On-Chip Self-Test Circuit Blocks for High-Speed Applications

by

Ekaterina Laskin

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

On-Chip Self-Test Circuit Blocks for High-Speed Applications

Ekaterina Laskin

Master of Applied Science, 2006
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

In this thesis, the advantages of parallel pseudo-random bit sequence (PRBS) generators for high-speed self-test applications are examined. An ultra-low-power, 4-channel $2^7 - 1$ PRBS generator with 60 mW per channel was designed, fabricated and measured to work correctly up to 23 Gb/s. The circuit was based on a 12-Gb/s, 2.5-mW BiCMOS current-mode logic (CML) latch topology, which, to the best of my knowledge, represents the lowest power for a latch operating above 10-Gb/s. The fabricated chip also included an integrated PRBS checker and error counter. Techniques for further power reduction, by eliminating the current source transistor, and speed improvements, by adding inductive peaking, are presented.

Acknowledgements

This work would not be possible without the support and contributions of several people. I would like to express my thanks to Prof. Sorin P. Voinigescu for providing me with this opportunity and for giving ample guidance along the way. I also thank all the residents of BA4182 for their help and support during design, tapeouts, and courses.

I would like to express my deepest appreciation to my family for their love and care. Thanks to all my friends who tried their best to keep me from working too hard.

The following organizations are acknowledged for their support. STMicroelectronics for providing design kits and fabrication; CMC for CAD support; NSERC, Micronet, and Gennum for financial support; and CFI and OIT for equipment.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BER	Bit-Error Rate
BiCMOS	Bipolar and CMOS
BIST	Built-in Self Test
CDR	Clock and Data Recovery
CML	Current-Mode Logic
CMOS	Complementary Metal Oxide Semiconductor
DEMUX	De-multiplexer
DFF	Data Flip-flop
DUT	Device Under Test
ECL	Emitter-Coupled Logic
FET	Field-Effect Transistor
FSM	Finite State Machine
f_T	Unity current gain frequency
HBT	Hetero-junction Bipolar Transistor
LFSR	Linear Feedback Shift Register
MIM	Metal-Insulator-Metal
MOSFET	Metal Oxide Semiconductor FET
MUX	Multiplexer
NFET	N-channel Field-Effect Transistor
NRZ	Non-Return to Zero
PRBS	Pseudo-Random Bit Sequence
SCFL	Source-Coupled FET Logic
SERDES	Serializer-Deserializer
SiGe	Silicon-Germanium
TFF	Toggle Flip-flop
TIA	Transimpedance Amplifier

1 Introduction

1.1. Built-In Self-Test for High-Speed Transceivers

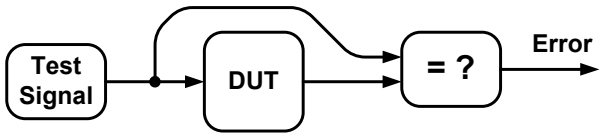
Communication networks continually evolve to cover larger areas, reach more places, and operate at increasing data rates. Currently, the 10-Gb/s Ethernet standard is being deployed for industrial and commercial use. Hence, research is focused on next generation 40-Gb/s Sonet and 100-Gb/s Ethernet systems. Recently, chipsets for 40-Gb/s transceivers [1] have been demonstrated. Furthermore, digital building blocks for 80-Gb/s applications and above [2, 3] are being developed in silicon, with increasing integration levels. As the data rates of state-of-the-art broadband circuits increase, these circuits outperform commercially available test equipment and verification of error-free operation becomes more challenging. Hence, custom circuits are needed to generate input data for testing purposes. This input data must resemble real input signals as closely as possible. Previously, pseudo-random bit sequence (PRBS) generators with sequence length of $2^{31} - 1$ have been developed [4, 5]. However, the testing circuits need to be compact and low power, suitable for use as circuit blocks that can be placed on the same chip as the system under test. For bit-error-rate testing, input signals with the suitable properties can be generated using linear-feedback shift registers (LFSR). The produced signals are called pseudo-random bit sequences due to their spectral content. Even though the LFSR structures are relatively simple, the main challenges lie in implementing them at high data rates with the restrictions of minimizing power and area.

A general setup for testing of broadband circuits is shown in Figure 1.1(a). Here, a test signal is fed into the device under test (DUT) and then compared with the output of the DUT. In this context, the term "DUT" refers to a broadband circuit. The general testing configuration shown in Figure 1.1(a) is not possible for most practical circuits because their input and output are far apart, such that the test signal cannot be connected directly as shown. On the other hand, if the input signal is known and if the system clock is recovered, it can be regenerated at the

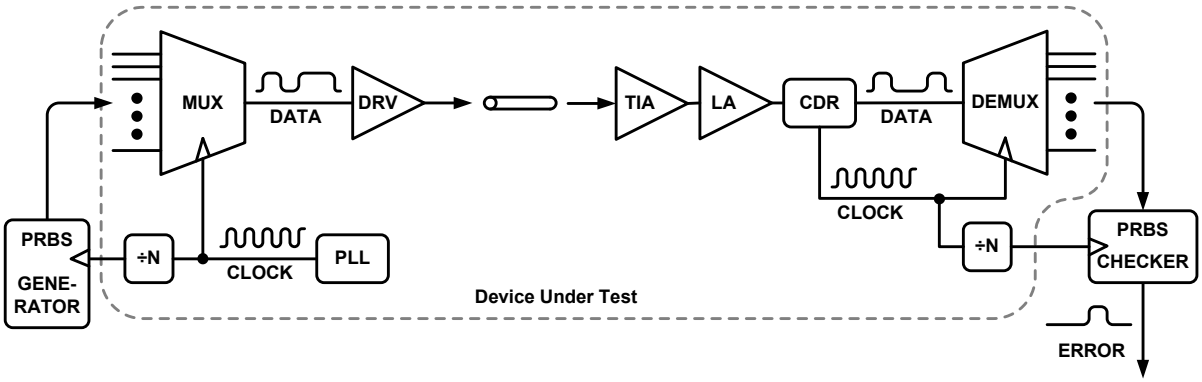
output side and then compared against the DUT output to check whether errors occur inside the DUT. This can be done with a PRBS generator, because the PRBS signal is deterministic and can be recreated exactly, once the clock signals are synchronized. The bit error rate test setup for a practical system is shown in Figure 1.1(b), where a PRBS generator produces the pseudo-random signal which is sent into the device being tested. On the output side of this device, a PRBS checker recreates the same pseudo-random signal as the generator and compares it with the output of the DUT to find errors. For this to happen the clock signals and data streams in the PRBS generator and error checker must be synchronized. The device shown in Figure 1.1(b) is a complete SERDES, however smaller sub-circuits, such as a TIA, driver, re-timer, CDR, MUX-DEMUX, transmitter, or receiver, can also be tested using the same method.

1.2. Speed and Power Optimization of High-Speed Logic

Although the main application of PRBS generators is testing, they contain digital blocks that can be reused in other applications. Thus, to achieve high system integration in the future, it is essential to find how the design of each constituent block can be optimized for speed and power.



(a) Conceptual setup for BER test of a device



(b) Use of a PRBS for system BER testing

Figure 1.1: Use of PRBS generator and checker for testing

At data rates above 10 Gb/s, current-mode logic or emitter-coupled logic are used to implement digital gates. In these types of circuits, power consumption is given by the supply voltage multiplied by the constant current that biases the gate. Previously, bipolar-only or MOS-only implementations of the gates have been made. However, bipolar-only implementations require a high supply voltage. MOS-only implementations operate from a lower supply but require more current to reach the same speed of operation. Hence, a BiCMOS gate implementation, which consists of both MOS and HBT devices [6], is explored in this thesis as an alternative configuration that demands less power to operate at a given speed.

Circuit design is a very challenging task, especially at very high data-rates that approach the maximum switching frequency of the transistors. To ease this problem, a procedure will be presented that explains how to choose transistor sizes and bias points to achieve high-speed gate design with low power consumption.

1.3. Objective of Thesis

The objective of this thesis is two-fold. The first target was to implement a $2^7 - 1$, 23-Gb/s PRBS generator and error checker blocks, such that they could later be used for testing of other circuits. The second goal was to find how the design of high-speed digital blocks can be optimized for minimal power while still operating at a given data rate (above 10 Gb/s).

Both of these objectives are combined in this thesis because the PRBS generator was originally designed to be the first stage of an 80-Gb/s transceiver with 10-Gb/s inputs. Thus, the lowest data rate in the generator is above 10 Gb/s, and the output of the generator works up to 23 Gb/s to have margin over 80 Gb/s after 4-to-1 multiplexing.

1.4. Contributions from this Work

The work to be presented in this thesis contains several contributions to the field of high-speed digital circuit design and testing.

The first contribution is the design of a low-power $2^7 - 1$ PRBS generator with 4, appropriately delayed, parallel output streams at 23-Gb/s each. This PRBS generator was fabricated and verified to operate correctly. Thus it can be integrated as a self-test reusable circuit block for other systems. The four outputs of the generator can be further multiplexed to an aggregate PRBS output at 92 Gb/s with minimal circuitry. The 4-channel PRBS generator consumes

235 mW from 2.5 V, which results in only 60 mW per output lane. The entire fabricated chip, which also includes an error checker and a 5-bit error counter, consumes 940 mW.

The second contribution is the design of a low-power SiGe BiCMOS CML latch, which worked up to 12 Gb/s, while consuming only 2.5 mW. This latch has enabled the low power performance of the PRBS generator circuit. It opens the possibility to develop highly integrated broadband systems that operate above 10 Gb/s, with low power consumption. An investigation into further power reduction and speed improvement up to 40 Gb/s of high-speed digital gates in SiGe BiCMOS and 65-nm CMOS technologies is also presented in this thesis.

Additionally, a thorough analysis of series and parallel PRBS generator architectures is presented, concerning their power requirements, their applicability to high-speed implementation, and the respective design challenges. The comparison will be done in terms of the circuit complexity (i. e. number of flip-flops, gates, and their fanout) required by each architecture to implement the needed function.

The PRBS error checker implementation that is described in this thesis is novel. As opposed to existing PRBS error checkers that produce three or fewer pulses for each error present in the signal, the proposed checker circuit generates only one pulse for each error in the input signal. Thus, it is more useful for precise bit error rate measurements.

2 PRBS Generator and Checker Architectures

This chapter presents the theory of generating pseudo-random bit sequences. It starts by presenting the mathematical properties of pseudo-random bit sequences in section 2.1. Then, the two existing series and parallel PRBS generator configurations are described in section 2.2 and section 2.3 respectively. Various PRBS checkers are shown in section 2.4. Finally, section 2.5 presents a novel comparison of the applicability of series and parallel pseudo-random bit sequence generator and checker topologies to high-speed applications.

2.1. Definition and Properties of Pseudo-Random Bit Sequences

Pseudo-Random Bit Sequence generators are finite state machines (FSM) consisting of a linear feedback shift register (LFSR) circuit. In general, a shift register has one input, and n memory elements, called the stages of the register. Each memory element of the register stores one bit of data, resulting in n outputs from the register. The contents of the shift register at any given time are called the state of the register. Feedback is added to a general shift register by means of a circuit that realizes a function of n inputs and 1 output. The n inputs of the function are taken from the n elements of the shift register and its one output is connected to the input of the shift register, as shown in Figure 2.1. A Feedback Shift Register is called *linear* when the feedback function can be expressed in the form [7]

$$g(x_1, x_2, ...x_n) = c_1x_1 \oplus c_2x_2 \oplus c_3x_3 \oplus ... \oplus c_nx_n \tag{2.1}$$

where each of the constants c_i is either 0 or 1, and $\oplus$ denotes addition modulo 2. When some of the constants c_i are 0, then the number of inputs of g is less than n. Addition of linear feedback to a shift register makes the output become periodic (theorem 1 in [7]). Each of the n stages of the LFSR can be either 1 or 0, resulting in a maximum of 2^n different states in which the LFSR

can be found. Depending on the feedback function g and on the initial state of the LFSR, the number of different states can be reduced down to 1.

Since any LFSR repeats its states periodically, the output of each stage is also periodic. The period of the output is the length of the sequence that the LFSR generates. Although the maximum number of states the LFSR can be in is 2^n, the maximum length of the sequence that can be generated by the same LFSR is $2^n - 1$. This is because a state with all 0's stored in the LFSR can never occur when the output is periodic (with a period greater than 1). Conversely the all 0 state can be regarded as being periodic with the period equal to 1. Depending on the feedback function g, it is possible to construct the same n-stage shift register to generate either sequences with maximum $2^n - 1$ length (also called "maximal-length", or m-sequences) or sequences with shortened lengths.

It so happens that m-sequences also have randomness properties that make them appear as noise. However, these sequences are periodic and therefore not truly random. They are also called pseudo-random bit sequences (PRBS). The randomness properties of PRBS are the following [7]:

1. A balance of 1 and 0 terms.

2. Two runs of length n for each run of length $n + 1$.

3. A two-level auto-correlation function.

A balance of 1 and 0 terms implies that the number of 1's and 0's in a pseudo-random sequence is different by at most 1. As a result, for longer sequences, the probability of 1's and 0's gets closer to $1/2$.

A run of length n is part of a sequence where n identical bits occur consecutively. Because in a PRBS there are two runs of length n for each run of length $n + 1$, $1/2$ of the number of

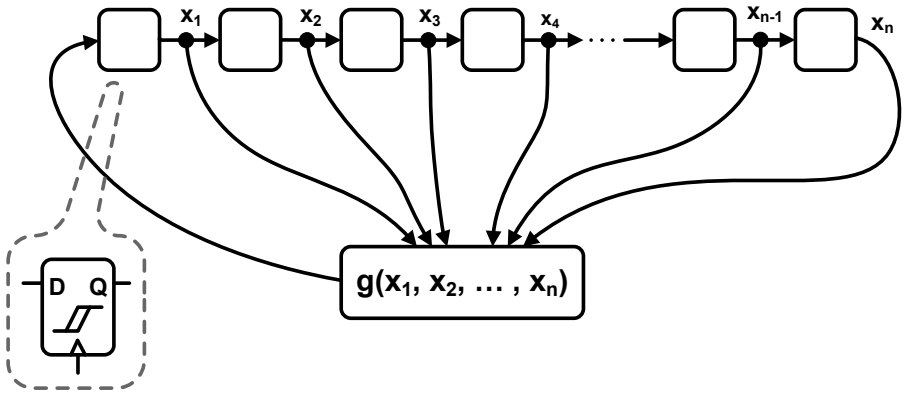


Figure 2.1: Diagram of a feedback shift register

runs is of length 1, $1/4$ of the runs is of length 2, $1/8$ of length 3, etc... Analogously, in a random coin toss experiment, 2 consecutive heads occur with probability $1/4$, 3 consecutive heads occur with probability $1/8$, etc...

The auto-correlation function of any binary periodic sequence $\{a_n\}$ with period p, which is composed of $+1$'s and -1's is defined as [7]

$$C(\tau) = \frac{1}{p}\sum_{n=1}^{p} a_n a_{n+\tau} \tag{2.2}$$

where τ is the phase shift in bits. $C(\tau)$ is always highest for $\tau = 0$ and drops off for larger τ. For a PRBS, $C(\tau) = 1$ for $\tau = 0$ and $C(\tau) < 1$ for $0 < \tau < p$. A two-level auto-correlation function confirms that the sequence is as close as possible to being random. This is because the sequence is uncorrelated with phase shifts of itself, unless the phase shift is equal to an integer number of sequence periods.

Thanks to these randomness properties, PRB-sequences are useful as data sources when testing other circuits or systems. On one hand they are close to being random and therefore exercise the device under test (DUT) for many different input combinations. On the other hand, they are completely defined and repeatable and therefore more useful than just digitized noise.

PRB-sequences can be generated from a LFSR with any number of stages by selecting an appropriate feedback function [8]. However certain sequence lengths are more widely used than others because they are standardized [9]. The common PRB-sequence lengths are $2^7 - 1$, $2^{15} - 1$, $2^{23} - 1$, and $2^{31} - 1$ requiring at least 7, 15, 23, and 31 memory elements in the LFSR, respectively. Longer sequence lengths are desirable because they repeat less often and have longer run lengths, which subjects the DUT to more exhaustive tests. However, the main challenge of building PRBS generators that produce longer sequence lengths is that they consume more power and occupy more die area.

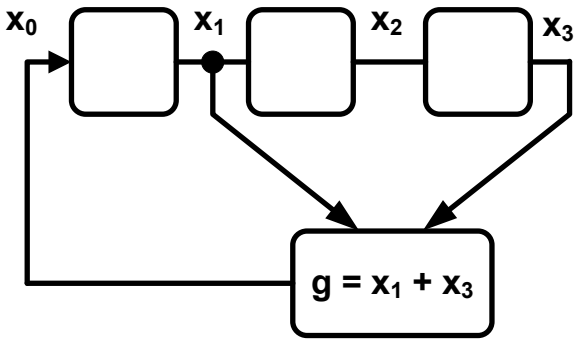
For a LFSR with n stages and a feedback function $g(x_1, x_2, ...x_n) = c_1x_1 \oplus c_2x_2 \oplus c_3x_3 \oplus ... \oplus c_nx_n$, the output of each stage is a PRBS $\{a_k\}$ with period $p = 2^n - 1$. As described in [7], $\{a_k\}$ satisfies the recurrence relation

$$a_k = \sum_{i=1}^{n} c_i a_{k-i} \quad \text{(modulo 2)}. \tag{2.3}$$

This can be seen by looking at the example of a PRBS with period $p = 7$, shown in Figure 2.2. The produced sequence is shown in columns.

The polynomial

$$f(x) = 1 + \sum_{i=1}^{n} c_i x^i \quad \text{(modulo 2)} \tag{2.4}$$



x_1	x_2	x_3
1	1	1
0	1	1
1	0	1
0	1	0
0	0	1
1	0	0
1	1	0
1	1	1

Figure 2.2: A 3-stage LFSR and one of its maximal-length sequences

is called the characteristic polynomial of the sequence $\{a_k\}$. The necessary condition for $\{a_k\}$ to be a PRBS is that $f(x)$ is irreducible, that is it cannot be factored. The necessary and sufficient condition is that $1 + x^m$ can be divided (modulo 2) exactly by $f(x)$ for $m = p$, but for no positive integer m smaller than p [7]. The number of distinct pseudo-random sequences of length $p = 2^n - 1$ is $\phi(p)/n$, where $\phi(p)$ is Euler's function[1]. For example there are 18 distinct characteristic polynomials that can generate a PRBS of length $2^7 - 1$ (Table 2.1). Out of these 18, the polynomials in rows 1-4 of Table 2.1 require the feedback function to add the outputs of only 2 stages, thus decreasing the amount of circuitry need to generate the PRBS. However, the PRBS shown in row 2 is used most often because it is one of the standard testing sequences [9].

In addition to the above definitions and randomness properties, m-sequences posses two other useful properties. These properties are very practical in applications for building PRBS generators, as will be shown later. First, PRB-sequences possess the "cycle-and-add" property [7]. This means that when two identical sequences, which are phase shifted with respect to each other, are added bitwise (modulo 2), then the result is an identical sequence but with some phase shift. This is true because PRB-sequences always satisfy a *linear* recurrence relation. Therefore, given two PRB-sequences A_i and A_j that satisfy a linear recurrence relation R, then $A_i + A_j$ also satisfies R and therefore is the same PRB-sequence. Here the addition is bit-wise and modulo 2.

The second property relates to decimation of PRB-sequences [7]. Decimation is defined as forming a sequence $\{a_{qk}\}$ from the sequence $\{a_k\}$ by taking every q^{th} bit of $\{a_k\}$ (q is a

[1]The totient function $\phi(p)$, also called Euler's totient function, is defined as the number of positive integers $\leq p$ that are relatively prime to (i.e., do not contain any factor in common with) p, where 1 is counted as being relatively prime to all numbers. Since a number less than or equal to and relatively prime to a given number is called a totative, the totient function $\phi(p)$ can be simply defined as the number of totatives of p. For example, there are eight totatives of 24 (1, 5, 7, 11, 13, 17, 19, and 23), so $\phi(24) = 8$ [10].

