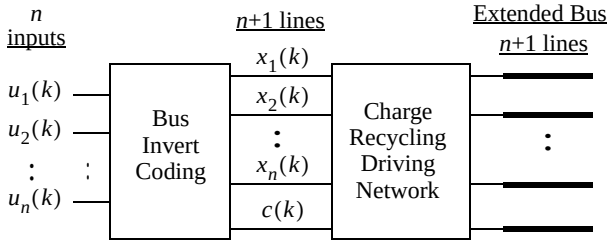

Figure 4.8: Combination of CRT with Bus Invert

4.13 CRT and Bus-Invert

In the previous sections we showed how CRT reduces energy consumption. In Figure 4.8 an architecture is presented, where CRT is combined with Bus-Invert coding [14].

The Bus Invert coding works in the following way. Let $u(k) = [u_1(k), u_2(k), \ldots, u_n(k)]^T$ be the new input vector and $x(k) = [x_1(k), x_2(k), \ldots, x_n(k)]^T$ be the new vector of the values of the lines. If the vector $u(k) \oplus x(k-1)$ contains more than $n/2$ ones then we set $x(k) = \overline{u(k)}$ and $c(k) = 1$, otherwise we set $x(k) = u(k)$ and $c(k) = 0$. This means that, instead of sending the actual date, the complementary data is sent and a control signal is set. On the receiver side, before the data is processed, an inversion of the data would be necessary if the control signal is asserted.

The combined performance of CRT and Bus Invert is shown in Figure 4.9. There is a small improvement compared to the results of Figure 4.7. For buses with 16 lines or more the energy saving is more than *50%*.

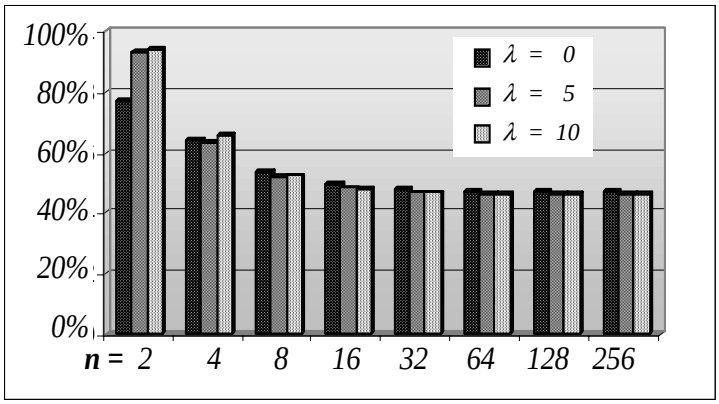

Figure 4.9: Energy with CRT and Bus Invert / Energy without them

4.14 A Circuit for CRT Drivers

To verify CRT, a circuit that implements the conceptual network of Figure 4.4 was designed and verified. The circuit implementation consists of the bus and the CRT drivers

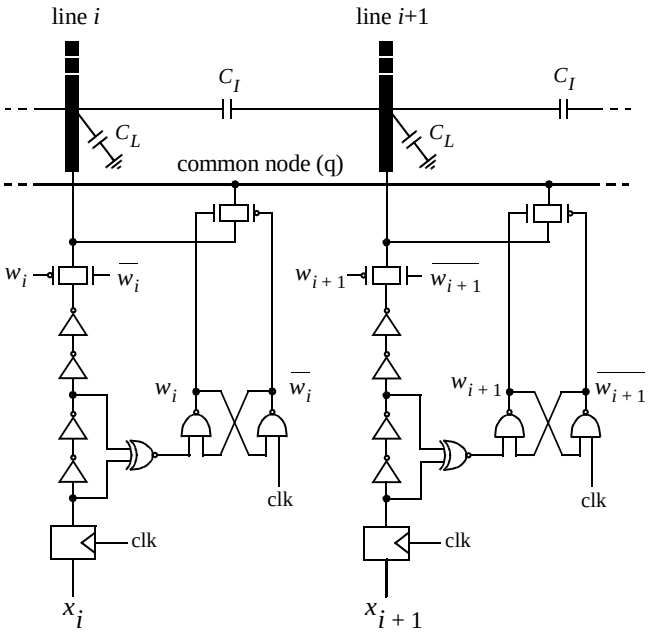

Figure 4.10: Efficient CRT - Driver

of the lines (Figure 4.10). The implementation is very simple and is explained next. If the CRT driver detects a transition, it connects the line to the common node (q). Otherwise the line stays connected to its regular driver (chain of inverters).