Row	Characteristic Polynomial	Pseudo-Random Bit Sequence
1	$x^7 + x + 1$	
2	$x^7 + x^6 + 1$	
3	$x^7 + x^3 + 1$	
4	$x^7 + x^4 + 1$	
5	$x^7 + x^3 + x^2 + x + 1$	
6	$x^7 + x^6 + x^5 + x^4 + 1$	
7	$x^7 + x^4 + x^3 + x^2 + 1$	
8	$x^7 + x^5 + x^4 + x^3 + 1$	
9	$x^7 + x^5 + x^2 + x + 1$	
10	$x^7 + x^6 + x^5 + x^2 + 1$	
11	$x^7 + x^5 + x^3 + x + 1$	
12	$x^7 + x^6 + x^4 + x^2 + 1$	
13	$x^7 + x^5 + x^4 + x^3 + x^2 + x$	
14	$x^7 + x^6 + x^5 + x^4 + x^3 + x$	
15	$x^7 + x^6 + x^3 + x + 1$	
16	$x^7 + x^6 + x^4 + x + 1$	
17	$x^7 + x^6 + x^5 + x^3 + x^2 + x$	
18	$x^7 + x^6 + x^5 + x^4 + x^2 + x$	

Table 2.1: All 18 possible characteristic polynomials for a $2^7 - 1$ PRBS

positive integer). Decimation of any PRB-sequence forms another PRB-sequence. However, the important property is that when q is a power of 2, decimation by q does not change the order of bits in the PRBS. That is, if $\{a_k\}$ is a PRB-sequence then $\{a_{qk}\}$ differs from $\{a_k\}$ by at most a phase shift, for $q = 1, 2, 4, 8, \ldots, 2^{n-1}$. This can be proved as follows. Let D be a unit delay operator, so that $Da_k = a_{k-1}$ and $D^2 a_k = a_{k-2}$. Then, from equations 2.3 and 2.4,

$$f(D)\{a_k\} = (1+\sum c_i D^i)\{a_k\} = \{a_k\}+\sum c_i D^i\{a_k\} = \{a_k\}+\sum c_i\{a_{k-i}\} = \{a_k\}+\{a_k\} = 0 \tag{2.5}$$

So $f(D)\{a_k\} = 0$ is equivalent to the recurrence relation of equation 2.3. Also, from the simplified binomial theorem[2], it follows that

$$f(x^{2^i}) = [f(x)]^{2^i}. \tag{2.6}$$

[2] $(a \oplus b)^{2^i} = a^{2^i} \oplus b^{2^i}$

Now, consider the sequence $\{a_{2k}\}$, formed by taking every second bit of $\{a_k\}$, or equivalently, by the operation

$$f(D^2)\{a_k\} = f(D)\{a_{2k}\}. \tag{2.7}$$

By using equations 2.6 and 2.7 together with $f(D)\{a_k\} = 0$, we obtain

$$f(D)\{a_{2k}\} = f(D^2)\{a_k\} = [f(D)\{a_k\}]^2 = f(D)[f(D)\{a_k\}] = f(D)\{0\} = 0. \tag{2.8}$$

Thus, $\{a_{2k}\}$ satisfies $f(D)\{a_{2k}\} = 0$, which is the same recurrence relation as equation 2.3 for $\{a_k\}$. Therefore, the sequence $\{a_{2k}\}$ is identical to $\{a_k\}$, except for a possible phase shift. The same proof can be extended to show that $\{a_{qk}\}$ for $q = 1, 2, 4, 8, \ldots, 2^{n-1}$ is also the same sequence as $\{a_k\}$.

The decimation property of PRBS proves to be most useful when applied in reverse. That is, two appropriately shifted sequences $\{a_{2k}\}$ and $\{a_{2k-j}\}$ can be multiplexed into $\{b_k\}$ by alternating the bits from $\{a_{2k}\}$ and $\{a_{2k-j}\}$. In order for $\{b_k\}$ to also be pseudo-random, the phase shift must be $j = (p-1)/2$ [11], where the length of the sequence is $p = 2^n - 1$ bits. This operation effectively doubles the bit rate of the generated PRBS.

To the best of my knowledge there exist two different circuit architectures to generate PRB-sequences. The first one is series architecture, which directly applies the LFSR theory. The second one still generates the same PRBS, but is optimized to generate several shifted sequences in parallel [12].

2.2. Series PRBS Generators

Series PRBS generators are, in effect, just shift registers with a linear feedback function that causes the shift register to output a maximal length sequence. As mentioned before, for each shift register, several different characteristic polynomials can generate a PRBS, however, typ-

Shift Register Length n	Characteristic Polynomial	PRBS Length
7	$x^7 + x^6 + 1$	$2^7 - 1 = 127$
9	$x^9 + x^5 + 1$	$2^9 - 1 = 511$
11	$x^{11} + x^9 + 1$	$2^{11} - 1 = 2,047$
15	$x^{15} + x^{14} + 1$	$2^{15} - 1 = 32,767$
23	$x^{23} + x^{18} + 1$	$2^{23} - 1 = 8,388,607$
29	$x^{29} + x^{27} + 1$	$2^{29} - 1 = 536,870,911$
31	$x^{31} + x^{28} + 1$	$2^{31} - 1 = 2,147,483,647$

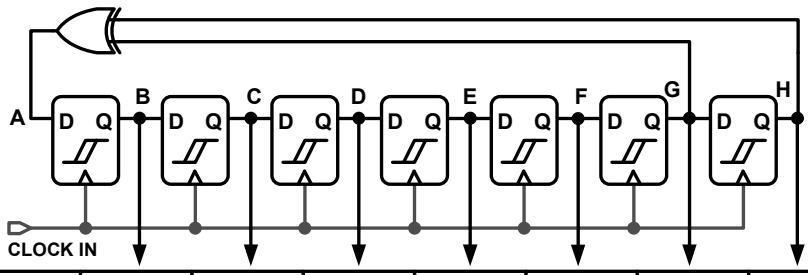
Table 2.2: Standard characteristic polynomials for various PRBS lengths

ically only certain characteristic polynomials are used. To minimize the number of gates required for addition, these polynomials are of the form $1 + x^i + x^n$, where n is the number of stages in the shift register. The PRBS characteristic polynomials that are used in measuring and testing equipment are summarized in Table 2.2 [9]. Figure 2.3 shows a schematic of a series PRBS generator, using D-flip flops as memory elements and an XOR gate as the adder, that generates a sequence 127 bits long.

From Figure 2.3 it can be observed that the PRBS bits are output sequentially (hence the name "series generator") from the generator. That is, during the first clock cycle, the stage outputs produce bits 1, 2, 3, 4, 5, 6, 7, during the second clock cycle the outputs have bits 2, 3, 4, 5, 6, 7, 8, during the third clock cycle bits 3, 4, 5, 6, 7, 8, 9 of the sequence are produced, and so on. Note that because the sequence is periodic, the starting bit can be chosen arbitrarily.

Because in series PRBS generators the sequence bits are generated sequentially, they are not well suited for multiplexing to higher bit rates. In this case, before multiplexing can be applied, additional circuitry is needed. To multiplex two PRB-sequences into a PRBS at double the original bit rate, the phase shift between the two incoming sequences must be $(p-1)/2$ [11], where $p = 2^n - 1$ is the length of the sequence in bits. For example for a $2^7 - 1$ PRBS, bits 1 and 64 of the sequence must be available during one clock cycle, bits 2 and 65 during the next clock cycle, etc...but these bits are not available at the same time in a series PRBS generator. Fortunately, this problem can be solved by using more circuitry and applying the "cycle-and-add" property of PRBS.

The "cycle-and-add" property states that by adding two sequences, a third sequence with a different phase is produced. But here the problem is reversed. We know the required phase of the sequence and need to find the phases of the other two (or more) sequences that, when added, give the required phase. Furthermore, the phases of the added sequences must be such



CLOCK 1	7	6	5	4	3	2	1
CLOCK 2	8	7	6	5	4	3	2
CLOCK 3	9	8	7	6	5	4	3
CLOCK 4	10	9	8	7	6	5	4
CLOCK 5	11	10	9	8	7	6	5
...	...	...	...	...	...	...	...

Figure 2.3: A series generator for a $2^7 - 1$ PRBS

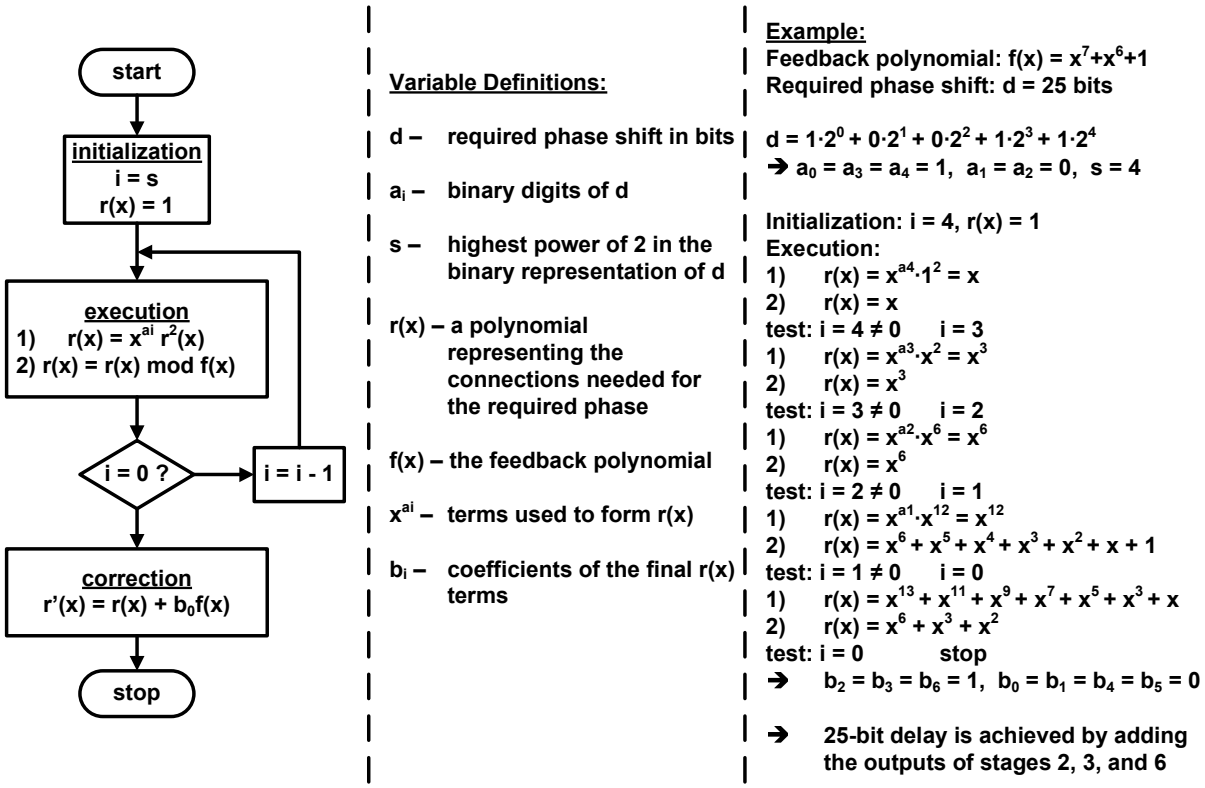


Figure 2.4: Algorithm for finding phases of added sequences to produce a sum sequence with the required phase

that they are directly available from the shift register stages. To do that, an efficient, $O(\log(n))$, algorithm exists [13]. The algorithm is shown in Figure 2.4. Sometimes, many adders are needed to produce the required phase for multiplexing, but since the beginning of the sequence is arbitrary, there is some freedom in the choice of the reference sequence for multiplexing. For example, to multiplex a $2^7 - 1$ PRBS, any of the pairs with phases 1 and 64, or 2 and 65, or 3 and 66, ..., or 7 and 70 can be used. Therefore, the algorithm of Figure 2.4 can be run iteratively to find the optimal (in terms of the required number of additions) pair of sequences to multiplex.

Multiplexing can be repeated more than once to increase the bit rate of the final sequence by $2, 4, 8, \ldots, 2^{n-1}$ times. To multiplex q times, q PRB-sequences are required at the original bit rate. Here q is a power of 2 because the basic multiplexing circuit combines 2 inputs into 1 at twice the bit rate. To ensure that the final output is a PRBS, the q original sequences must be equally spaced apart by $(p-1)/q$ bits in phase, where $p = 2^n - 1$ is the length of the sequence in bits. However, generating all the appropriately phased sequences may require a large amount of additions (XOR gates), and offset the benefit of generating the PRBS at a lower bit rate.

2.3. Parallel PRBS Generators

It is possible to produce a PRBS using a structure different than a LFSR, as long as it implements the correct characteristic polynomial. Parallel PRBS generators are an extension of series generators and can produce several shifted sequences in parallel. The phase shift between the parallel output sequences is such that they can be directly multiplexed to higher bit rates. Any series PRBS generator with n memory elements can be restructured into a series-parallel PRBS generator with n memory elements and k parallel outputs at the expense of $k-1$ additional XOR gates. When $k = n$, the generator is completely parallel [14].

For analyzing and constructing parallel PRBS generators, it is convenient to represent the circuit using a transition matrix $\mathbf{T}$ that represents how data is transferred between the memory elements of the generator. If $U(j)$ is an $n \times 1$ vector of 1's and 0's that represents the state (what is stored in each memory element) of an n-stage PRBS generator at the j^{th} clock cycle, then $\mathbf{T}$ is an $n \times n$ matrix that can be used to find the state of the generator at the next clock cycle.

$$\mathbf{U}(j+1) = \mathbf{T} \cdot \mathbf{U}(j). \tag{2.9}$$

Consequently, the state of the generator after k clock cycles will be

$$\mathbf{U}(j+k) = \mathbf{T}^k \cdot \mathbf{U}(j). \tag{2.10}$$

The transition matrix $\mathbf{T}$ is closely related to the characteristic polynomial of the m-sequence. The columns of $\mathbf{T}$ correspond to the data stored in the stages of the PRBS generator, assuming the stages are numbered in the same direction as the data is shifting. The rows of $\mathbf{T}$ correspond to the connections that exist between stages. That is, each 1 in a particular row corresponds to a connection made from a stage represented by the column of the 1 to a stage represented by the row of the 1. In rows where more than one 1 appear, the inputs are summed (modulo 2) before the connection is made. All other entries of the matrix are 0 [14]. For a series PRBS generator (LFSR), $\mathbf{T}$ contains 1's in the diagonal below the main diagonal and 1's in the first row and columns numbered the same as the exponents of the characteristic polynomial. An example of a transition matrix $\mathbf{T}$ that describes a $2^5 - 1$ series PRBS generator with the characteristic polynomial $f(x) = x^5 \oplus x^3 \oplus 1$ is shown in Figure 2.5, which also includes several periods of the produced sequence.

When the number of outputs of an n-stage PRBS generator is increased to $1 < k < n$, then $\mathbf{T^k}$ describes the connections between the PRBS generator blocks; it is constructed as follows. A diagonal array of $(n-k)$ 1's appears k rows below the main diagonal. The k^{th} row of $\mathbf{T^k}$ is the same as the first row of $\mathbf{T}$. Row i, for $i < k$, is the same as row 1 of $\mathbf{T}$

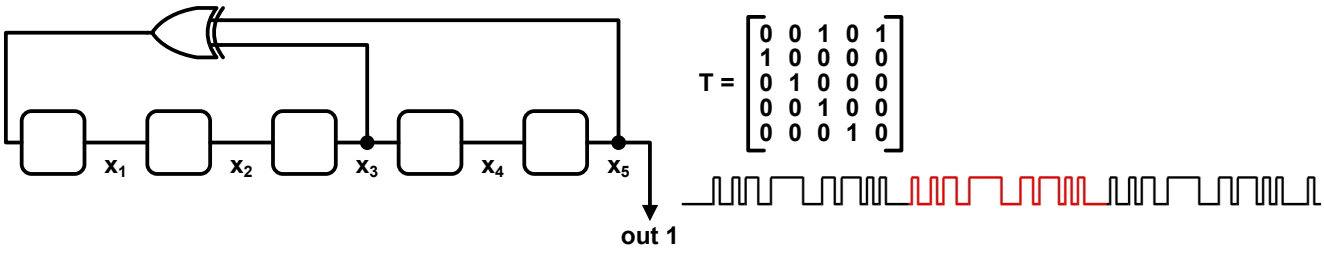
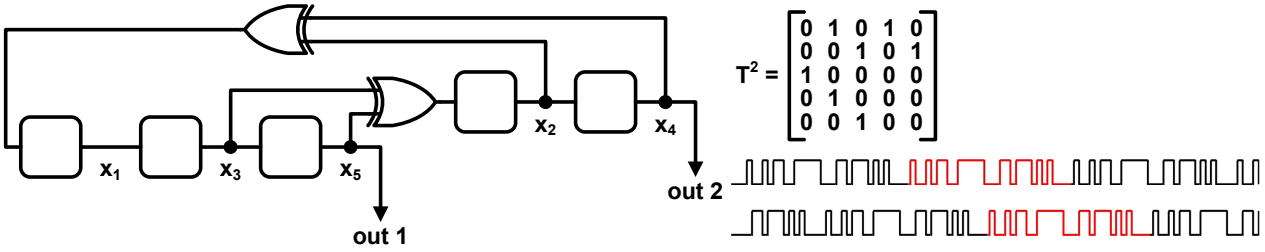


Figure 2.5: A $2^5 - 1$ series PRBS generator with its transition matrix

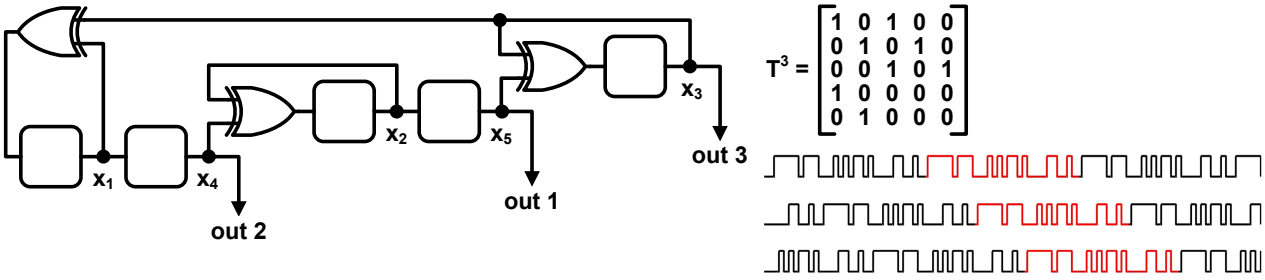
shifted left by $k - i$ column positions, with zeros shifted in from the right-hand side; if 1's are shifted out to the right, then a copy of row k is added (modulo 2) to row i. To illustrate this process, 2^5-1 series-parallel PRBS generators implementing the characteristic polynomial $f(x) = x^5 \oplus x^3 \oplus 1$ are shown in Figure 2.6 for all 5 possible values of k with the corresponding $\mathbf{T^k}$ [14, 15]. Beside each generator, the k produced sequences are shown for several periods. One of the periods is highlighted to emphasize the phase shift between them. Note that only $\mathbf{T^2}$ and $\mathbf{T^4}$ of Figure 2.6 produce the exact same sequences as that of $\mathbf{T}$ in Figure 2.5, while $\mathbf{T^3}$ and $\mathbf{T^5}$ produce different sequences that are also PRBS. This effect is a consequence of the decimation property of pseudo-random bit sequences.

Parallel PRBS generators operate by decimation. That is, when k parallel outputs are generated from a PRBS, a transition matrix $\mathbf{T^k}$ is used, meaning that in one clock cycle, according to equation 2.10, the generator goes through k states. As mentioned before, decimating a PRB-sequence changes its phase. This effect is also shown in Figure 2.6. Conveniently, since the the k parallel sequences are decimated by k and phase shifted accordingly, they can be easily serialized into one PRBS stream at k times the original bit rate. This can be done by direct multiplexing and without requiring additional phase shifting circuitry as in the case of a series PRBS generator. It should be noted that the k parallel PRBS streams require $k - 1$ additional adders (XOR gates) compared to a series PRBS generator.

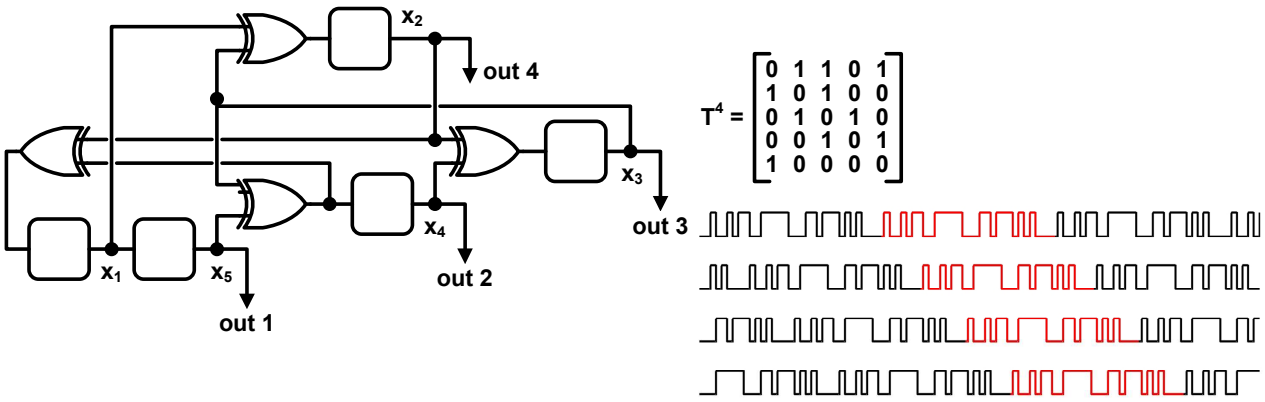
The small parallel generators shown in Figure 2.6 are not practical. Usually, in practice, sequence lengths of $2^7 - 1$, $2^{15} - 1$, $2^{23} - 1$, $2^{31} - 1$ are required and $k = 2, 4, 8 \ldots$ is used because it is easy to multiplex. Two examples of practical parallel PRBS generators are shown in Figure 2.7. The first one (Figure 2.7(a)) is a $2^7 - 1$ PRBS generator with 8 parallel outputs. The table shows the phase-shift in bits of each output. The second schematic (Figure 2.7(b)) shows a $2^{31} - 1$ PRBS generator with 8 parallel outputs. Due to space limitation, the memory elements (flip-flops) of the chain are shown as small numbered blocks. Each number represents the delay from the beginning of the chain, analogous to Figure 2.6.