The CRT driver operates as follows. The switches w_1, w_2, … in Figure 4.3 are realized here by the pair of transmission gates. The charge recycling phase begins when CLK becomes 1.

If the input x_i changes value, the cascaded inverter delay makes a negative spike appear at the output of the XNOR gate as it is shown in Figure 4.11.

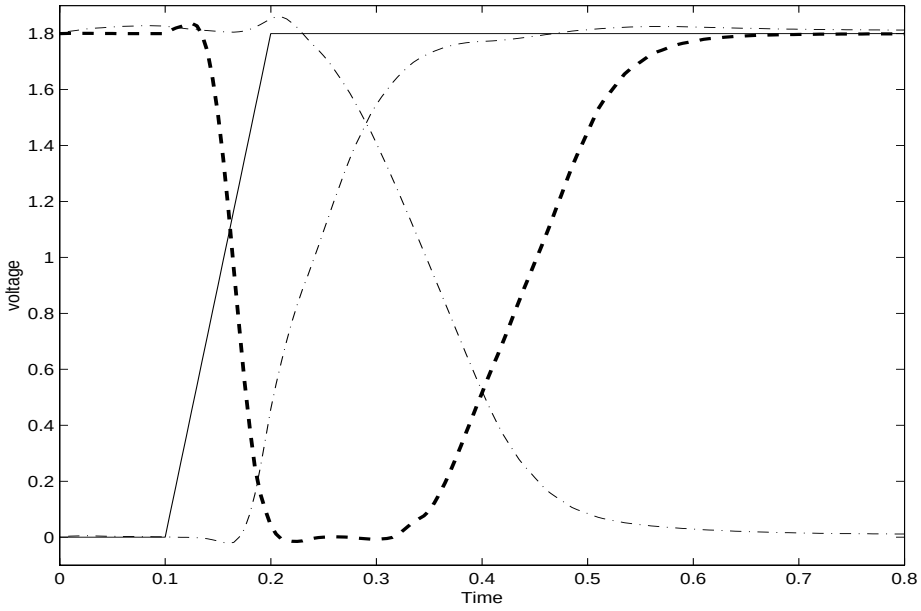

Figure 4.11: Transition detection

The XNOR output voltage is shown in the figure with the thick dotted line. The XNOR is designed to have a very fast fall time and a very slow rise time. The sizes of the NMOS and PMOS transistors were chosen so that the spike has the appropriate width to set the cross coupled NAND gates.

The dotted line that transitions from 0 to 1, is the output of the NAND gate denoted as w_i in Figure 4.10. The other dotted line corresponds to the output of the second NAND gate and drops

from 1 to 0. Note that the output of this gate crosses the rising edge of the spike below the threshold of the other NAND gate, so the change of the state is secured.

When the outputs of the NAND gates change value, each bus line is connected to the common node *q* through the transmission gate and is isolated from the driver. The charge recycling phase ends on the falling edge of the clock. This resets the NAND pair, isolates the line from the common node (*q*) and connects it to the buffer chain. If the input x_i does not make a transition, the NAND pair remains reset during the whole clock cycle and the line remains connected to the buffer chain. The same circuit can be used unchanged for buses with arbitrary number of lines.

The proposed CRT driver was designed and laid out in 0.18μ technology. Using this driver we tested the CRT for a 4-line and an 8-line bus. The layout of both the CRT and the standard drivers for the two cases are shown in Figure 4.12.

The control circuit that detects the transition and sets the switches could have been implemented using a XOR to produce a positive pulse that would set a cross coupled NOR gate pair. This specific implementation, however, would have resulted in a design that would require an increased amount of area. For a very small output rise time, large PMOS devices are needed for the XOR gate, for fast transition detections.

Figure 4.12: Layout of the CRT drivers

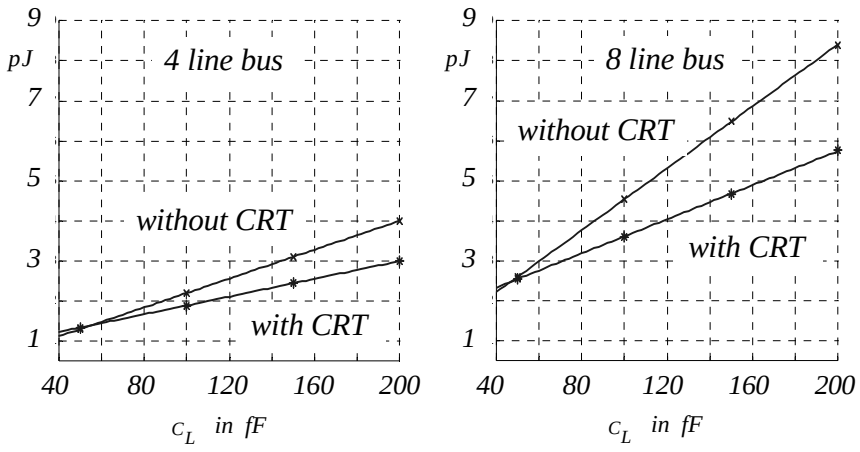

Figure 4.13: Average energy per cycle of a 4 and 8 line buses with and without CRT

4.15 Simulation and Results

The CRT drivers of Figure 4.10 were used to drive the lines of a four and an eight line bus, $n = 4, n = 8$. A netlist was extracted from the layout of the drivers for the simulation with HSPICE. The lines were modeled as in Figure 4.2 and for the capacitor C_L we used the values $50fF, 100fF, 150fF$ and $200fF$.

Note that these values could represent not only the line capacitors but all the loads that can appear on buses. This is particularly the case of reconfigurable interconnect networks (e.g. in FPGAs) where long buses are loaded by the parasitic capacitances of several mosfets resulting in total capacitive loads of the size of a few picofarads [2].

The clock frequency in the simulations is $100Mhz$ and the buses were fed with uniformly distributed i.i.d. sequences of data. In Figure 4.13, the average energy per cycle is shown for the four line bus (left) and the eight line bus (right). The curves in the graphs showing higher energy consumption correspond to the standard buses. The curves showing lower consumption correspond to buses with CRT drivers.

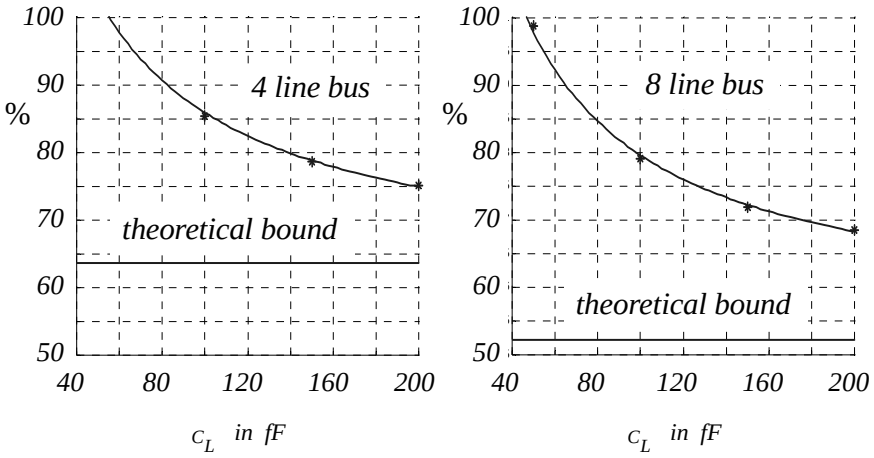

Figure 4.14: Normalized energy using CRT (HSPICE simulation)

In Figure 4.14 we see the average energy using CRT as a percentage of the average energy without CRT for the 4-line and 8-line buses. Again, the ratios are parametrized to C_L. The flat lines correspond to the minimum possible ratios resulting from the theoretical analysis and shown in Figure 4.7.