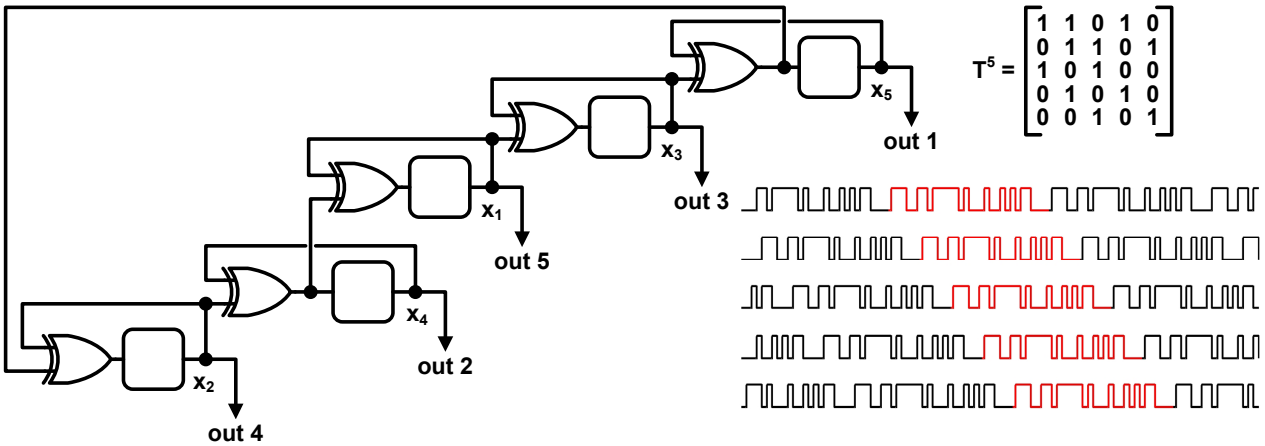
(a) $2^5 - 1$ PRBS generator with 2 parallel outputs



(b) $2^5 - 1$ PRBS generator with 3 parallel outputs

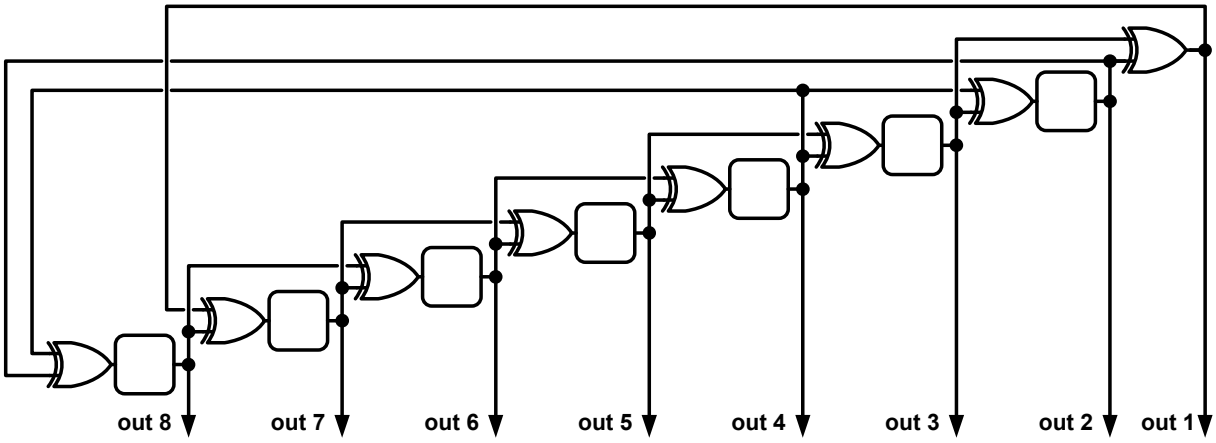


(c) $2^5 - 1$ PRBS generator with 4 parallel outputs



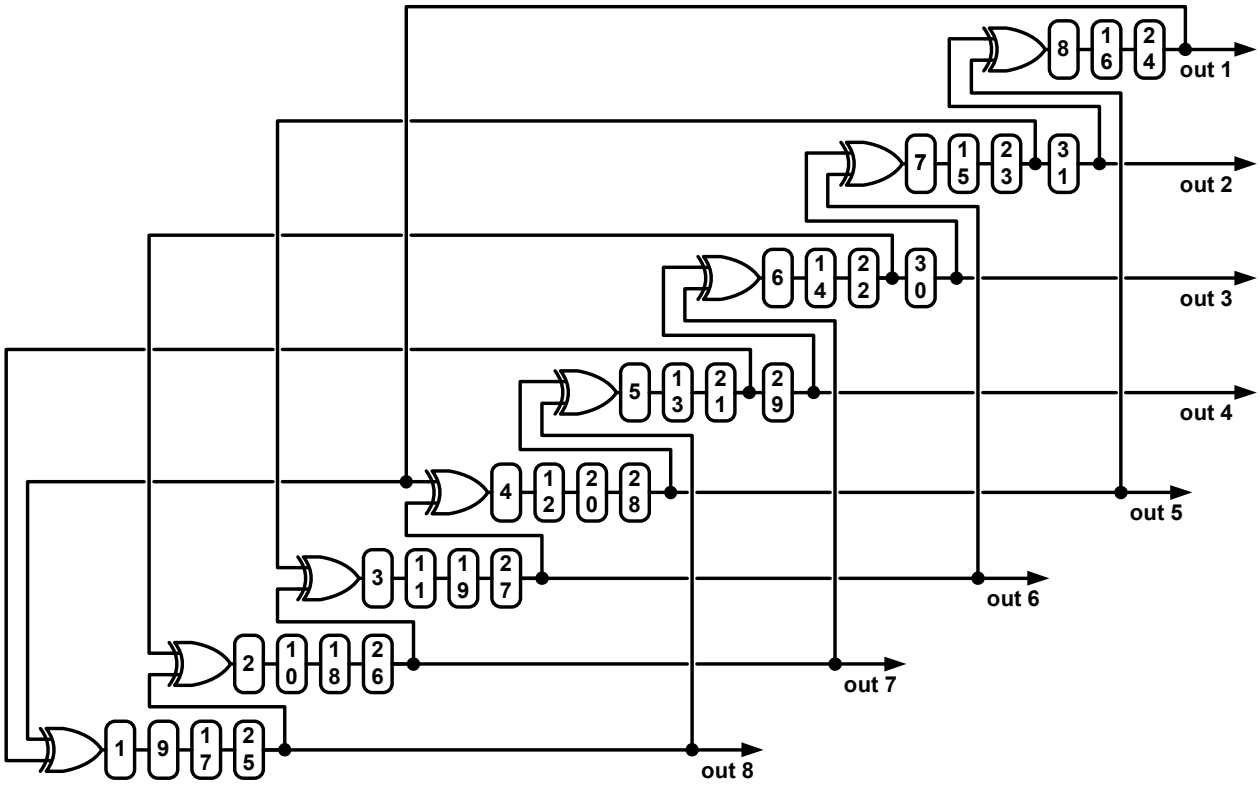
(d) $2^5 - 1$ PRBS generator with 5 parallel outputs

Figure 2.6: Series-parallel $2^5 - 1$ PRBS generators with their transition matrices and output sequences



CLOCK 1	112	96	80	64	48	32	16	0
CLOCK 2	113	97	81	65	49	33	17	1
CLOCK 3	114	98	82	67	50	34	18	2
CLOCK 4	115	99	83	68	51	35	19	3
CLOCK 5	116	100	84	69	52	36	20	4
...	...	...	...	...	...	...	...	...

(a) $2^7 - 1$ PRBS generator with 8 parallel outputs



(b) $2^{31} - 1$ PRBS generator with 8 parallel outputs

Figure 2.7: Examples of practical parallel PRBS generators

2.4. PRBS Checker Design

PRBS generators are used as data sources on the transmit side, or on the input of a device under test (DUT). PRBS checkers are needed on the receive side, or at the output of a DUT, to verify correct transmission of data or to find the bit-error-rate (BER) of the DUT. To check a pseudo-random sequence for correctness, it has to be compared to a reference sequence on the receiver. For that, the incoming sequence and the local reference have to be synchronized. Furthermore, correct synchronization must be maintained over long periods of time.

Synchronization can be achieved quickly by using the idea presented in [16]. This idea makes use of the series PRBS generator structure, but with the feedback loop broken. Figure 2.8 shows how this can be done for a $2^5 - 1$ PRBS, discussed earlier. Here, the feedback loop is broken between nodes A and G, which used to be one node in the generator case. Node A becomes an input, and node G becomes an output. If a pseudo-random bit sequence is connected to the created input at node A, the created output G will give an identical sequence because the input and output used to be one node. Now it becomes easy to check the input signal for errors. The input signal is simply compared to the output using an XOR gate. When the input signal is a PRBS, the comparator will produce a zero output. Whenever a wrong bit appears in the input, it will propagate through the shift register and give rise to a wrong bit, making the two inputs of the comparator different, which will indicate an error. As shown in Figure 2.8, a buffer or a retiming flip-flop may be required at the checker input to reduce the load at that node. Although this checker was shown for a $2^5 - 1$ PRBS, the same idea can be used to make a checker for a PRBS of any length.

This simple checker has two drawbacks. First, for errors that occur rarely (spaced apart by more than the shift register length), the checker indicates 3 errors for every error that is actually

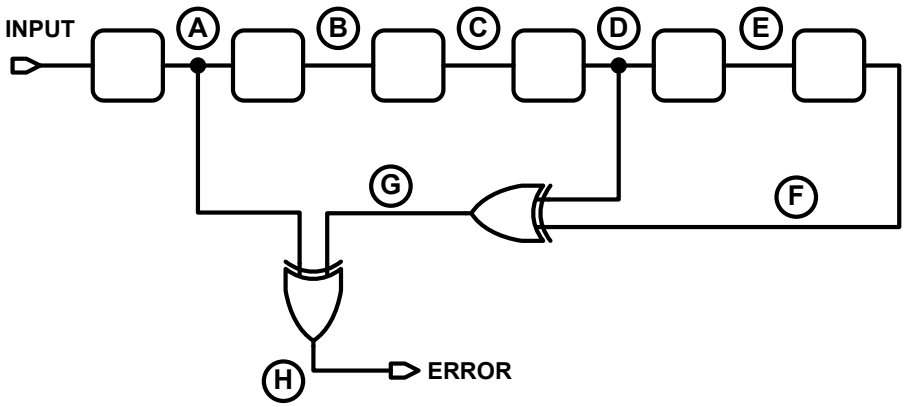


Figure 2.8: A simple PRBS checker

present in the input. Second, for errors that occur more often, the checker indicates 3 or less errors, depending on the spacing of the errors. For precise BER counts, it would be desirable to reduce the variability in the number of detected errors. This problem is also illustrated in Table 2.3 [16]. Letters A-H of the table column names correspond to the node names of Figure 2.8. The left part of Table 2.3 shows how the checker works with correct PRBS input. The absence of errors can be seen because the bits of node A are equal to the bits of column G. The right part of Table 2.3 shows how errors propagate through the checker. Errors e_1 and e_2 (top part of the table) occur far apart in the input and therefore each of them generates 3 error bits in the output node H. However, errors e_3 and e_4 occur close together, and both of them generate only 4 error bits in the output.

The PRBS checker behaviour can also be analyzed mathematically. Assume that the signal at each node, S_{node}, consists of a PRBS bit D^i and an error bit E^i, where i indicates a delay of i clock cycles with respect to a set reference. With reference to Figure 2.8, and noting that $D^3+D^5 = D^0$ (the characteristic polynomial for this PRBS), the signals at the important nodes are:

$$\begin{aligned}
S_A &= D^0 \oplus E^0 \\
S_D &= D^3 \oplus E^3 \\
S_F &= D^5 \oplus E^5 \\
S_G &= S_D \oplus S_F = D^3 \oplus D^5 \oplus E^3 \oplus E^5 = D^0 \oplus E^3 \oplus E^5 \\
S_H &= S_A \oplus S_G = D^0 \oplus E^0 \oplus D^0 \oplus E^3 \oplus E^5 \\
&= E^0 \oplus E^3 \oplus E^5
\end{aligned} \tag{2.11}$$

Indicating that when one error appears at the input node A then three errors appear at the output node H.

An extension of the small $2^5 - 1$ PRBS checker of Figure 2.8 into a more practical $2^7 - 1$ PRBS checker version is shown in Figure 2.9. However, this $2^7 - 1$ PRBS checker, as before,

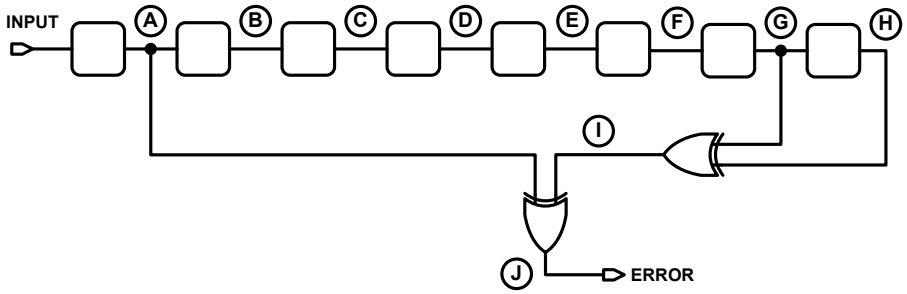


Figure 2.9: A $2^7 - 1$ PRBS checker

	No Errors							Remote Errors or Close Errors							
	A	B	C	D	E	F	G	A	B	C	D	E	F	G	H
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	1 (e_1)	0	1	0	0	0	0	1 (e_1)
3	1	0	0	1	0	0	1	1	1 (e_1)	0	1	0	0	1	0
4	0	1	0	0	1	0	0	0	1	1 (e_1)	0	1	0	0	0
5	1	0	1	0	0	1	1	1	0	1	1 (e_1)	0	1	0	1 (e_1)
6	1	1	0	1	0	0	1	1	1	0	1	1 (e_1)	0	1	0
7	0	1	1	0	1	0	0	0	1	1	0	1	1 (e_1)	1	1 (e_1)
8	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0
9	1	0	0	1	1	0	1	0 (e_2)	0	0	1	1	0	1	1 (e_2)
10	1	1	0	0	1	1	1	1	0 (e_2)	0	0	1	1	1	0
11	1	1	1	0	0	1	1	1	1	0 (e_2)	0	0	1	1	0
12	1	1	1	1	0	0	1	1	1	1	0 (e_2)	0	0	0	1 (e_2)
13	1	1	1	1	1	0	1	1	1	1	1	0 (e_2)	0	1	0
14	0	1	1	1	1	1	0	0	1	1	1	1	0 (e_2)	1	1 (e_2)
15	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0
16	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0
17	1	0	0	0	1	1	1	1	0	0	0	1	1	1	0
18	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0
19	0	1	1	0	0	0	0	1 (e_3)	1	1	0	0	0	0	1 (e)
20	1	0	1	1	0	0	1	1	1 (e_3)	1	1	0	0	1	0
21	1	1	0	1	1	0	1	0 (e_4)	1	1 (e_3)	1	1	0	1	1 (e)
22	1	1	1	0	1	1	1	1	0 (e_4)	1	1 (e_3)	1	1	0	1 (e)
23	0	1	1	1	0	1	0	0	1	0 (e_4)	1	1 (e_3)	1	0	0
24	1	0	1	1	1	0	1	1	0	1	0 (e_4)	1	1 (e_3)	1	0
25	0	1	0	1	1	1	0	0	1	0	1	0 (e_4)	1	0	0
26	1	0	1	0	1	1	1	1	0	1	0	1	0 (e_4)	0	1 (e)
27	0	1	0	1	0	1	0	0	1	0	1	0	1	0	0
28	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0
29	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0
30	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
31	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0
32	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 2.3: PRBS checker behaviour without errors (left), with errors spaced far apart (right top), and with errors close together (right bottom)

has the problems of indicating 3 error pulses for each error in the input, and producing fewer error pulses when the errors in the input are very close to each other. To solve this problem, a new PRBS checker configuration was developed. For a $2^7 - 1$ PRBS, the new configuration is shown in Figure 2.10. The analysis of error propagations inside the checker nodes can be repeated in equation 2.12 for the checker of Figure 2.10, noting that in this case the relationship

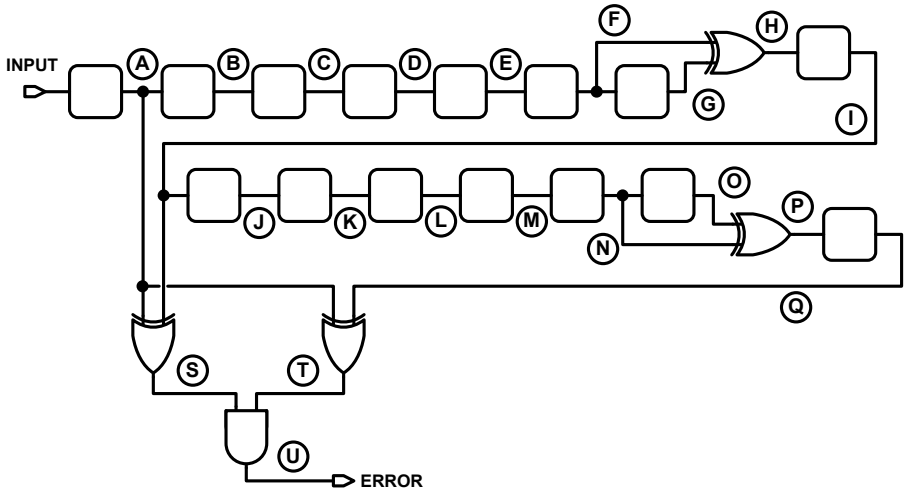


Figure 2.10: A modified $2^7 - 1$ PRBS checker

$D^6 + D^7 = D^0$ holds.

$$
\begin{aligned}
S_A &= D^1 \oplus E^1 \\
S_F &= D^6 \oplus E^6 \\
S_G &= D^7 \oplus E^7 \\
S_H &= S_F \oplus S_G = D^6 \oplus D^7 \oplus E^6 \oplus E^7 = D^0 \oplus E^6 \oplus E^7 \\
S_I &= D^1 \oplus E^7 \oplus E^8 \\
S_N &= D^6 \oplus E^{12} \oplus E^{13} \\
S_O &= D^7 \oplus E^{13} \oplus E^{14} \\
S_P &= S_N \oplus S_O = D^6 \oplus D^7 \oplus E^{12} \oplus E^{13} \oplus E^{13} \oplus E^{14} = D^0 \oplus E^{12} \oplus E^{14} \\
S_Q &= D^1 \oplus E^{13} \oplus E^{15} \\
S_S &= S_A \oplus S_I = D^1 \oplus D^1 \oplus E^1 \oplus E^7 \oplus E^8 = E^1 \oplus E^7 \oplus E^8 \\
S_T &= S_A \oplus S_Q = D^1 \oplus D^1 \oplus E^1 \oplus E^{13} \oplus E^{15} = E^1 \oplus E^{13} \oplus E^{15} \\
S_U &= S_S \wedge S_T = (E^1 \oplus E^7 \oplus E^8) \wedge (E^1 \oplus E^{13} \oplus E^{15}) \\
&= E^1
\end{aligned}
\tag{2.12}
$$

Thus, in this checker configuration, only one error pulse appears for each error in the input signal. This enables more precise counting of errors.

The PRBS checker of Figure 2.10 was developed as a possible replacement for the checker of Figure 2.9 to be able to provide more precise bit error rate (BER) counts. The new checker (Figure 2.10) achieves this goal by ensuring that only one error pulse is produced for each

error in the input. Another possibility for precise BER counts (which was not explored in this thesis) is to modify the basic PRBS checker circuit to ensure that always three error pulses are produced for each error in the input. In this case a precise BER count will be obtained after division by 3. In addition, the rate of occurrence of errors that are spaced close enough to each other to affect BER counts has to be significant to justify the extra circuitry in the new PRBS checker.