As it should be expected, for higher capacitive loads we get higher percentages of energy saving. This is because the average energy per cycle of the additional circuitry of the drivers is relatively independent of the loads. For larger loads this additional energy becomes less significant.

The energy savings for the case of a 4-line bus were up to 25% as the bus length increased and for the case of an 8-line bus up to 32%.

It is interesting to look at the waveforms of the individual lines during the two steps of the CRT. Figure 4.15 shows the waveforms of the line voltages of the 4-line bus. In this particular case, one line experiences a $1 \rightarrow 0$ transition and the rest three lines make a $0 \rightarrow 1$ transition. Since all lines transit, they are all connected first to the common node *q*.

The final voltage at node *q* during the charge redistribution period is *Vdd/4* and correspond to the converging point of the waveforms at time $T/2 = 5ns$.

Right after the positive edge of the clock, there is a small transition phase and is due to the fact that there is a little delay before the switches that connect the drivers to the bus are turned off. This is for the case that the line is changing value. We can see this on Figure 4.15, right after the positive edge of the clock. The gate voltage of the *p* and *n* transistor on the transmission gates are set after an XNOR gate delay and the delay of the cross-coupled NAND gates. However, it doesn't really affect the whole operation much since the driving capacitances are large, and the state of the bus practically stays unchanged for this short period.

It is interesting to note that for an individual transition the maximum energy saving with CRT occurs when all lines transition and adjacent lines transition in opposite directions. This generalizes to the fact that CRT preforms very well when the sequences of the transitions of adjacent lines are negatively correlated.

4.16 Design Challenges

It is clear from Figure 4.14 that, when the length of the bus is very small, the savings that we get

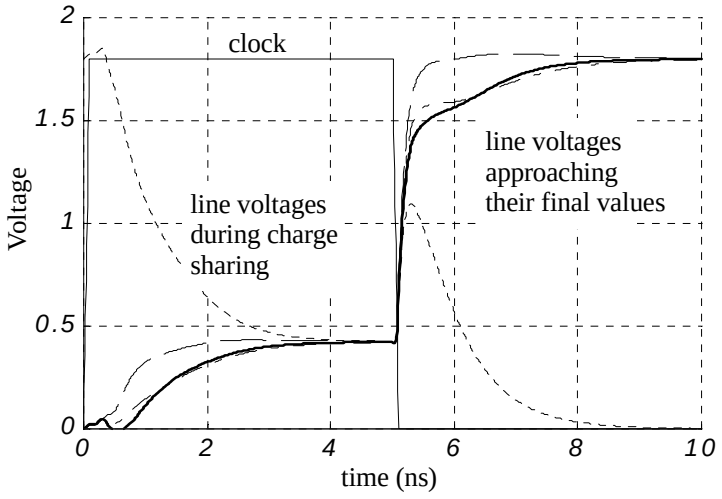

Figure 4.15: Line voltage waveforms during the two steps of CRT

using charge recycling are not significant. Moreover, in extremely short buses the CRT results in power consumption greater that the case where it is not used. Since, the additional circuitry that implements the CRT introduces some energy overhead, the design had to be very efficient.

This particular implementation was chosen among others because it requires the minimum amount of gates (1 XOR gate, 2 NAND gates and 2 Transmission Gates) for the implementation of the technique. There is also no need of additional registers for the detection of transitions, no extra control signals are needed since all the necessary control is generated in every line and there is also no need for an extra clock, because the switches are reset when the clock goes low.