2.5. PRBS Generator Architecture Comparison for High-Speed Operation

For very high-speed generation of PRBS sequences, it is useful to know which architecture is optimal for a particular application. The different options that can be compared are parallel versus series PRBS generator configuration and the level of multiplexing. The level of multiplexing determines how much slower, relative to the final output, the core generator can be operated. However, if the multiplexing is too deep, too much power might be spent in the multiplexer itself.

Table 2.4 presents a comparison of series and parallel generators, in terms of the number of gates required and the maximum fanout of gates needed to build the generator. Fanout in the PRBS generator chain determines the maximum speed at which the core generator can be operated for a given gate topology. The number of gates determines the area of the PRBS generator. It is also related to the operation speed because greater area implies the some gates have to drive longer lines, which limits the overall achievable bit rate. The overall power that the generator will consume is also directly proportional to the number of blocks required to build it. Note that the count of blocks indicated in Table 2.4 does not include the blocks required to build the multiplexers needed to increase the core generator bit rate to the final bit rate.

From Table 2.4, it is apparent that parallel PRBS generators outperform series generators in all cases where the PRBS is generated below the full data rate and multiplexing is applied. In practice, for the sequence to be generated at full rate, a more complex and power hungry design is required for each flip-flop in the chain. On the other hand, when multiplexing is used, only the last stage of multiplexing needs to operate at the full data rate. This greatly simplifies the overall design and results in a smaller and more power efficient circuit.

Parallel PRBS generators have several other advantages over series generators in high-speed applications. First the fanout of the XOR gates and flip-flops is uniform throughout the structure, making it easier to design each block and to lay them out. Second, re-timing of each

		Series Architecture			Parallel Architecture		
		No. of Blocks		Max.	No. of Blocks		Max.
Generator Type	Rate	DFFs	XORs	Fanout	DFFs	XORs	Fanout
2^7-1 x^7+x^6+1	full	7	1	2	-	-	-
	1/2	9	2	2	7	2	2
	1/4	11	5	3	7	4	2
	1/8	15	11	3	8	8	2
$2^{15}-1$ $x^{15}+x^{14}+1$	full	15	1	2	-	-	-
	1/2	17	2	2	15	2	2
	1/4	19	5	3	15	4	2
	1/8	23	15	4	15	8	2
	1/16	31	38	8	16	16	2
$2^{23}-1$ $x^{23}+x^{18}+1$	full	23	1	2	-	-	-
	1/2	25	2	2	23	2	2
	1/4	27	6	2	23	4	2
	1/8	31	18	4	23	8	2
	1/16	39	52	6	23	16	2
$2^{31}-1$ $x^{31}+x^{28}+1$	full	31	1	2	-	-	-
	1/2	33	2	2	31	2	2
	1/4	41	6	2	31	4	2
	1/8	39	17	4	31	8	2
	1/16	47	48	7	31	16	2
	1/32	63	186	7	32	32	2

Table 2.4: Comparison of the circuitry required for series and parallel PRBS generators of different sizes

combinational logic gate is essential for correct operation above 10 Gb/s because gate delays are a large fraction of the clock cycle. Conveniently, parallel generators are structured such that all parallel outputs are automatically re-timed and there is only one XOR gate between each two flip-flops. On the other hand, series generators require a very large number of XOR gates to produce appropriately shifted sequences as the multiplexing ratio is increased (Table 2.4) making them highly impractical. Third, since all outputs of the parallel PRBS generators are re-timed, the first stage of multiplexing can be simplified. Instead of the usual high-speed multiplexer that consists of five latches and a selector [17], only one latch and a selector are needed in this case. This further saves power and area.

2.6. Summary

This chapter described how pseudo-random bit sequences can be created in several different ways and what are the benefits and drawbacks of each approach. Series PRBS generators have a very simple structure, but are cumbersome to use when multiplexing to higher speeds. Parallel PRBS generators have more gates in their structure, but generate sequences suitable for direct multiplexing. These properties make parallel generators more suitable for high-speed applications. This chapter also described how pseudo-random bit sequences can be checked for errors.

3 2^7-1 PRBS Generator and Checker Chip Design

In this chapter, the procedure used to design a $2^7 - 1$ PRBS generator and checker system will be outlined. The chip described in this chapter was designed and fabricated in STMicroelectronics' 0.13 μm SiGe BiCMOS process technology. First, system-level considerations will be presented in section 3.1. Then, section 3.2 will present the design details of each building block that was used in the system. Simulation results of the system performance before and after layout will be shown in section 3.3.

3.1. System Description and High-Level Simulations

A $2^7 - 1$ PRBS generator and checker system was designed to be used as an integrated self-test block. The primary design goals of built-in self-test (BIST) blocks are low power consumption and small area. Thus, these were also the requirements of this system. The third requirement was to make the PRBS generator compatible with testing 80-Gb/s systems. Therefore, the generator must be able to generate 4 pseudo-random sequences at 20-Gb/s or higher data rates. A block diagram of the entire system that was implemented on chip is shown in Figure 3.1.

The only high-speed input to the system is a 12-GHz clock signal. The clock signal is distributed by a tree of clock buffers to each of the chip components. The pseudo-random bit sequence is generated at half-rate (12 Gb/s) by the $2^7 - 1$ PRBS generator block. The generator block has 8 outputs and each of them is delayed appropriately for multiplexing. The eight 12-Gb/s PRBS streams are fed directly to an 8-to-4 multiplexer (MUX), which produces four PRBS streams at 24 Gb/s each. One of the four streams is provided off-chip for testing. The other three outputs are terminated on-chip. Even though the four 24-Gb/s signals are generated such that they are ready to be multiplexed further to 80 Gb/s by a 4-to-1 MUX, the 4-to-1 MUX is not integrated on this chip.

The other part of the system is used for verification. In a real testing system, the checker

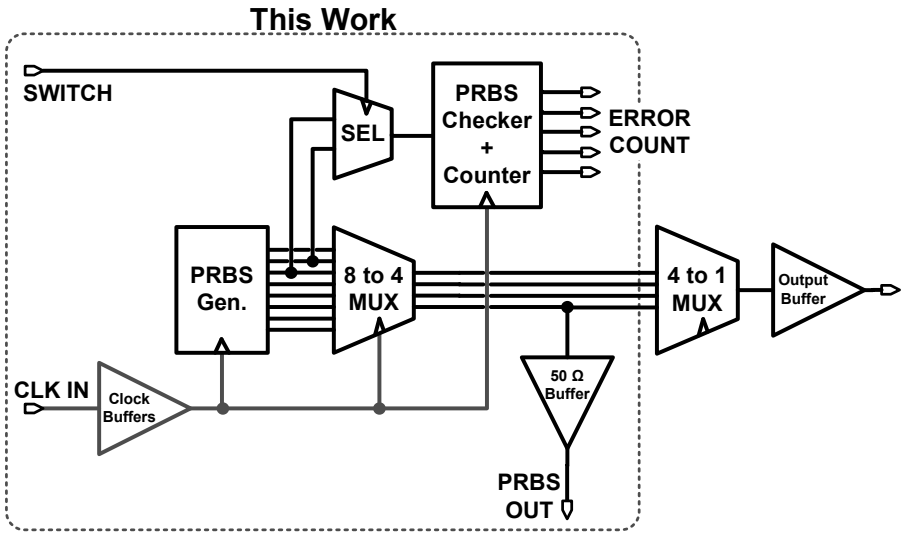


Figure 3.1: Top-level schematic of the designed chip

takes its input signal from the output of the device that is being tested. Then, this signal is de-multiplexed down to the data rate at which the checker operates, and the checker tests the signal for errors. However, in this case, both the PRBS generator and the PRBS checker are on the same chip. To be able to test the checker, its input is taken directly from the generator.

The input signal to the PRBS checker is at 12 Gb/s. Before entering the checker, this signal passes through a selector switch (SEL). The selector can be controlled manually to switch between two PRBS sequences with different phase. At the switching point, errors are introduced into the signal. This signal with manually controlled errors is fed into the checker to enable its verification. The PRBS checker operates at 12 Gb/s and outputs one at-speed pulse for each detected error. These error pulses are then counted by a 5-bit counter. The five error-count bits are provided as outputs for off-chip error monitoring.

3.1.1. MATLAB Simulations

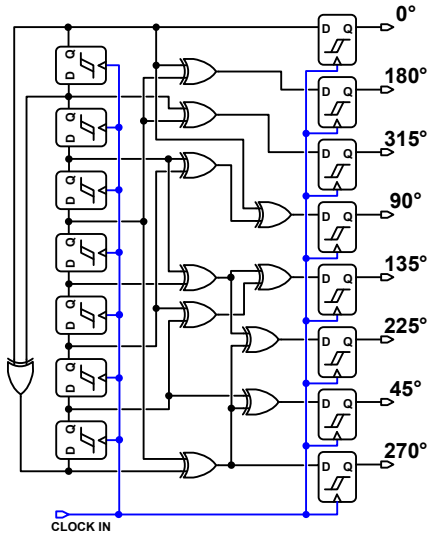
The first decision in the design of the system was to choose between a series or a parallel PRBS generator topology. The parallel topology inherently generates 8 PRBS outputs that are ready for multiplexing. However, for a series topology, the optimal way of generating 8 appropriately delayed signals has to be found. Therefore, the algorithm to determine the connections needed for delayed PRBS signals (Figure 2.4) was implemented in MATLAB. The code is attached in Appendix A.1. This algorithm was run iteratively to find the combination that requires the minimum amount of XOR gates. Note that, because the zero-delay reference is arbitrary, several options exist for selecting the delays even though the delay spacing is fixed.

3.1.2. Simulink Functional Simulations

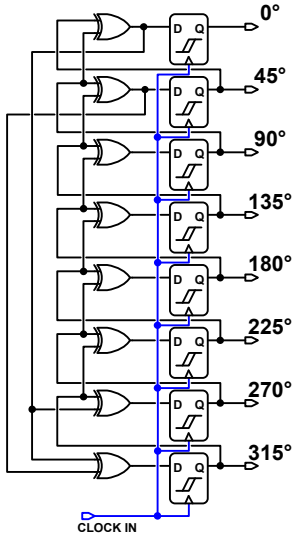
Simulink was used for functional simulation of the PRBS generator, 8-to-4 multiplexer, and checker blocks. It was also employed to verify functional correctness of the parallel PRBS generator by comparing its output to that of the simplest series generator. This was possible to carry out by visual inspection because the sequence length is only 127 bits. These simulations were also used to estimate the power and area (in terms of number of gates) requirements of the parallel and series generator. The final versions of both generators are shown in Figure 3.2.

In the case of the series PRBS generator, re-timing flip-flops are required after the combinational logic to align all signals with the clock, before multiplexing. The second problem with combinational logic is that it requires the fanouts of the shift register flip-flops to be different, and therefore have different delays. Even very small timing variations can significantly affect operation at high speeds. The total number of gates needed in this case is 11 XOR gates and 15 D-flip-flops, resulting in an estimated power of 242 mW for 12-Gb/s operation.

The parallel generator avoids the problems mentioned above thanks to its regular structure. The outputs are automatically re-timed and delayed appropriately. The fanout for all XOR gates and flip-flops is uniform, thus delays are equalized. The total number of gates needed in this case is 8 XOR gates and 8 D-flip-flops, consuming approximately 140 mW at 12 Gb/s. The parallel PRBS generator was chosen to be implemented for this system because it saves area and 42 % power compared to the series generator.



(a) Series $2^7 - 1$ PRBS generator with 8 outputs



(b) Parallel $2^7 - 1$ PRBS generator with 8 outputs

Figure 3.2: Parallel and series implementations of $2^7 - 1$ PRBS generators with 8 outputs

The schematic of the PRBS checker is shown in Figure 2.10. Its operation was also verified with Simulink.

The useful attributes of Simulink are that the simulations can be performed quickly and Simulink's schematics are useful for debugging Cadence schematics. The Simulink schematics are given in Appendix A.2.

3.1.3. Verilog Timing Simulations

After functional verification, Verilog was used for preliminary timing simulations of the generator, the MUX, the PRBS checker, and the clock tree. In these simulations the maximum allowed delay was found for each building block, depending on its fanout. A clock frequency of 15 GHz was used to have margin over the target 12-GHz clock frequency. The maximum allowed block delays, summarized in Table 3.1, were obtained from Verilog simulations and used as design targets in transistor-level design of the building blocks. The Verilog code is located in Appendix A.3.

3.2. Design of Building Blocks

Once high-level system simulations were completed, each individual block was designed at the transistor level and simulated using Spectre. The chip was designed in the STM's 0.13 μm SiGe BiCMOS technology with heterojunction bipolar transistor (HBT) f_T of 160 GHz [18]. This process technology has 6 metal layers. The design procedure and schematics of each block will be given in this section.

3.2.1. 1-mA and 2-mA Latches

Three types of latches were designed for different parts of the system. All three employ the same basic BiCMOS CML topology but have different component values. This is done to

	Fanout = 1	Fanout = 2	Fanout = 3
Maximum XOR delay	16 psec	19 psec	22 psec
Maximum Latch delay	16 psec	19 psec	22 psec

Table 3.1: Maximum allowed block delays for system operation with 15-GHz clock

customize each latch to its load conditions and thus save power where the load (fanout) is small. Schematics for the three latches are shown in Figure 3.3.

To bias the latches for fastest switching time, the tail current I_{tail} and the transistor sizes are related by

$$W = \frac{I_{tail}}{J_{peak-f_T,MOS}} = \frac{I_{taill}}{0.3mA/\mu m} \tag{3.1}$$

for MOSFETs in any technology, and by

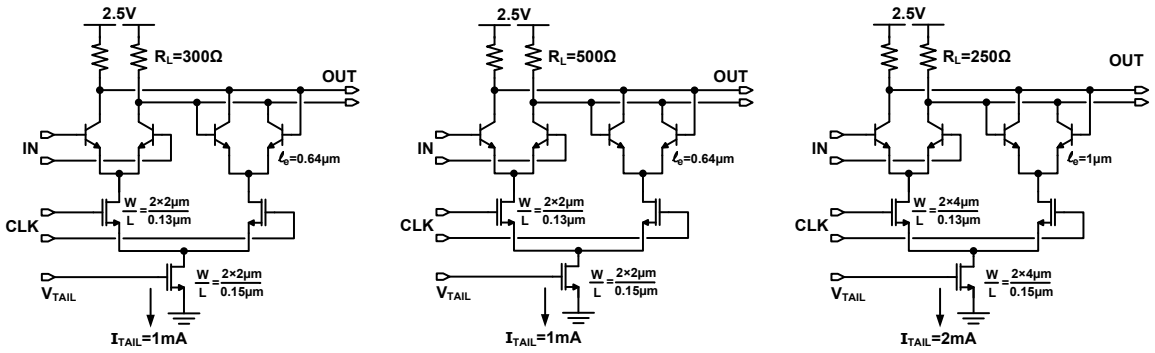
$$l_e \times w_e = \frac{I_{tail}}{J_{peak-f_T,HBT}} = \frac{I_{tail}}{6mA/\mu m^2} \tag{3.2}$$

for HBTs in the 0.13 μm SiGe BiCMOS technology [5].

In this work, it was important to achieve the lowest power consumption possible. Therefore, latches with low fanout, like master latches of D-flip-flops, were designed with the lowest tail current possible that still obeys equations 3.1 and 3.2. The lowest tail current is set by the minimum allowed HBT size in the technology, which is $l_e \times w_e = 0.64\mu m \times 0.2\mu m$. Thus, the tail current was chosen to be 1 mA. Next, the MOSFETs were sized according to equation 3.1. Simulations indicated that the 1-mA latches worked up to 15 Gb/s, which met the design goal, so it was not necessary to increase the current for achieving the desired bit rate.

The output swing $\Delta V = R_L \times I_{tail}$ of the latches was changed depending on the next stage coming after the latch. If the latch was used to drive the HBT pair of a BiCMOS block, ΔV was set to $300mV$, which is adequate to fully switch an HBT differential pair (Figure 3.3(a)). If the latch was used to drive the MOS pair through a stage of emitter followers, then ΔV was set to $500mV$, which is required to switch a MOSFET differential pair in 0.13 μm technology (Figure 3.3(b)).

In places where the fanout of a latch was larger than 2, the latch tail current was increased



(a) 1-mA, 300-mV swing latch (b) 1-mA, 500-mV swing latch (c) 2-mA, 500-mV swing latch

Figure 3.3: Detailed schematics of the designed latches

to 2 mA (Figure 3.3(c)). Transistor sizes were increased accordingly. This configuration was used in the slave latches of D-flip-flops that had to drive two XOR gates, a 2-to-1 MUX, and the associated interconnect.

All latches and gates in this chip use the BiCMOS CML logic topology, but differ from previous designs [6]. In the current design the feedback source followers are removed to save power, and peaking inductors are removed to save area. These changes are permitted because the parallel PRBS architecture allows the shift register to operate at lower bit rates than in [5].

3.2.2. 12-Gb/s DFF

D-flop-flops are used in the core part of the PRBS generator, the PRBS checker and the error counter. A schematic of the DFF is illustrated in Figure 3.4, showing the master and slave latches and the clock emitter followers.

The DFF topology is also an improved version of the one presented in [6]. The clock source followers are replaced by emitter followers which are able to drive a larger capacitance per unit current. Notice that the DFF contains only one set of emitter followers for both latches. This is done to both reduce the load on the clock distribution buffers and to save power compared to a DFF configuration where each latch has its own emitter followers.

The D-flip-flops used in the PRBS generator have the 1-mA latch of Figure 3.3(a) as the master and the 2-mA latch of Figure 3.3(c) as the slave. The slave latch of each generator DFF needs a larger tail current because it has to drive two XOR gates and a 2-to-1 MUX. Together with the clock emitter-followers, this results in a current of 5 mA from 2.5 V, thus a power dissipation of only 12.5 mW for a DFF that operates at 12 Gb/s.

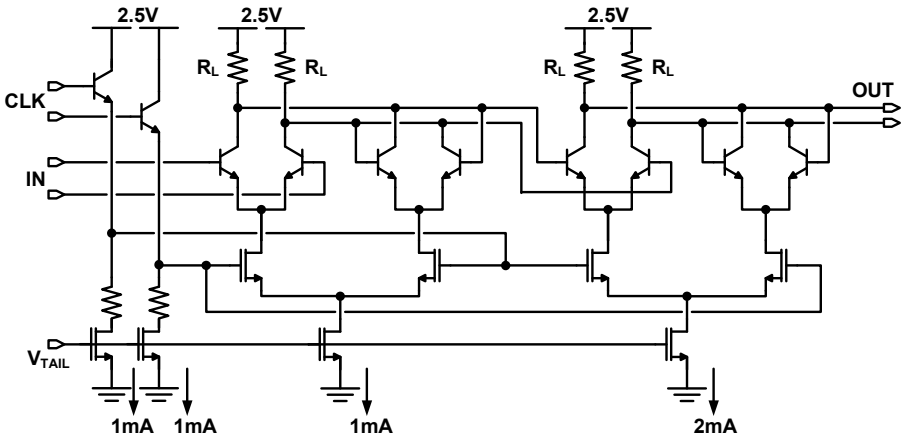


Figure 3.4: D-flip-flop schematic

Unlike the flip-flops of the generator chain, in the PRBS checker and counter most flip-flops have a fanout of 1. In this case, the latches of Figure 3.3(a) and 3.3(b) are used as the master and slave, respectively, for a total 12-Gb/s DFF power of 10 mW.

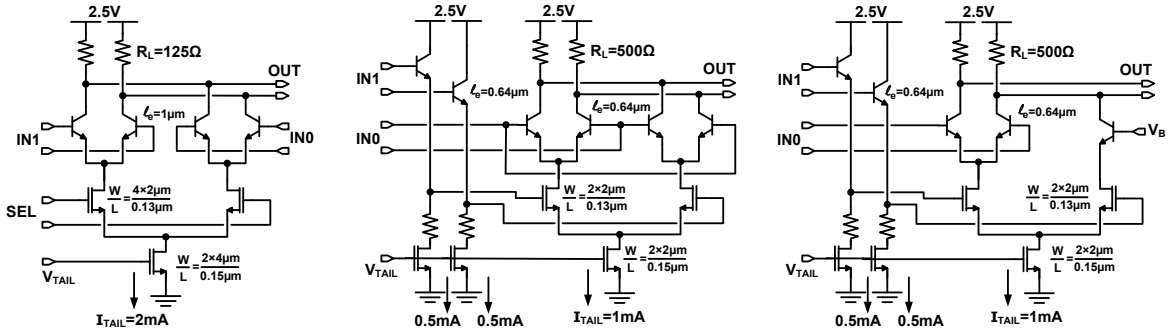
3.2.3. Selector, XOR, and AND Gates

In addition to latches, the other digital blocks that are used in this system include selectors, XOR gates, and AND gates. Their schematics are shown in Figure 3.5. Selectors (Figure 3.5(a)) are employed in the final stage of each 2-to-1 MUX. To achieve a 24 Gb/s signal at the output, the tail current was chosen to be 2 mA, with a single-ended swing of 250 mV. Transistors were sized by following the same procedure as for the latch.

XOR gates (Figure 3.5(b)) and AND gates (Figure 3.5(c)) are designed with 1-mA tail currents because their fanout is 1 in most cases. However, they differ from the latch and the selector topology by having emitter-followers at one of the inputs. These emitter-followers are necessary to step-down the DC voltage level from the top HBT pair to the bottom MOS transistor pair; they cannot be shared between gates.

3.2.4. 24-Gb/s 2-to-1 MUX

The 2-to-1 MUX block is repeated four times to build the 8-to-4 MUX that outputs four 24-Gb/s PRBS streams. The 2-to-1 MUX schematic is shown Figure 3.6. Note that only one latch and one selector are used to build the MUX, unlike the more common 5 latches and selector configuration [17]. This is acceptable because the signals going from the PRBS generator into the MUX are already re-timed.



(a) Selector (b) XOR gate (c) AND gate

Figure 3.5: Detailed schematics of other blocks

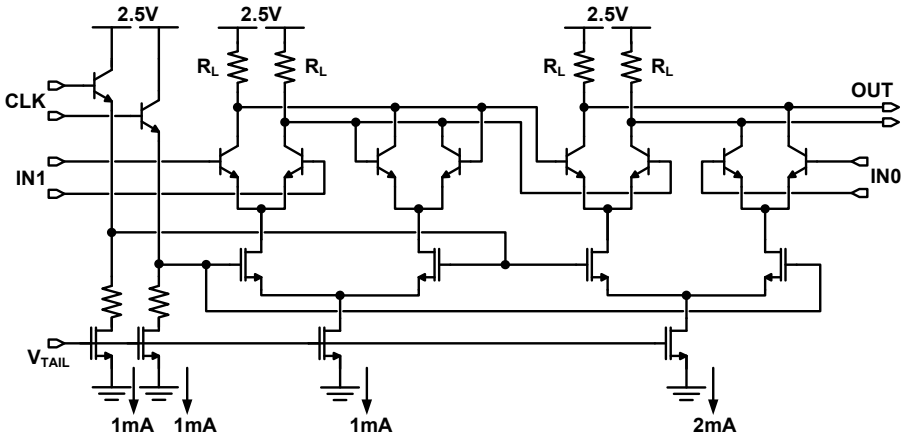


Figure 3.6: 2-to-1 MUX schematic

Since the non-latched input to the selector comes from the generator DFFs, which have 500 mV swing, the latched input must also have 500 mV swing. Therefore, a 1-mA latch with 500 mV swing (Figure 3.3(b)) is used in the 2-to-1 MUX in front of the selector. The clock emitter followers are shared between the latch and the selector of the 2-to-1 MUX, as in the DFF.

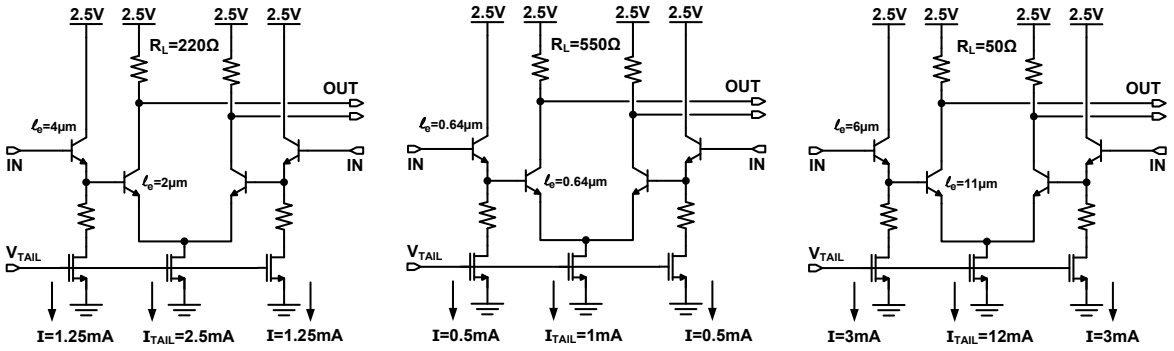
3.2.5. Clock, Data, and Output Buffers

One of the most important parts of the PRBS generator and checker system is the clock tree. It is a tree of CML buffers designed to deliver the 12-GHz clock signal synchronously to all latches in the system. The schematic of one clock buffer is shown in Figure 3.7(a). It consists of an HBT differential pair preceded by emitter followers. The swing is set to 500 mV, to be able to switch the MOS transistors at the clock inputs of the latches. The tail current in the differential pair is set to 2.5 mA for adequate bandwidth.

To reduce the number of clock buffers in the system, and thus to save power, the fanout of each buffer is set to 4. This high fanout is possible because in each flip flop, the emitter followers on the clock path are shared among the two latches. They also serve as the final stage of clock buffering.

It is very important to ensure that the paths traveled by the clock signal have identical delays. This ensures that the clock arrives to all flip-flops with the same phase. Thus, a lot of attention was payed in the layout to ensure equal-length connections between clock buffers and from the clock buffers to the flip-flops.

In addition to clock buffers, two other types of buffers are used in the system. These are data buffers and 50-Ω output buffers, shown in Figure 3.7(b) and Figure 3.7(c) respectively.



(a) Clock buffer schematic (b) Data buffer schematic (c) 50Ω output buffer schematic

Figure 3.7: Detailed schematics of buffers

Data buffers are employed as intermediate buffers to enhance the signal, or before driving a large load. 50-Ω output buffers are used only on the outputs, to drive external 50-Ω loads. The 50-Ω load and the 300-mV swing requirement restrict the tail current in these buffers to be 12 mA.

3.3. System-Level Design and Simulations

A detailed schematic of the entire system that was implemented on-chip is illustrated in Figure 3.8. The power consumption of each sub-block is summarized in Table 3.2.

Throughout the design, every building block, such as a latch or an XOR gate, was extensively simulated at the transistor level using Spectre to ensure that it behaves as desired. Simulated eye diagrams of a latch and a D-flip-flop with a 15-GHz clock input are shown in Figure 3.9.

	Power Consumption
$2^7 - 1$ PRBS Generator	145 mW
8-to-4 MUX	50 mW
PRBS Generator Clock Tree	75 mW
$2^7 - 1$ PRBS Error Checker	200 mW
5-Bit Error Counter	80 mW
PRBS Checker Clock Tree	75 mW
Output Buffers	300 mW
Total	925 mW

Table 3.2: Power consumption of the chip sub-blocks

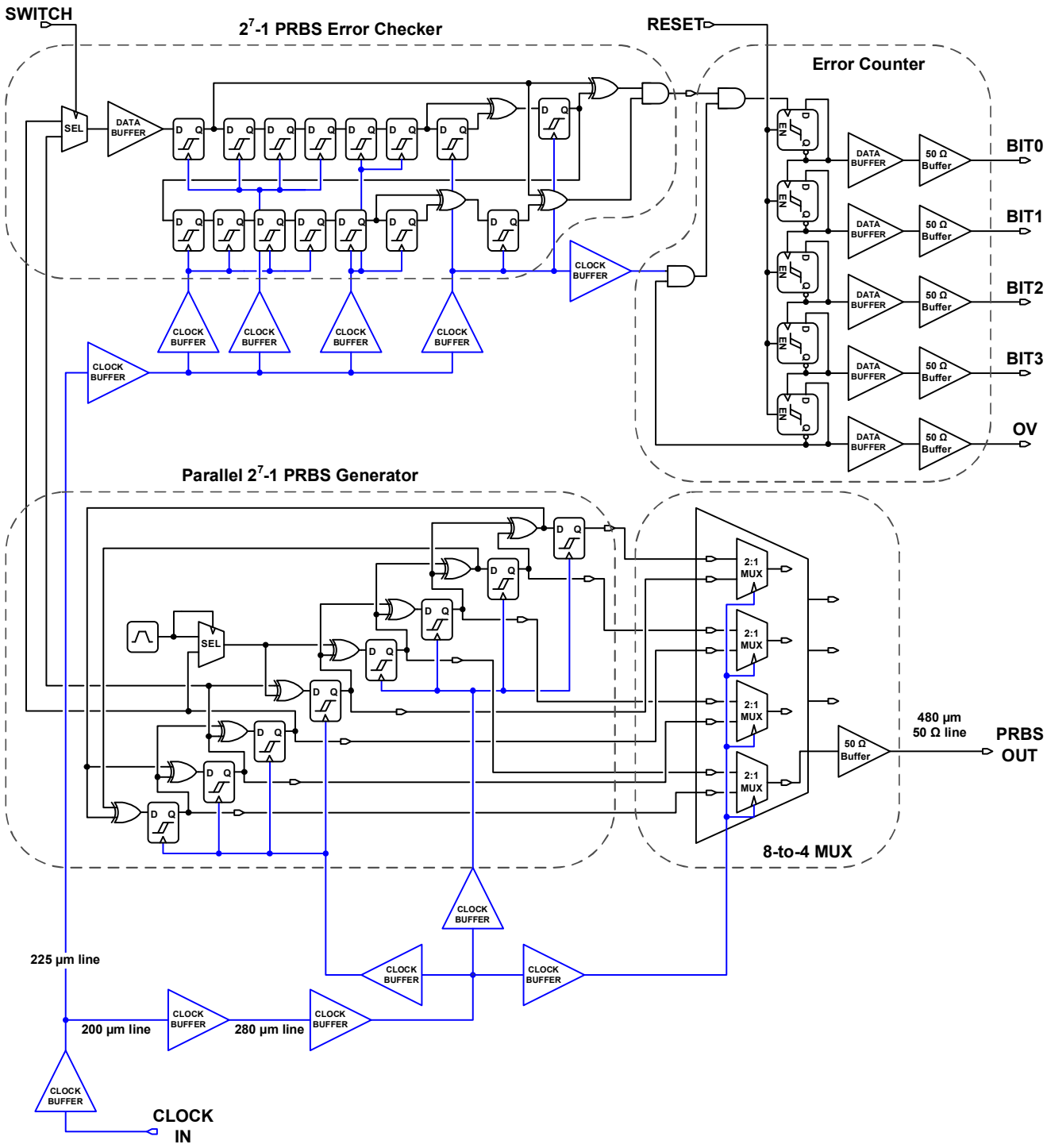


Figure 3.8: Detailed system schematic

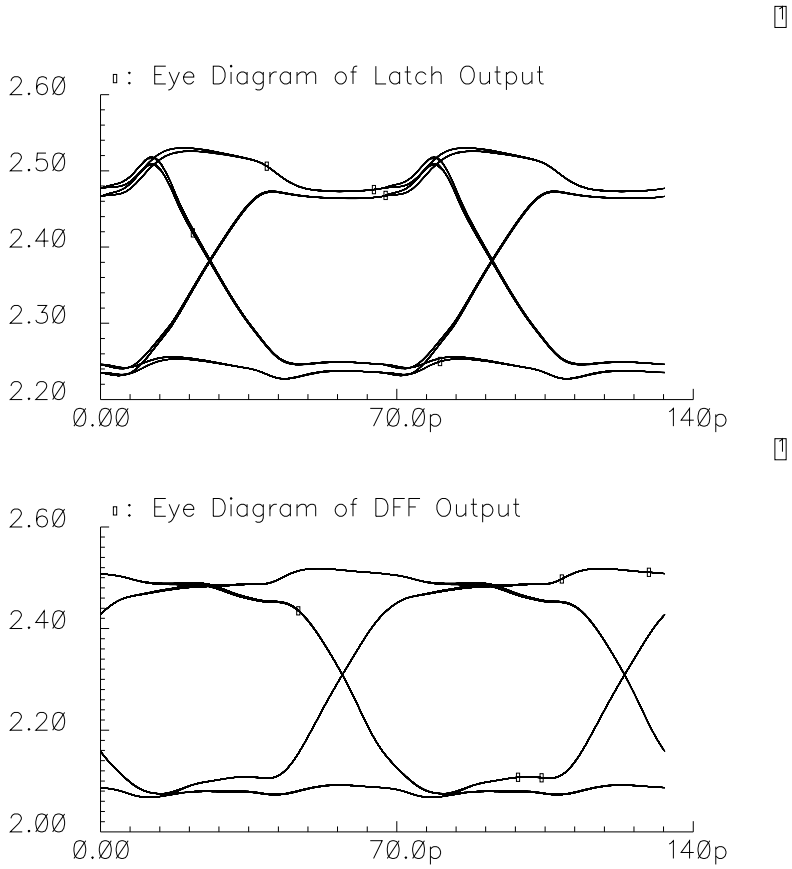


Figure 3.9: Latch (top) and DFF (bottom) simulation with 15-GHz clock

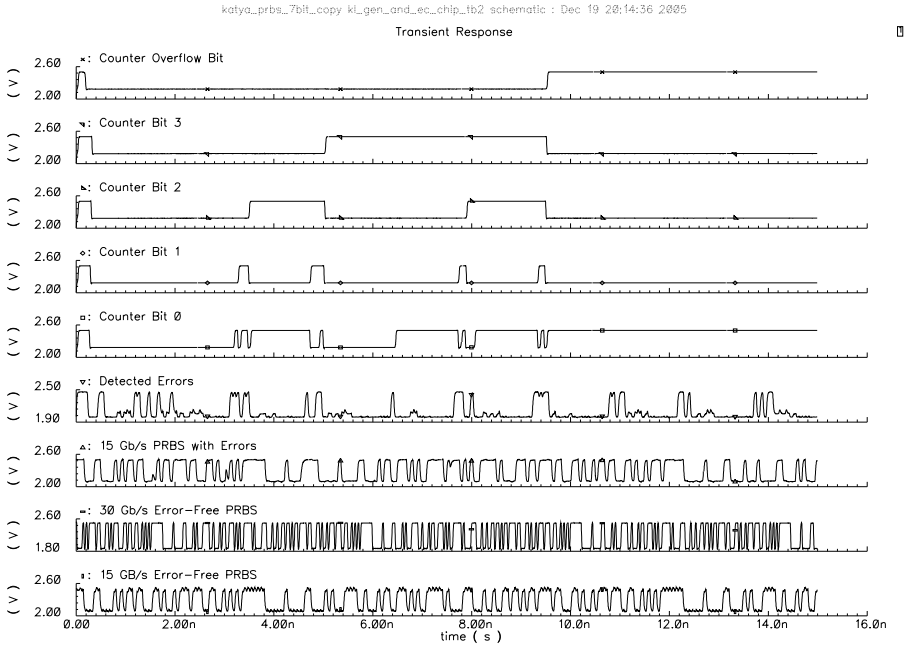
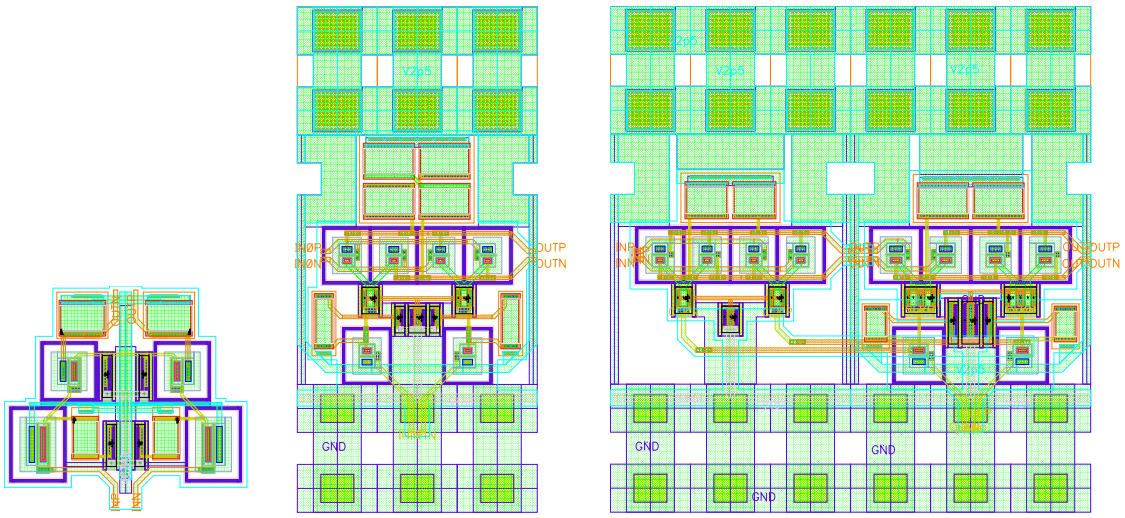


Figure 3.10: System simulation with 15-GHz clock

After the operation of building blocks was verified, the system sub-blocks, like the generator and the checker, were simulated by themselves. Finally, the sub-blocks were assembled, and the entire system was simulated, including a model of the test setup. A simulation of the system with input clock frequency of 15 GHz is shown in Figure 3.10. Figure 3.10 shows (from bottom to top) the 15-Gb/s error-free $2^7 - 1$ PRBS signal coming from the generator; the 30-Gb/s output signal from the 8-to-4 MUX; the distorted 15-Gb/s input to the PRBS checker; the errors detected by the checker; and the 5-bit count of the error counter.

During the layout design process, a lot of attention was paid to the symmetry of each building block, and to the interconnect between blocks to minimize the impact of transistor variations and mismatches. All blocks used unit-size transistors with identical orientation to form all other transistors sizes. Since all signals are differential, the path lengths of the two wires carrying the signal were equalized, especially in the clock distribution tree. 1-pF MIM capacitors were added in the current mirror for decoupling. Additional decoupling capacitors were connected between ground and VDD for filtering of power supply noise. Metal 1 and metal 2 were used for ground and VDD, respectively. All high-speed signals were routed in metals 5 and 6 to reduce their capacitance to ground. Control signals were routed using metal 3. The layouts of the clock buffer, the XOR gate, and the flip-flop and given in Figure 3.11.

After the layout was completed, simulations with extracted parasitics were carried out. The extracted parasitics included series resistances, shunt, and coupling capacitances of all nodes. A simulation of the system after extraction is shown in Figure 3.12. In this figure, the



(a) Clock buffer layout ($30\mu m \times 25\mu m$) (b) XOR gate layout ($30\mu m \times 63\mu m$) (c) D-flip-flop layout ($60\mu m \times 63\mu m$)

Figure 3.11: Layouts (from left to right) of the clock buffer, XOR gate, and flip-flop (2 latches and emitter followers)

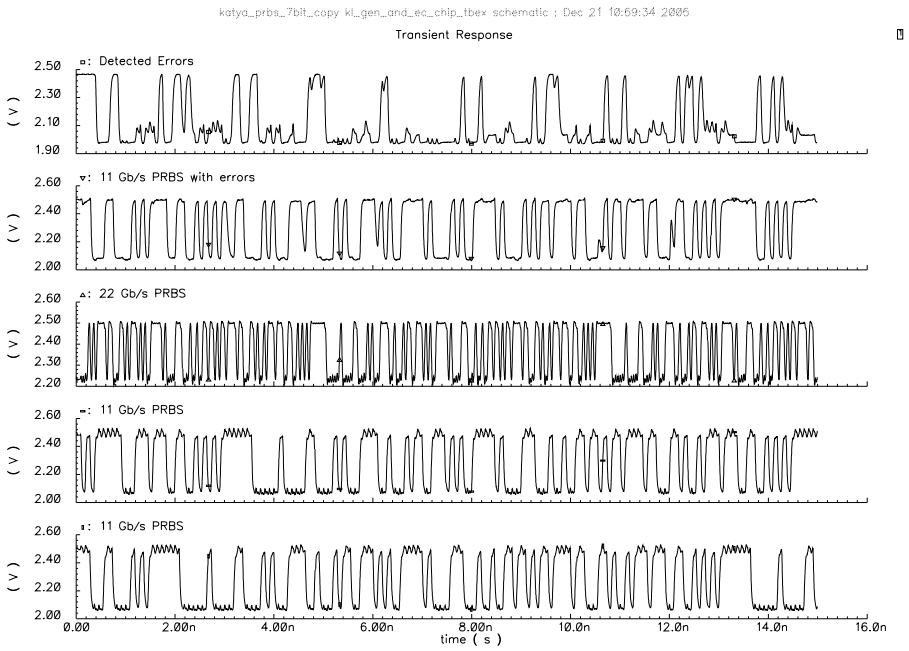


Figure 3.12: System simulation after extraction with 11-GHz clock

signals (from bottom to top) are: two 11-Gb/s error-free $2^7 - 1$ PRBS signals coming from the generator; the 22-Gb/s output signal from the 8-to-4 MUX; the distorted 11-Gb/s input to the PRBS checker and the errors detected by the checker.