The biggest advantage of this implementation, however, is that it can be implemented for an arbitrary number of bus lines. This is because the control circuit is independent on the other lines.

Chapter 5

Conclusions

5.1 Conclusions

In deep sub-micron technologies, the strong capacitive coupling between the lines of the bus, shifts large components of delay and power of integrated circuits, to the interconnect network of a circuit.

As technology scales down device sizes, buses can no longer be modeled as a set of independent lines, but rather as a distributed network where the output transient response is strongly affected by all the input transitions.

Interwire parasitic capacitances have become dominant, degrading the performance and increasing the dynamic power dissipation in buses. So the delay and power associated with buses have become a serious problem.

This thesis has examined practical circuit approaches, that address the issues of delay and power in deep sub-micron buses.

A practical coding implementation for speed increase was designed and the performance was analyzed. Coding eliminates input transitions that result in large delays by imposing redundancy on the bus. The circuit was designed in a 0.13μ CMOS process with dual threshold devices. The implementation has shown a significant increase of 36% on the frequency of operation compared to conventional buses.

Using charge recycling the dynamic power dissipated on large buses can be reduced substantially. When bus lines make one to zero transitions, the charge that is stored in those lines is used to partially charge lines making a zero to one transition, therefore achieving reuse. Effective charge recycling drivers were designed, and the performance

was tested on a 4 and 8 line bus. The net energy savings, in the second case were 32%, which proves the advantage of charge redistribution in buses with large capacitances.

Appendix A

Cypress 39K200 CPLD Power Breakdown

The next sections describe the methodology that was followed for the 39K200 Cypress CPLD Device Power Breakdown. A brief description of the 39K Cypress CPLD design family is given first, to make the understanding of the methodology easier. The Block diagram of the Cypress CPLD is shown in the Figure A.1

A.1 CPLDs and FPGAs

CPLD stands for Complex Programmable Logic Devices and are very similar in functionality with FPGAs (Field Programmable Gate Arrays). However, there are two basic

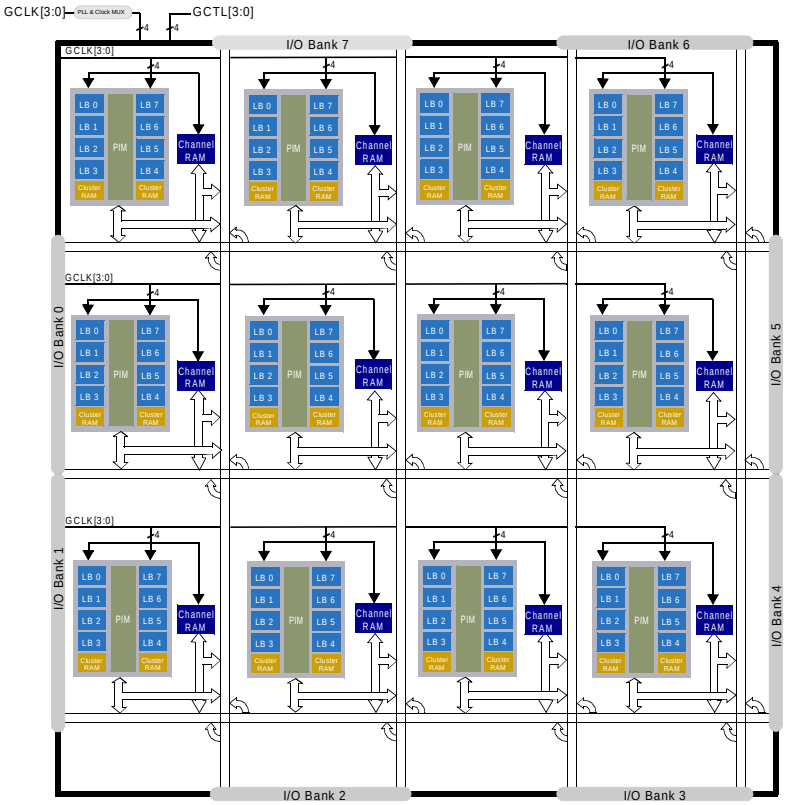

Figure A.1: Cypress 39K CPLD Device

differences between CPLDs and FPGAs. The first is the granularity of the blocks that are responsible for the logic operations. The different sub-parts of the CPLD are shown in Figure A.2 with more detail. Every cluster consists of 8 Logic Blocks (LB) that are connected with the Cluster PIM (Programmable Interconnect Matrix). In the clusters, there are also two Cluster Memory Blocks (CMB).