3.4. Summary

This chapter described the design considerations for the $2^7 - 1$ PRBS generator and checker chip, starting from high-level MATLAB and Verilog simulations. Detailed schematics of all system blocks were shown, together with the ideal and extracted simulation results of the entire chip. The chip was designed and fabricated in STMicroelectronics' 0.13 μm SiGe BiCMOS process technology.

4 Experimental Results

This chapter will describe the experiments performed to measure the performance of the fabricated $2^7 - 1$ PRBS generator and checker. The chip is shown in section 4.1. Section 4.2 will present the measurement procedure of the PRBS generator and the obtained results. Similarly, the measurement procedure and results for the PRBS checker will be described in section 4.3.

4.1. Fabrication and Test Equipment

The chip was designed in the STM's 0.13 μm SiGe BiCMOS technology with HBT f_T of 160 GHz [18]. The die photo of the fabricated chip is shown in Figure 4.1, with the PRBS generator and checker identified. The total, pad-limited chip area is $1mm \times 0.8mm$. The PRBS generator and 8-to-4 MUX together occupy and area of $393\mu m \times 178\mu m$ and consume 235 mW. The PRBS checker and error counter have an area of $308\mu m \times 349\mu m$ and power consumption of 350 mW. The rest of the power is consumed in the output buffers, adding up to a total measured power consumption of 940 mW.

The fabricated chip was tested using an Agilent E4448A PSA series spectrum analyzer for verifying the bit rate and periodicity of the generated PRB-sequence on one of the two differential outputs. Furthermore, an Agilent 86100C DCAJ oscilloscope was employed to monitor the other differential output. The oscilloscope is capable of identifying, locking, and characterizing the jitter of digital sequences as long as $2^{15} - 1$ at data rates beyond 40 Gb/s. In the absence of a 40-Gb/s BERT, use of the oscilloscope was essential for confirming the correctness of the generated sequence.

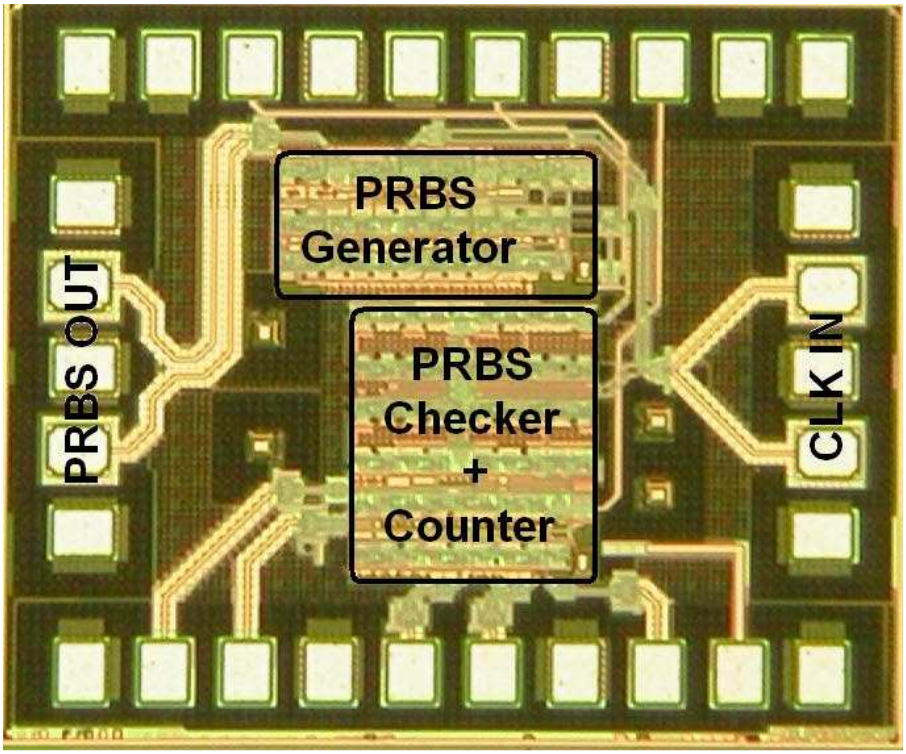


Figure 4.1: Die photo of the fabricated chip

4.2. 2^7-1 PRBS Generator Measurements

On-chip testing of the $2^7 - 1$ PRBS generator was performed first. The test setup for generator measurements and the results are described next.

4.2.1. Generator Test Setup

A detailed measurement setup for the PRBS generator circuit is shown in Figure 4.2. A 20-GHz signal source is used to generate the clock, which is passed through a 40-GHz-bandwidth power splitter. One output of the splitter is used to synchronize the oscilloscope, and the other output is used as the clock input to the chip. The clock signal is applied to only one side of the differential clock input. To be able to bias the clock input at the right DC voltage, it is passed through a bias-tee before connecting it to the chip. The other clock input is also connected to a bias-tee, and then terminated in 50 Ω.

The input clock and the output PRBS signal are provided onto and off the chip using differential 67-GHz GSGSG probes. The output signal was taken from both sides of the differential output. One side was connected through a DC-blocking capacitor to the remote head of the

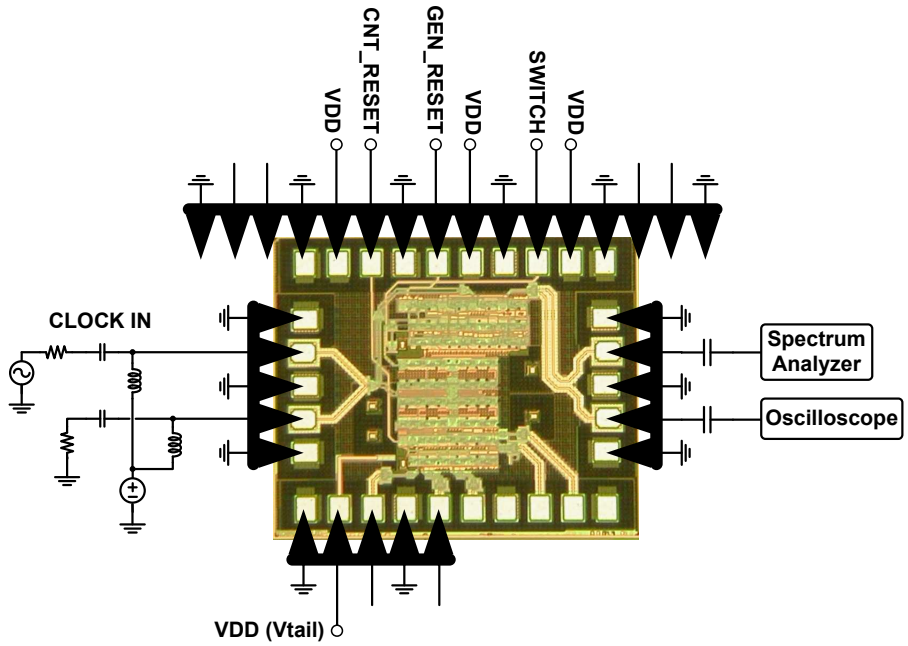


Figure 4.2: Measurement setup for PRBS generator

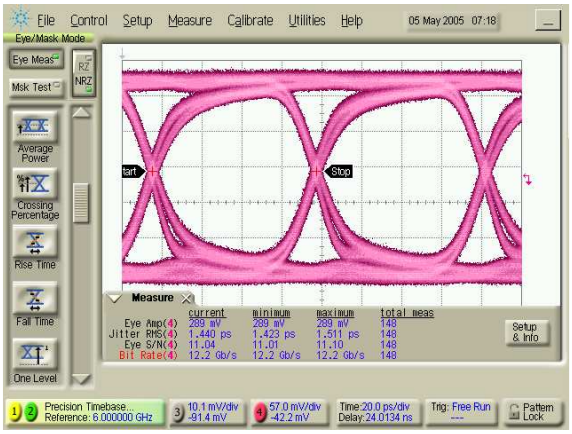
digital oscilloscope. The other output was connected through another blocking capacitor to the spectrum analyzer.

To provide all the necessary DC signals and supply current to the generator, a GPPGPPGPPGPPGPPG DC probe was used on the top side of the chip and a PGPPG probe was used on the bottom. The top DC probe used is bigger than required because there was no DC probe of the right size available. The unconnected pads on the bottom are the outputs of the error counter. A second DC probe (like the one on top), could not be connected on the bottom, because it did not fit on the probe station. Therefore, during the first round of testing, only the PRBS generator could be tested.

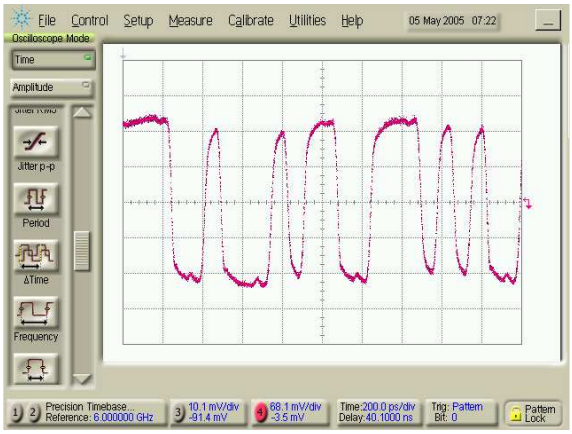
4.2.2. Generator Measurement Results

The $2^7 - 1$ PRBS generator (together with the 8-to-4 MUX) was tested by first applying a relatively slow clock signal and verifying the correctness of the generated sequence. The measurement results of the 12-Gb/s PRBS are shown in Figure 4.3 with a 6-GHz clock signal.

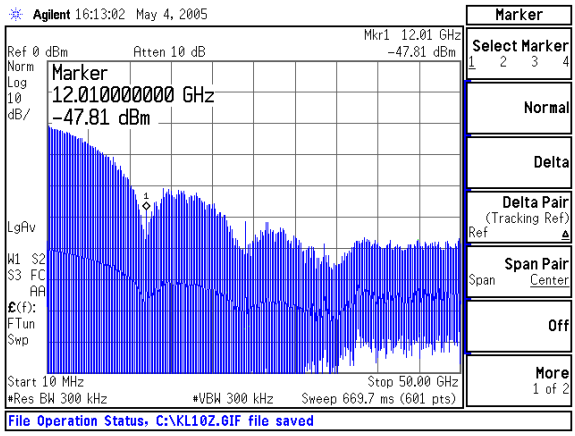
Figure 4.3(c) shows the spectrum of the 12-Gb/s PRBS output. It has a $\sin(x)/x$-type shape with nulls at multiples of the clock frequency, indicating NRZ logic. A zoomed-in version of the same spectrum is shown in Figure 4.3(d), with spectral tones spaced apart by 94.5 MHz. This tone spacing is equal to the bit rate divided by the sequence length $94.5MHz = \frac{12\,Gb/s}{127\,bits}$,



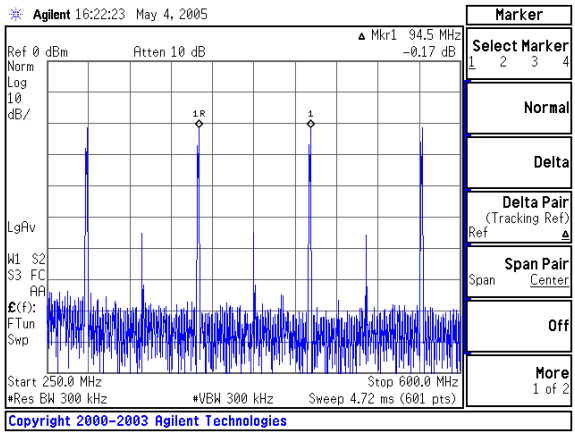
(a) Eye diagram of the generated PRBS at 12 Gb/s



(b) Locked time-domain sequence at 12 Gb/s



(c) Spectrum of the generated PRBS at 12 Gb/s



(d) Spectrum of the generated PRBS at 12 Gb/s (zoomed)

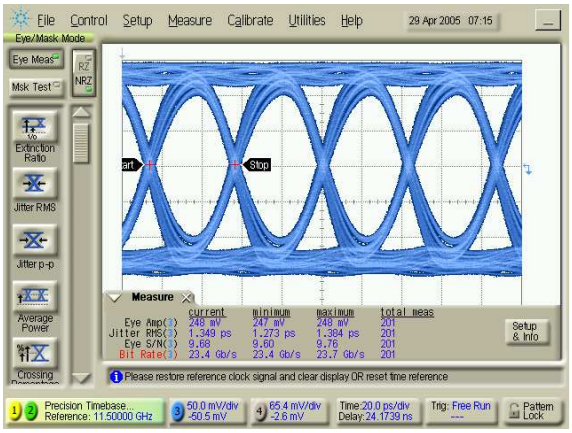
Figure 4.3: Measurement results of the PRBS generator at 12-Gb/s

indicating that the correct pattern length of 127 bits is achieved. Figure 4.3(a) shows a fully-open eye diagram at 12 Gb/s. However, this does not guarantee that every bit of the generated sequence is correct. To confirm the correctness of the sequence, the oscilloscope was locked to a 127-bit long pattern, and the pattern was checked bit-by-bit by scrolling through it (Figure 4.3(b)).

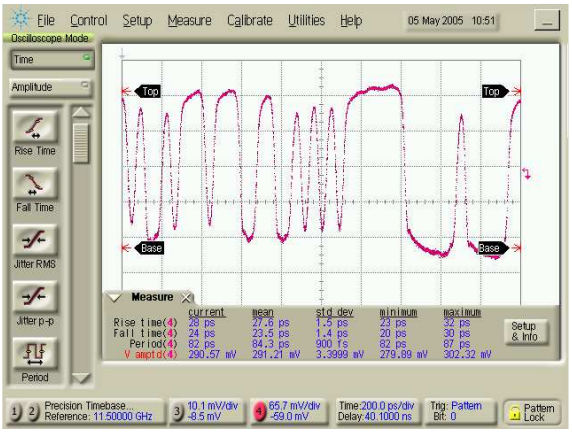
The same procedure was repeated for output data rates ranging from 8 Gb/s up to 24 Gb/s in 2-Gb/s steps. Output characteristics at these bit rates are summarized in Table 4.1. The highest bit rate at which the PRBS generator was found to work correctly is 23 Gb/s. Plots confirming correct operation at 23 Gb/s are shown in Figure 4.4. The time-domain sequence is correct (Figure 4.4(b)) and the spectral tones are spaced apart by $180.9MHz = \frac{23\,Gb/s}{127\,bits}$ (Figure 4.4(d)).

Output Bit Rate	Jitter (rms)	Eye Amplitude	Rise Time	Fall Time	Eye SNR
8 Gb/s	1.39 psec	299 mV	23.3 psec	21.1 psec	14.58
10 Gb/s	1.25 psec	294 mV	22.2 psec	21.1 psec	12.57
12 Gb/s	1.44 psec	289 mV	22.67 psec	16.0 psec	11.04
14 Gb/s	1.453 psec	282 mV	21.78 psec	15.56 psec	10.07
16 Gb/s	1.534 psec	276 mV	22.22 psec	16.0 psec	9.32
18 Gb/s	1.451 psec	268 mV	20.44 psec	18.67 psec	9.29
20 Gb/s	1.337 psec	251 mV	-	-	11.07
22 Gb/s	1.518 psec	257 mV	20.67 psec	18.67 psec	7.75
23 Gb/s	1.349 psec	248 mV	20.0 psec	14.44 psec	9.68
24 Gb/s	1.276 psec	268 mV	-	-	8.02

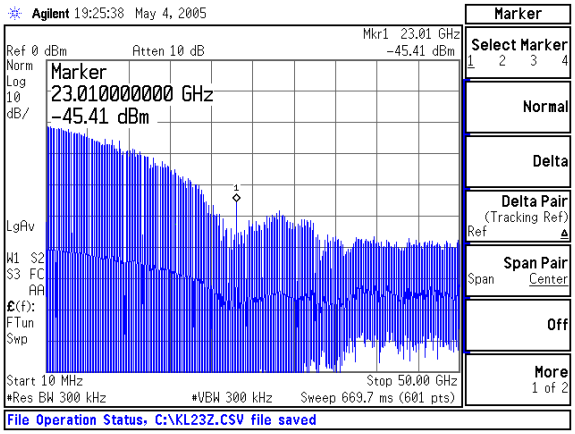
Table 4.1: Performance summary of the PRBS generator at different bit rates



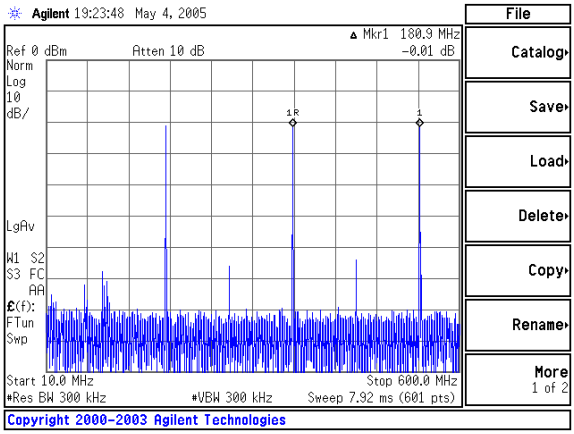
(a) Eye diagram of the generated PRBS at 23 Gb/s



(b) Locked time-domain sequence at 23 Gb/s



(c) Spectrum of the generated PRBS at 23 Gb/s



(d) Spectrum of the generated PRBS at 23 Gb/s (zoomed)

Figure 4.4: Measurement results of the PRBS generator at 23-Gb/s

To further illustrate that the correct sequence is generated at bit rates up to 23 Gb/s, the output was saved using the oscilloscope, and plotted against an ideal $2^7 - 1$ PRBS generated using MATLAB, as shown in Figure 4.5. Correct PRBS generation was also obtained with clock frequencies as low as 100 MHz, demonstrating the very wide bandwidth of the PRBS generator.

With a 12-GHz input clock and a 24-Gb/s output, an open eye was obtained (Figure 4.6(a)). Also, the spectrum tones have the right spacing of $189.2MHz = \frac{24\,Gb/s}{127\,bits}$ (Figure 4.6(c)). However, the oscilloscope could not be locked to the sequence to observe it in time domain. Therefore, even though all logic blocks inside the generator operate up to 24 Gb/s, their delay relative to the clock cycle time limits error-free PRBS generation to 23 Gb/s. The $2^7 - 1$ PRBS generator produces 4, appropriately delayed, parallel output streams at 23 Gb/s each, which can be further multiplexed to an aggregate PRBS output at 92 Gb/s with minimal circuitry. The 4-channel PRBS generator consumes 235 mW from 2.5 V, which results in only 60 mW per output lane.

The PRBS generator produces an output at 24 Gb/s, which means that, in the generator core, latches that consume 2.5 mW are switching at 12 Gb/s. To the best of my knowledge, this is the lowest power latch operating above 10 Gb/s in any technology [19]. This BiCMOS CML latch implementation works with 1-mA tail current from a 2.5-V supply. Other recently reported sub-3.3V bipolar logic families [20–22] consume significantly more power because they require doubling the tail current for a given logic function. While 130-nm or 90-nm MOS CML latches operate from 1.5-V or lower supplies, they require more than 2 times higher tail currents and inductive peaking to operate above 10 Gb/s, thus offsetting the advantage provided by the lower supply voltage [23].

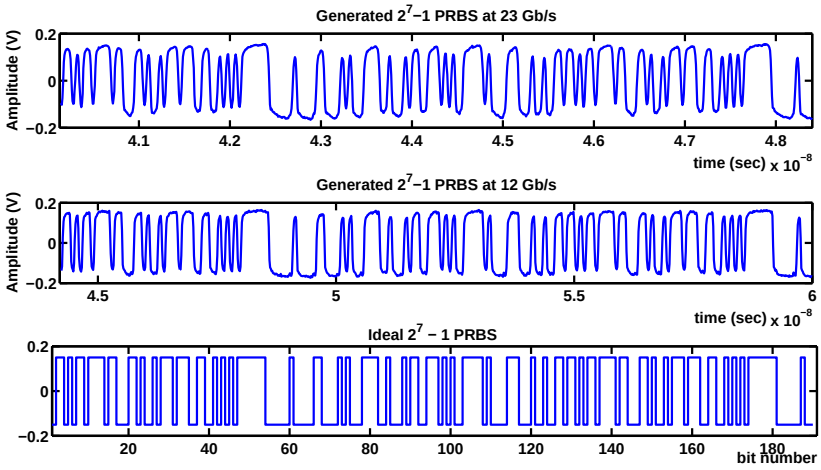
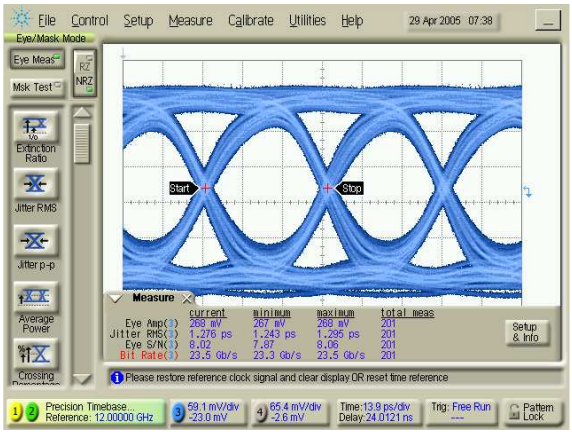
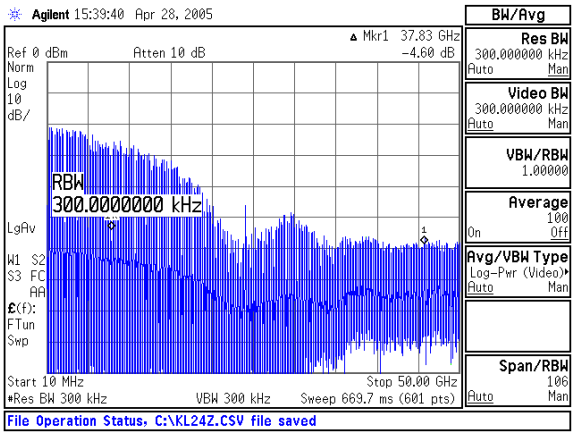


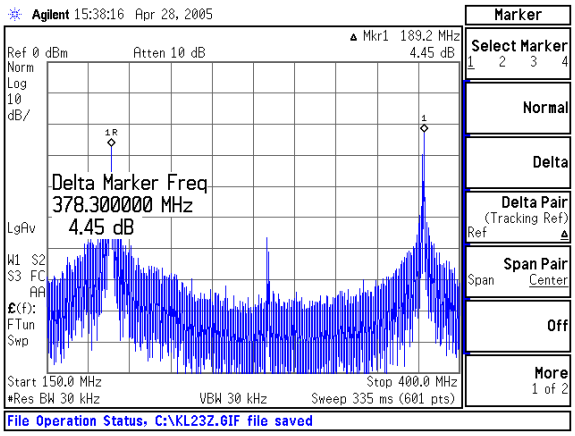
Figure 4.5: Measured 23-Gb/s (top), measured 12-Gb/s (middle), and ideal (bottom) time domain $2^7 - 1$ PRB-sequences



(a) Eye diagram of the output at 24 Gb/s



(b) Spectrum of the generated PRBS at 24 Gb/s



(c) Spectrum of the generated PRBS at 24 Gb/s (zoomed)

Figure 4.6: Measurement results of the PRBS generator at 24-Gb/s

4.3. PRBS Checker Measurements

This section presents the test setup and measurement results of the PRBS checker and error counter. These tests were performed a few months after the generator measurements.

4.3.1. Checker Test Setup

The test setup for the PRBS checker, shown in Figure 4.7, is similar to the generator test setup. In this case GPPGPPGPPG DC probes were used both on the top and at the bottom to be able to apply all DC inputs to the chip and to have access to all counter outputs. As before, the output PRBS was observed using a spectrum analyzer and a digital oscilloscope to confirm correct

operation of the generator during checker measurements. The counter outputs were connected to another, low speed, oscilloscope.

4.3.2. Checker Measurement Results

As was shown previously in the chip schematic of Figure 3.8, access to the PRBS checker is possible only through the generator on the input side, and only through the error counter on the output side. The input to the checker comes from the PRBS generator though a selector. The selector is manually controlled by a power supply to switch between two possible inputs to the checker. At the switching time, the checker is not synchronized to the input from the generator, and errors result. The output of the checker is accessible only as a bit count from the 5-bit error counter.

When measuring, the PRBS generator was first set up to generate a correct output, which means that a correct PRBS was applied to the checker. At this point all outputs of the counter have to be at logic 0. Then, the voltage value of the SWITCH input was changed to introduce errors into the checker. However, as shown in Figure 4.8, the error counter outputs were observed to switch all the time, without regard to the correctness of the generator output. This

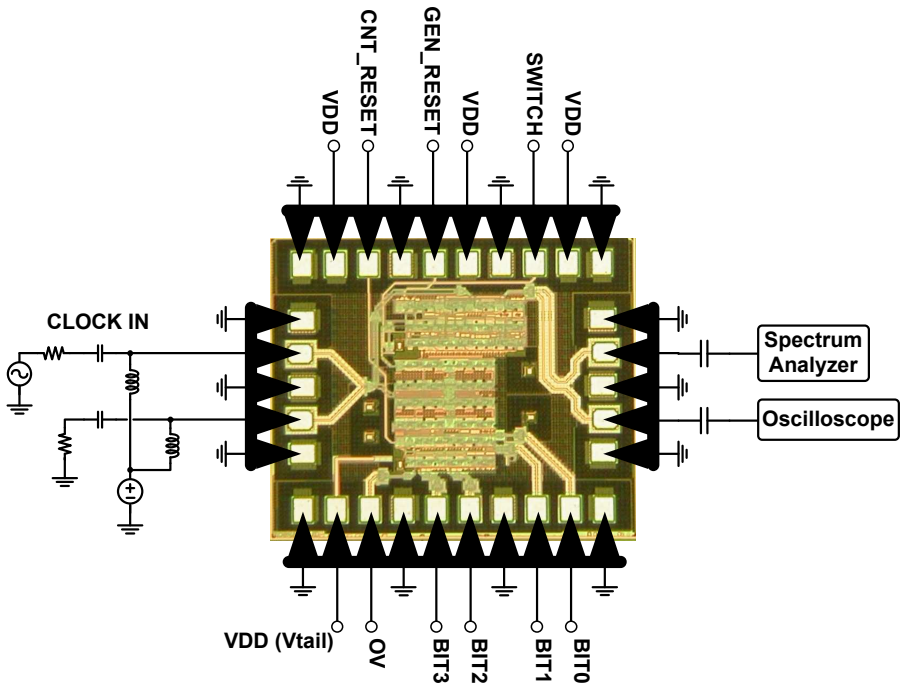


Figure 4.7: Measurement setup for PRBS checker

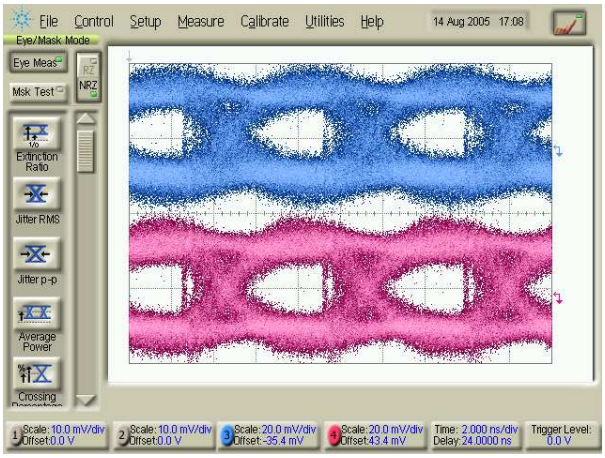


Figure 4.8: Bit0 (bottom) and Bit1 (top) outputs of the error counter

means that errors were detected by the checker, and error pulses were produced constantly, even with a correct PRBS input.

The possible cause of this problem is that there is no synchronization of the clock with the data on-chip. Instead, the PRBS output from the generator is connected directly to the error checker by a $200\,\mu m$-long transmission line. However, interconnect and circuit delays may offset the checker clock signal with respect to its data input. In this case, the checker would always observe a wrong PRBS input and produce pulses, which are then counted by the error counter.

Originally, the plan was to have a separate input into the checker. However, due to the lack of high-speed probes, the design was changed such that the error checker input comes directly from the PRBS generator. A variable delay block is needed between the PRBS generator and the error checker to compensate for the propagation delay of the logic and along the connecting line.

Thus, correct functionality of the PRBS checker could not be detected during this round of testing. More measurements will have to be performed to clarify the exact cause of the problem and solve it.

4.4. Summary

Measurement results of the fabricated PRBS generator and error checker chip were presented in this chapter. The $2^7 - 1$ PRBS generator was found to operate correctly up to 23 Gb/s. The individual generator blocks switch at a bit rate of 24 Gb/s. Thus, individual latches that consume 2.5 mW worked at 12 Gb/s. To the best of my knowledge, this represents the lowest

power latch operating above 10 Gb/s in any technology. Unfortunately, there were problems with testing the error checker and counter.

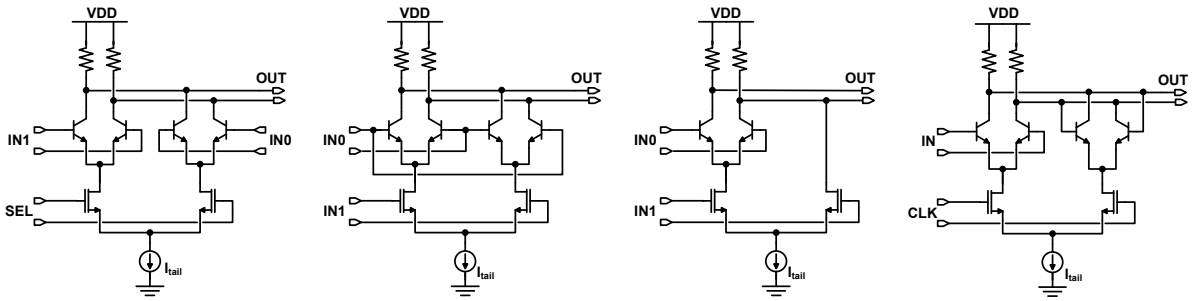
5 Comparison and Power Optimization of High-Speed Logic Topologies

In the previous chapters various PRBS generator topologies were compared in terms of their power and speed characteristics, and the fabricated chip was described. In this chapter avenues for further speed improvements and possible power reduction of high speed digital blocks will be presented. This chapter will concentrate on the transistor-level gate design. Several high-speed logic families (such as CML and CMOS) will be compared in terms of their speed performance, power and area requirements.

In section 5.1 an overview of existing high-speed digital gates will be given. Current-mode logic latch design will be outlined in section 5.2. Possible improvements to the existing latch topology will be presented in section 5.3. The performance of different BiCMOS latches will be compared in section 5.4 and a comparison with 65-nm CMOS logic will be done section 5.5.

5.1. High-Speed Digital Logic Gates

Usually, in high-speed digital applications, CML logic gates are implemented as cascoded differential pairs. The logic function is achieved by switching a constant current I_{tail} from one side of the differential pair to the other side. The currents are transformed into low and high voltage levels when they pass through the load resistors. The difference between the low and high levels (logic swing) is smaller than the supply voltage, which is one of the reasons for this topology being fast. The collection of logic gates implemented in this manner is called current-mode logic (CML) [24]. Some of the logic gates of this family are shown in Figure 5.1. An OR gate is not shown because it can be formed from an AND gate by DeMorgan's Theorem, that is, by negating the inputs and the output. Negation of signals does not require extra circuitry because all gates are fully differential. Figure 5.1 shows the logic blocks composed of both NMOS and SiGe hetero-junction bipolar transistors (HBT), hence, this logic topology will be referred to as SiGe BiCMOS CML logic.



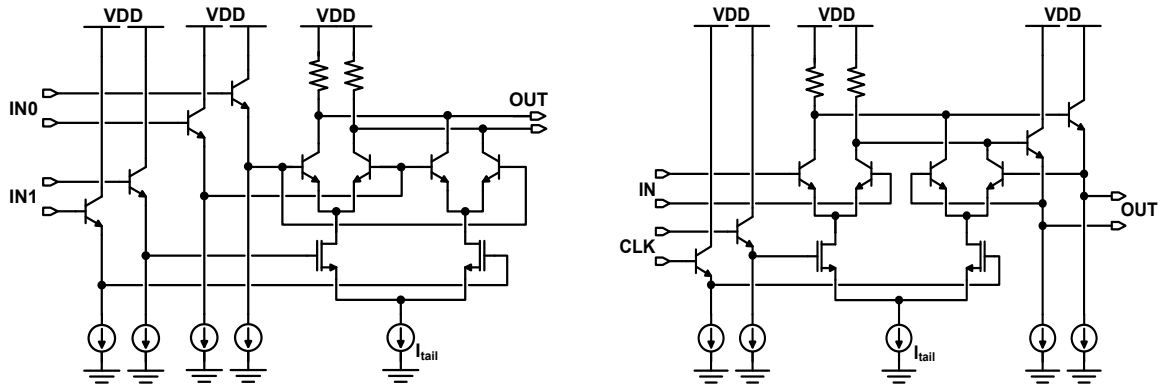
(a) Selector gate (b) XOR gate (c) AND gate (d) Latch

Figure 5.1: CML digital gates

CML gates can also be implemented with only NMOS devices, in which case they will be referred to as MOS-CML logic [25]. The original implementation of CML gates was with only bipolar devices. Nowadays bipolar devices usually require higher supply voltages than MOSFETs due to larger V_{BE}, so their use is becoming infrequent.

Another variation that is often done to CML gates is to add source- or emitter followers on the inputs and/or the outputs of the gate. When implemented with MOS transistors, this logic family is called Source-Coupled FET Logic (SCFL) [24]. In the bipolar version, it is called Emitter-Coupled Logic (ECL) [26]. ECL logic can usually operate faster, because the added emitter/source followers decouple the node capacitances, at the expense of power consumption. Some ECL gates are shown in Figure 5.2. Similarly, gates with double emitter followers can be built, with even higher supply voltage and current consumption. It should be noted that circuits with followers may become unstable, and that source followers degrade the signal. Due to the high power consumption of ECL gates, they will not be described here further.

The performance of CML digital gates highly depends on the f_T of the transistors. The



(a) ECL XOR gate (b) ECL latch

Figure 5.2: ECL gates

frequency f_T is defined as the frequency at which the current gain (h_{21}) of the transistor is equal to 1. f_T depends on the dc bias current through the transistor, and reaches a maximum for some bias current. The bias current for which f_T is maximum, when normalized to the transistor size, is called the peak-f_T current density. For optimal performance, the transistors in a logic gate have to be biased such that they are at the peak-f_T current density. Optimal performance means that the maximum switching speed is obtained for a given current consumption.

For NMOS transistors, the peak-f_T current density, J_{pf_TMOS}, stays constant at $0.3\,mA/\mu m$ as long as the constant field scaling rules apply [27]. This implies that transistors in different technology nodes, or with different widths and lengths at the same technology node, must be biased at peak-f_T current density when the bias current I_{DS} and the transistor width W are related by $I_{DS}/W = 0.3\,mA/\mu m$. Figure 5.3 shows plots of f_T for several technology nodes and NMOS transistor sizes [6].

For HBT transistors, the peak-f_T current density (J_{pf_THBT} in $mA/\mu m^2$) varies across technology nodes but remains constant to a first order approximation for transistors with different emitter lengths in a given technology. The f_T versus transistor bias current is plotted in Figure 5.4 for $0.25\,\mu m$ SiGe HBTs and several emitter lengths [24].

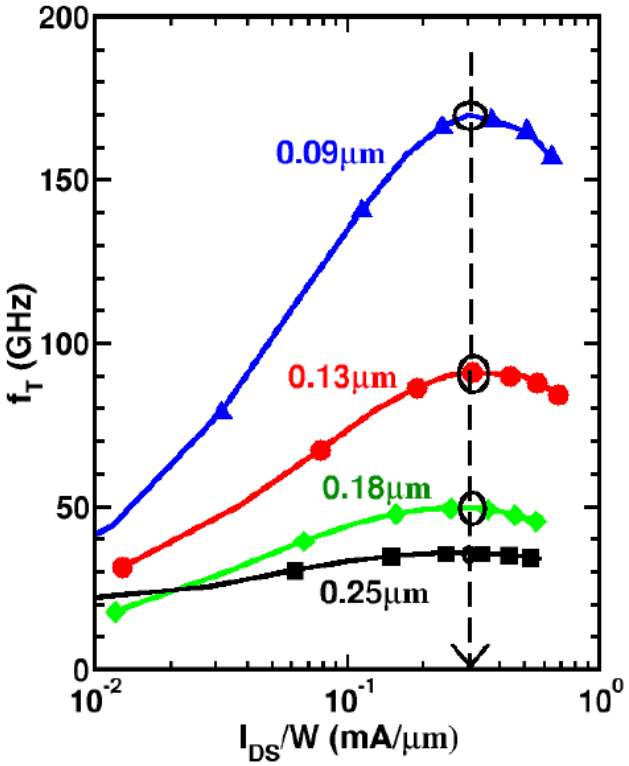


Figure 5.3: NMOS f_T as a function of current density for several technology nodes and transistor sizes

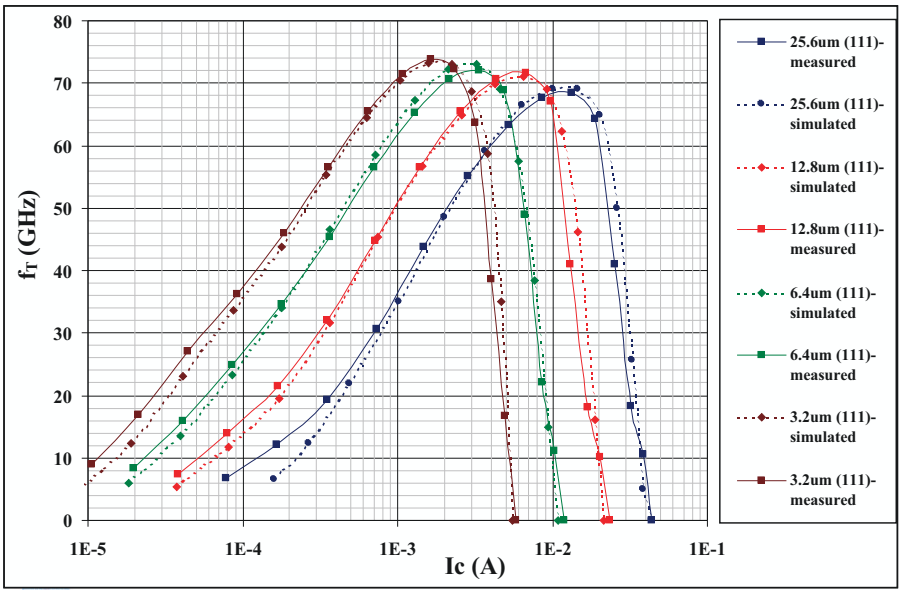


Figure 5.4: HBT f_T as a function of bias current for several transistor sizes

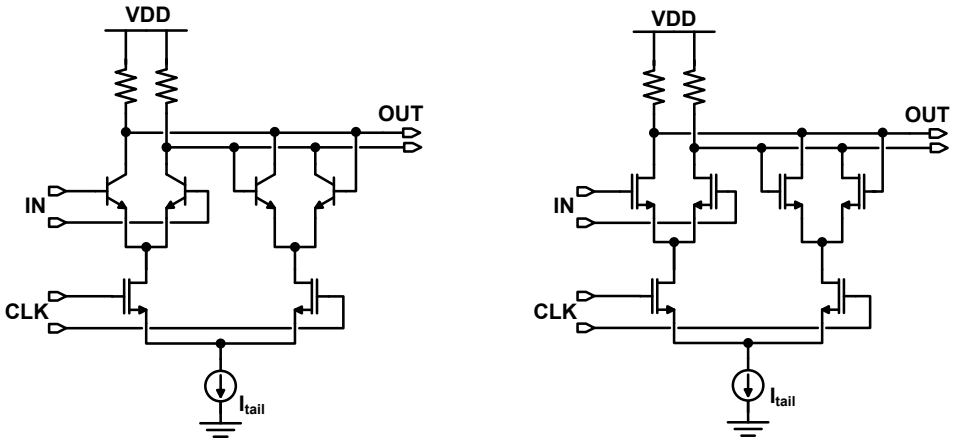
5.2. SiGe BiCMOS CML Logic / CML Latch Design

This section will present several possible BiCMOS CML latch topologies. The latch is chosen as a representative block because it contains the largest output capacitance. Furthermore, the latch operates at the full clock frequency, rather than at the data rate. Design of the latch is critical because it is needed in data re-timing and synchronization, which is essential to ensure the correct operation of high-speed logic. The design of latches shown in Figure 5.5 will be described in this section, and followed by a performance comparison.