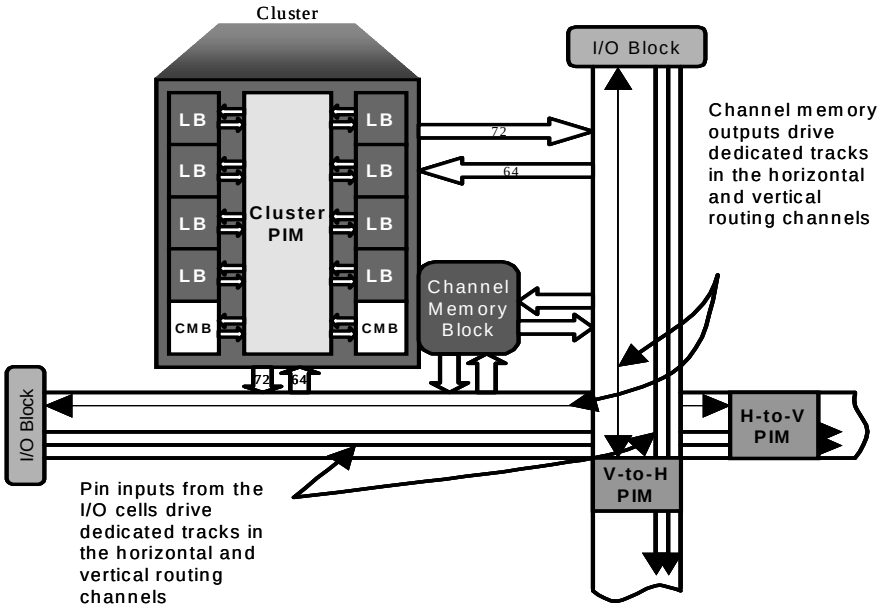

Figure A.2: Cluster and Channels

The second difference is related to the Channels (Interconnect Network). When data from the clusters gets into the channels, the data is available throughout the whole width and height of the channels, independently if it is needed or not. The H-to-V (Horizontal to Vertical) and V-to-H (Vertical to Horizontal) Blocks just short the vertical with horizontal channels.

A.2 Methodology

The whole chip can be broken into 5 big conceptual units:

a) Channels and Channel PIMs

This unit consists of the interconnect network (both data and configuration bit lines). The channel memory, for the purpose of the power breakdown, is included in the Memory Block unit and not here.

b) Cluster Block (LBs and PIMs)

It includes the Logic Blocks and the Cluster PIM. Again the Cluster Memory Block is included in the Memory Block.

c) Memory Blocks

This block groups the Cluster Memory and the Channel Memory Blocks, as seen in Figure A.2. They consist of static memory cells.

d) Clock and Control Tree

The Clock and Control Network are very similar in the design and they are grouped together. They include the necessary drivers for routing the clock and control signals. (Note that the control tree has nothing to do with the configuration of the device. The configuration bits are included in the Channel Block.)

e) I/O Block

The I/O Block includes the Input and Output Cells of the Chip. There are I/O blocks in the top and bottom of every Vertical Channel and the left and right side of every Horizontal Channel (see Figure A.1).

f) DC component

This unit includes the circuit that is responsible for the Control Tree, the PLL that is responsible for the clock and the Regulator Block. The Regulator block allows a 3.3V or 2.5V on the pins of the device, while running the core at 1.8V.

Each unit was broken into sub-blocks and the average current of each block was measured using PowerMill.

The CPLD can operate under two different modes. In the first mode (Mode 1) the chip operates as only a programmable device, where the embedded memory isn't utilized at all. In the second mode (Mode 2) there is an option in the user to utilize the static memory. The total embedded SRAM can be up to 500K-bits, in all the channel and cluster memory blocks.

Different parameters were provided by Cypress, namely the percentage of horizontal and vertical channel routes used, the percentage channel routes switching and the percentage of channel PIMs used. For the cluster blocks, information was provided about the number of cluster used in a design, the average number of logic blocks used in a design and the average number of cluster PIMs utilized.

Parameters about the memory blocks include the percentage of cluster and channel memory used (only in the second mode). For the I/O blocks the average number of enabled Inputs and Outputs were provided. An important note should be made on this point. The average current of a switching enabled output-cell driver was measured without including the output loading. This is the main reason that in the graphs provided below the power dissipated on the I/O blocks is not very significant.

A.3 Results

Based on the measured average currents and the statistical data that was provided there are the two graphs for Mode 1 and 2 and correspond to typical operation of the chip. The two graphs are given in Figure A.3 and Figure A.4 respectively.

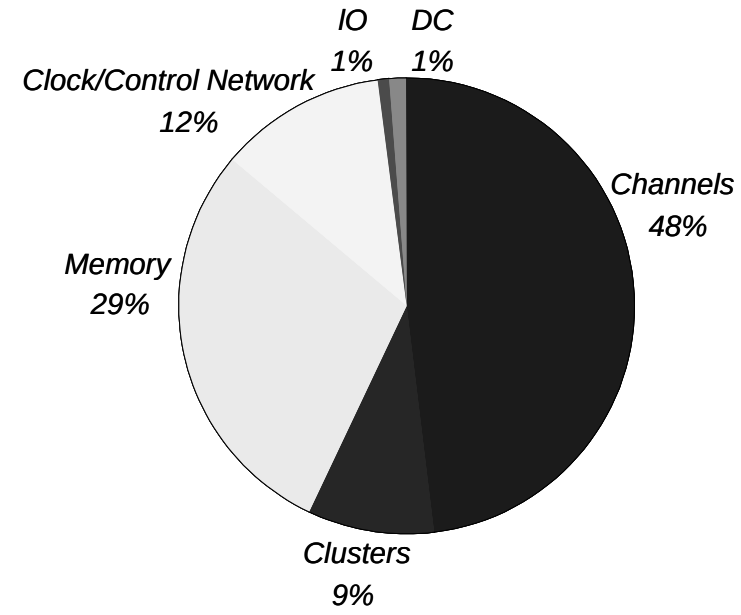

Figure A.3: Power Breakdown - Mode 1

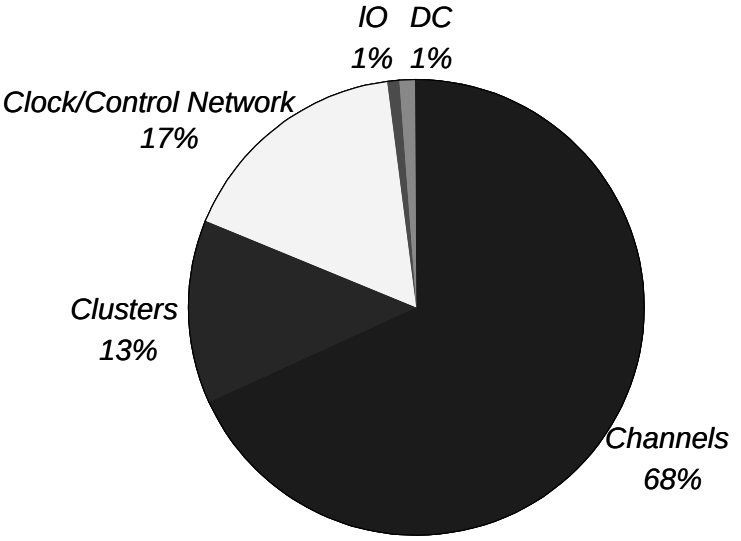

Figure A.4: Power Breakdown - Mode 2

References

[1] T. Sakurai, "Design Challenges for 0.1u and Beyong," *Design Automation Conference, 2000, Proceedings of the ASP-DAC 2000. Asia and South Pacific*.

[2] E. Kusse, J. Rabaey, “Low-Energy Embedded FPGA Structures”, *ISLPED ‘98.* August ‘98 Monterey USA.

[3] P. Sotiriadis, A. Chandrakasan, "Reducing bus delay in sub-micron technology using coding", *IEEE Asia and South Pacific Design Automation Conf.*, 2001, pp. 109-114.

[4] P. Sotiriadis, A. Chandrakasan, "Coding for Energy Reduction in Deep Sub-Micron Technologies, The Transition Pattern Coding Approach" , *accepted for publication in the IEEE Transactions on Circuits and Systems* , March 2001.

[5] P. Sotiriadis, A. Chandrakasan, "Reducing bus delay in sub-micron technology using coding", *IEEE Asia and South Pacific Design Automation Conf.*, 2001, pp. 109-114.

[6] P. Sotiriadis, "Interconnect Modeling and Optimization in Deep Sub-Micron Technologies", Ph.D. Thesis, MIT 2002

[7] H. Zhang, J. Rabaey, “Low swing interconnect interface circuits,” *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 161-166, August 1998.

[8] Y. Nakagome, K. Itoh, M. Isoda, K. Takeuchi, M. Aoki, “Sub-1-V swing internal bus architecture for future low-power ULSI’s,” *IEEE Journal of Solid-State Circuits,* pp. 414-419, April 1993.

[9] H. Yamauchi, H. Akamatsu, T. Fujita,”An asymptotically zero power charge-recycling bus architecture for battery-operated ultrahigh data rate ULSI's”, *Journal of Solid-State Circuits, IEEE, Vol. 30 Issue: 4*, pp. 423 -431, April 1995.

[10] K.Y. Khoo, A. Willson, Jr., “Charge recovery on a databus,” *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 185-189, 1995.

[11] B. Bishop, M. J. Irwin, “Databus charge recovery: Practical consideration,” *International Symposium on Low Power Electronics and Design*, pp. 85-87, August 1999.

[12] V. Lyuboslavsky, B. Bishop, V. Narayaran, M. J. Irwin, "Design of Databus Charge Recovery Mechanism", *ASIC/SOC Conference, 2000. Proceedings. 13th Annual IEEE International*, pp 283-287, September 2000.

[13] B. Bishop, V. Lyuboslavsky, N. Vijaykrishnan, M.J. Irwin, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 9, NO. 1* pp 104-106, February 2001.

[14] M. Stan, W. Burleson, “Low-power encodings for global communication in cmos VLSI,” *IEEE Transactions on VLSI Systems*, pp. 49-58, Vol. 5, No. 4, Dec. 1997.

[15] P. Sotiriadis, A. Wang, A. Chandrakasan, “Transition Pattern Coding: An approach to reduce Energy in Interconnect”, *ESSCIRC 2000.*

[16] S. Ramprasad, N. Shanbhag, I. Hajj, “A coding framework for low-power address and data busses,” *IEEE Transactions on VLSI Systems*, pp. 212-221, Vol. 7, No. 2, June 1999.

[17] J.M. Rabaey, *Digital Integrated circuits*. Prentice Hall 1996.

[18] H.B. Bakoglou, *Circuits, Interconnections, and Packaging in VLSI*. Addison-Wesley Pub. Co., 1990

[19] P. Sotiriadis, T. Konstantakopoulos, A. Chandrakasan, "Analysis and Implementation of Charge Recycling for Deep Sub-micron Buses", *IEEE/ACM Int. Sym. on Low Power Electronics and Design 2001, pp. 364-369.*