The design of a CML logic gate starts by selecting dc voltage levels at each node. The dc levels have to be such that when the inputs and output nodes are balanced (zero differential signal) then all transistors are in saturation. Thus the V_{DS} of NMOS and V_{CE} of HBT transistors have to be approximately $0.7\,V$ and $0.9\,V$ respectively (in $0.13\,\mu m$ technology). Next, the tail current I_{tail} and load resistors R_L are chosen to produce the desired voltage swing ΔV.

$$\Delta V = I_{tail} R_L \tag{5.1}$$

ΔV is the single-ended voltage difference between logic-low and logic-high of the gate. When the inputs and output are balanced, the voltage drop across R_L is $0.5 \cdot \Delta V$. The supply voltage V_{DD} required for this gate is given by the sum of all V_{BE} or V_{GS} voltage drops in the transistor stack and the voltage drop across R_L. The power consumption of the gate is then $I_{tail} V_{DD}$.



(a) BiCMOS CML latch (b) MOS CML latch

Figure 5.5: CML latch schematics

The switching speed of the gate depends on ΔV, I_{tail}, and node capacitances:

$$switching\ time \propto \frac{\Delta V \times C/W}{I_{tail}/W} \tag{5.2}$$

Where C/W and I_{tail}/W are technology parameters. Hence, to increase the switching speed, the bias point must be chosen such that the ΔV needed to fully switch the transistors is small. Also, the transistors themselves must be small to minimize capacitance, but the current must be as large as possible. However, increasing the current density beyond the peak-f_T current density increases ΔV. For MOSFETs, the region below the peak-f_T current density bias corresponds to operation according to the square law model. In the square law region, the swing required to fully switch the transistor is given by [28]

$$\Delta V > \sqrt{2} V_{EFF} \tag{5.3}$$

However, when the device is biased at or above the peak-f_T current density, the square law no longer applies. In this case the swing required to switch the transistor is approximately [6]

$$\Delta V > 2 V_{EFF} \tag{5.4}$$

The ΔV of eq. 5.4 is larger than that of eq. 5.3 both because the coefficient is larger and V_{EFF} is larger. From this discussion it follows that, for best performance, MOSFETs have to be biased close to, but below the peak-f_T current density. Therefore, the bias current I_{tail} is chosen to be

$$I_{tail} = W_{gate} \times J_{pf_T MOS} \tag{5.5}$$

With this bias current, the MOSFETs are biased at half peak-f_T current density when the inputs are balanced, and at zero or full peak-f_T current density when the inputs are switched to one side. V_{EFF} that corresponds to half peak-f_T current density is $300\,mV$ in a $0.13\,\mu m$ technology. To account for temperature and process variations, ΔV is chosen to be $400\,mV$ to $500\,mV$.

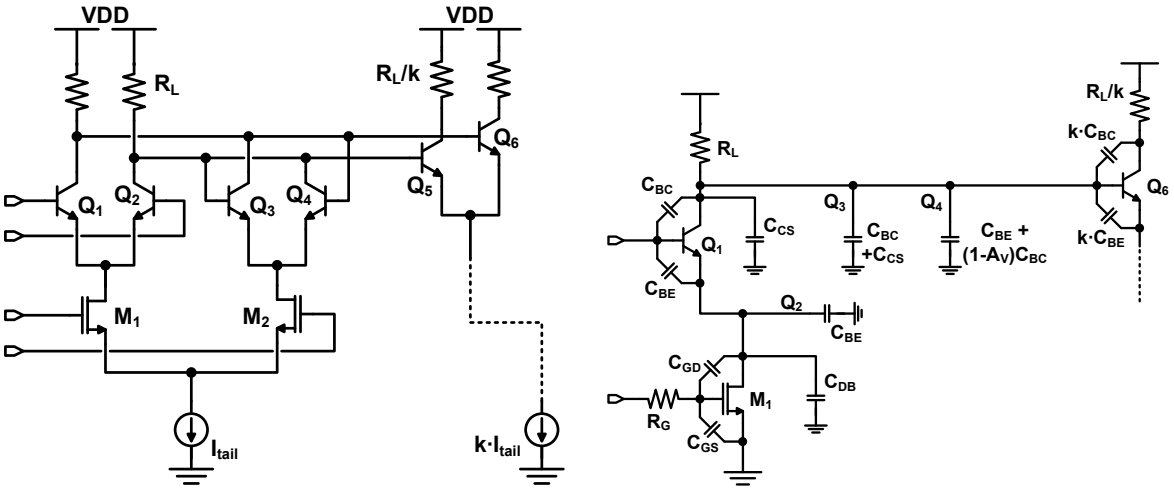
The swing needed to fully switch a bipolar transistor can be as low as four times the thermal voltage [28], but in practice needs to be $200\,mV$ to $300\,mV$ when temperature, process variations, and $R_E I_{tail}$ voltage drop are taken into account. The tail current for SiGe HBT transistors is chosen such that it corresponds to 0.75 times peak-f_T current density when the inputs are balanced, or to 1.5 times peak-f_T current density when the inputs are switched [29].

$$I_{tail} = 1.5 \times w_e \times l_e \times J_{pf_T HBT} \tag{5.6}$$

Even though in this case the peak-f_T current density is not constant across technologies, it is still constant for different-size HBTs, when the bias current is normalized to the emitter area.

In addition to the latch topologies shown in Figure 5.5, other modifications are possible for increasing speed and/or reducing power. To increase the switching speed, peaking inductors can be added to the load. Inductors reduce the capacitance term of equation 5.2 by resonating it out. For power reduction it is possible to operate the latches without the current source transistor. This allows to decrease the power supply voltage while using the same total current, thus the power consumption is reduced [19].

The performance of latches can be compared based on their time constant τ. The time constant is suitable for comparison because both the propagation delay through the latch and the rise and fall times are proportional to it. An approximation of τ for latches can be derived similarly to τ for cascode circuits [6]. τ is a sum of open-circuit-time-constants at the input and output nodes and accounts for the fanout k. Equation 5.7 gives an approximation of τ for a BiCMOS cascode latch. When deriving equation 5.7, it is assumed that the latch is loaded by a similar latch, but with a tail current of $k \times I_{tail}$ in which all transistors and their capacitances are k times larger. Also, it is assumed that the output of the latch is connected to the top (bipolar) pair and not to the bottom MOSFET pair. Figure 5.6 illustrates the relevant parasitic capacitances used to derive the latch time constant τ. The first term of equation 5.7 is the time constant (τ_{in}) at the clock input of the latch. The second term (τ_{mid}) is the time constant in the middle (cascode) node of the latch. The third term (τ_{out}) represents the charging of output capacitances by the tail current. It takes into account the Miller capacitance of the latching



(a) BiCMOS CML latch with load (b) One side of (a) with relevant capacitances

Figure 5.6: Time-constant derivation for a BiCMOS latch

pair. The fourth term (τ_{fanout}) describes the fanout of the latch.

$$\begin{aligned}\tau_{BiCMOS-Latch} \approx \tau_{in} + \tau_{mid} + \tau_{out} + \tau_{fanout} = \\ = \frac{R_G}{R_L}\frac{\Delta V}{I_{tail}}\left\{C_{GS} + \left(1 + \frac{g_{m,MOS}}{g_{m,HBT}}\right)C_{GD}\right\} \\ + \frac{2C_{BE} + C_{DB} + C_{GD}}{g_{m,HBT}} \\ + \frac{\Delta V}{I_{tail}}\left\{2C_{BC} + 2C_{CS} + C_{BE} + \left(1 + g_{m,HBT}R_L\right)C_{BC} + C_{INT}\right\} \\ + k\frac{\Delta V}{I_{tail}}\left\{C_{BE} + \left(1 + g_{m,HBT}R_L\right)C_{BC}\right\}\end{aligned} \tag{5.7}$$

A similar expression can be derived for MOS-CML latches, where all devices are NMOS transistors (Figure 5.5(b)). It is given in equation 5.8.

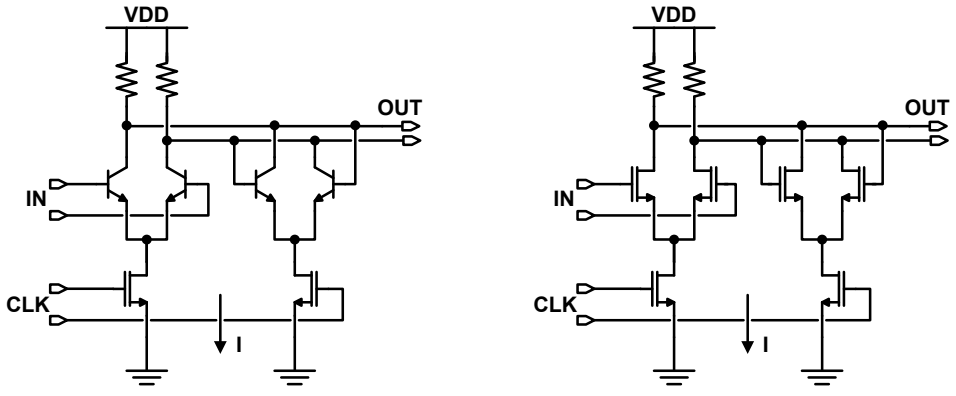
$$\begin{aligned}\tau_{MOS-Latch} \approx \tau_{in} + \tau_{mid} + \tau_{out} + \tau_{fanout} = \\ = \frac{R_G}{R_L}\frac{\Delta V}{I_{tail}}\left\{C_{GS} + 2C_{GD}\right\} \\ + \frac{2C_{GS} + 2C_{SB} + C_{DB} + C_{GD}}{g_{m,MOS}} \\ + \frac{\Delta V}{I_{tail}}\left\{2C_{GD} + 2C_{DB} + C_{GS} + \left(1 + g_{m,MOS}R_L\right)C_{GD} + C_{INT}\right\} \\ + k\frac{\Delta V}{I_{tail}}\left\{C_{GS} + \left(1 + g_{m,MOS}R_L\right)C_{GD}\right\}\end{aligned} \tag{5.8}$$

5.3. Power and Speed Optimizations of CML Latches

After presenting the basic CML latches and how they are designed, this section will describe the possible modifications that can be done to further reduce power consumption and improve the speed at which these latches operate.

The CML latches presented in the previous section require a supply voltage of $2.5\,V$. As seen in Figure 5.5, $0.4\,V$ - $0.7\,V$ are spent on the transistor that sets the tail current in the latch. This transistor can be eliminated to reduce power consumption without sacrificing performance. The new latch configuration is shown in Figure 5.7. Now, the latch can operate from $1.8\,V$, or lower, with the same current as before. The speed performance is maintained because ΔV, I_{tail}, and all capacitances are kept constant. However, precaution must be taken in the design process to ensure that the current through the latches of Figure 5.7 is the same as in the latches of Figure 5.5. The supply voltage can be further reduced with newer process technologies, in which smaller voltage drops are needed for each stacked MOSFET transistor.

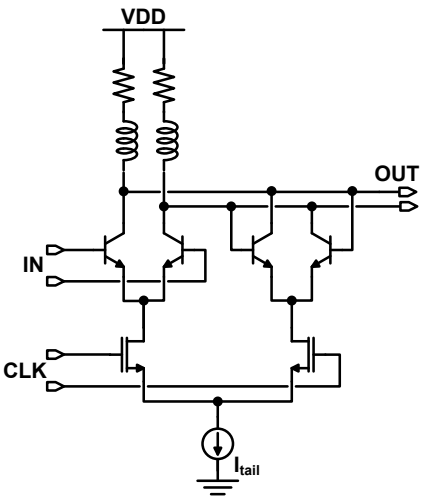
Biasing of the CML latches of Figure 5.7 proceeds similarly to the earlier case. Since there now are two separate branches that go to ground, the current in each branch is $I_{branch} = 0.5 \cdot I_{tail}$, where I_{tail} is the corresponding tail current of the latches in Figure 5.5. The MOS transistors are sized such that $W_{gate} = I_{branch}/0.5J_{pf_TMOS}$ when there is zero differential input to the latch. The SiGe HBT transistors are sized such that $w_e \times l_e = I_{branch}/0.75J_{pf_THBT}$ when there is zero differential input to the latch. This choice of I_{branch} and transistor sizes results in peak-f_T current density biasing and thus maintains the optimal switching characteristics described in section 5.2. The time constant τ for each latch of Figure 5.7 can also be calculated using equations 5.7 and 5.8.



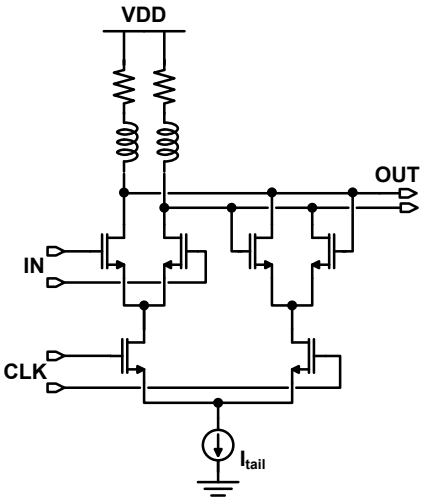
(a) BiCMOS CML latch without current source

(b) MOS CML latch without current source

Figure 5.7: CML latch schematics without current source and V_{DD} reduced to $1.8V$

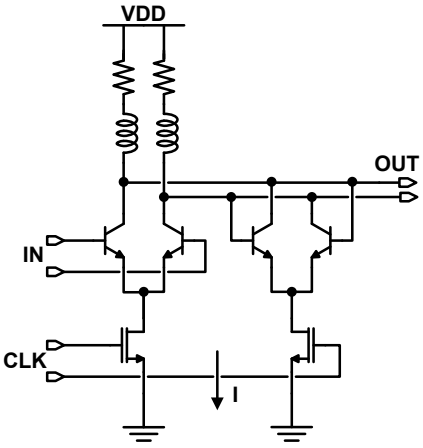


(a) BiCMOS CML latch with inductive peaking

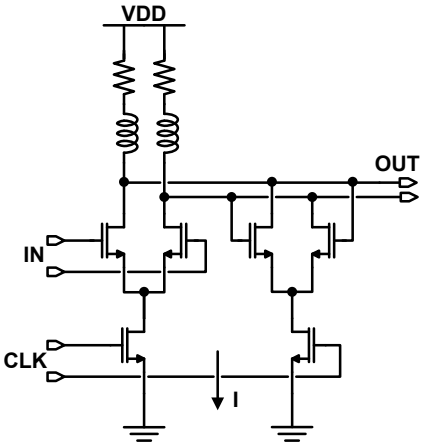


(b) MOS CML latch with inductive peaking

Figure 5.8: CML latch schematics with shunt inductive peaking



(a) BiCMOS CML latch with inductive peaking and without current source



(b) MOS CML latch with inductive peaking and without current source

Figure 5.9: CML latch schematics with shunt inductive peaking and without current source

The second modification that can be applied improves speed without requiring more power. This is achieved by adding shunt peaking inductors to the latches discussed so far. The inductors are added as part of the load and are used to extend the bandwidth of the circuit by reducing the effect of the output capacitance (which is dominant). Since the inductors are connected in series with the load resistors, they can be added to the latches of both Figures 5.5 and 5.7. The latch topologies with inductors are shown in Figures 5.8, and 5.9.

The peaking inductors do not affect the biasing of the transistors, but they reduce the time constant of the latch, and thus allow it to operate faster. For flat group delay response (i.e.

minimum deterministic jitter), the inductor value is selected according to

$$L = \frac{C_{out} R_L^2}{3.1} \tag{5.9}$$

where R_L is the load resistance and C_{out} is the total capacitance at the output node. This value of L improves the output time constant 1.6 times. For a BiCMOS CML latch, the improved τ becomes

$$\begin{aligned}\tau_{BiCMOS-Latch-Ind} \approx \tau_{in} + \tau_{mid} + \tau_{out} + \tau_{fanout} = \\ = \frac{R_G}{R_L}\frac{\Delta V}{I_{tail}}\left\{C_{GS} + \left(1 + \frac{g_{m,MOS}}{g_{m,HBT}}\right) C_{GD}\right\} \\ + \frac{2C_{BE} + C_{DB} + C_{GD}}{g_{m,HBT}} \\ + \frac{\Delta V}{I_{tail}}\frac{\{2C_{BC} + 2C_{CS} + C_{BE} + (1 + g_{m,HBT}R_L)\, C_{BC} + C_{INT}\}}{1.6} \\ + k\frac{\Delta V}{I_{tail}}\frac{\{C_{BE} + (1 + g_{m,HBT}R_L)\, C_{BC}\}}{1.6}\end{aligned} \tag{5.10}$$

And for a MOS-CML latch the improved τ is

$$\begin{aligned}\tau_{MOS-Latch-Ind} \approx \tau_{in} + \tau_{mid} + \tau_{out} + \tau_{fanout} = \\ = \frac{R_G}{R_L}\frac{\Delta V}{I_{tail}}\{C_{GS} + 2C_{GD}\} \\ + \frac{2C_{GS} + 2C_{SB} + C_{DB} + C_{GD}}{g_{m,MOS}} \\ + \frac{\Delta V}{I_{tail}}\frac{\{2C_{GD} + 2C_{DB} + C_{GS} + (1 + g_{m,MOS}R_L)\, C_{GD} + C_{INT}\}}{1.6} \\ + k\frac{\Delta V}{I_{tail}}\frac{\{C_{GS} + (1 + g_{m,MOS}R_L)\, C_{GD}\}}{1.6}\end{aligned} \tag{5.11}$$

5.4. Performance Comparison and Simulation Results

This section presents a performance comparison of all the latches described in this chapter. The comparison is carried out both with hand calculations based on technology data and with simulations of the latches under identical conditions. The calculations and the simulations are conducted for two technologies, to be able to predict the feasibility of the proposed latch topologies for future applications. The first is a production $0.13\,\mu m$ SiGe BiCMOS technology

with transistor f_T of 160 GHz [18]. The second is a $90\,nm$ SiGe BiCMOS technology under development with transistor f_T of 230 GHz [30].

To make the comparison fair, all latches were designed to operate with a total current consumption of 1 mA. A current of 1 mA was chosen because it is the current that keeps a minimum-size HBT in the $0.13\,\mu m$ SiGe BiCMOS technology biased as outlined in the previous section. Then, the maximum bit rate at which the latch operated properly was observed. Proper operation condition is reached when the output swing is equal to the designed swing.

For hand calculations, equations 5.7, 5.8, 5.10, and 5.11 were used. The model parameters of the two technologies that were used in calculations are summarized in Table 5.1. The device sizes used to realize the 1-mA latches are given in Table 5.2 for each latch configuration. Finally, Table 5.4 compares the performance of the latches based on power consumption, the calculated time constant τ, and the maximum simulated speed of operation.

Simulated eye diagrams of the various BiCMOS CML latch topologies are demonstrated in Table 5.3. Eye diagrams at the maximum bit rate where the output swing is correct are shown. The bit rate and power consumption obtained in these simulations are summarized in Table 5.4. The latches in each simulation had a fanout of $k = 1$.

The latch shown in Figure 5.5(a) was fabricated in $0.13\,\mu m$ SiGe BiCMOS technology, as part of the PRBS generator that was described in chapters 3 and 4. It was found to work

Device Parameter	$0.13\,\mu m$ NMOS	160-GHz SiGe HBT	$90\,nm$ NMOS	230-GHz SiGe HBT
g_m at peak f_T	1.1 $mS/\mu m$	38 $mS/\mu m$	1.4 $mS/\mu m$	74 $mS/\mu m$
C_{GD}, C_{BC}	0.5 $fF/\mu m$	11 $fF/\mu m^2$	0.4 $fF/\mu m$	8 $fF/\mu m^2$
C_{GS}, C_{BE}	1.25 $fF/\mu m$	2.75 $fF/\mu m$	1.0 $fF/\mu m$	2.0 $fF/\mu m$
C_{DB}, C_{CS}	1.7 $fF/\mu m$	1.1 $fF/\mu m$	1.1 $fF/\mu m$	0.8 $fF/\mu m$
J_{pf_TMOS}, J_{pf_THBT}	0.3 $mA/\mu m$	6 $mA/\mu m^2$	0.3 $mA/\mu m$	13 $mA/\mu m^2$
w_e	-	0.2 μm	-	0.15 μm

Table 5.1: Model parameters for $0.13\,\mu m$ and $90\,nm$ SiGe BiCMOS technologies

	$0.13\mu m$ SiGe BiCMOS				$90nm$ SiGe BiCMOS			
Latch	NMOS $(\frac{W}{L})$	HBT (l_e)	R_L	L	NMOS $(\frac{W}{L})$	HBT (l_e)	R_L	L
Fig. 5.5(a)	$2 \times \frac{2\,\mu m}{0.13\,\mu m}$	$0.64\,\mu m$	$400\,\Omega$	-	$2 \times \frac{2\,\mu m}{90\,nm}$	$0.50\,\mu m$	$200\,\Omega$	-
Fig. 5.7(a)	$2 \times \frac{1\,\mu m}{0.13\,\mu m}$	$0.64\,\mu m$	$400\,\Omega$	-	$2 \times \frac{1\,\mu m}{90\,nm}$	$0.50\,\mu m$	$200\,\Omega$	-
Fig. 5.8(a)	$2 \times \frac{2\,\mu m}{0.13\,\mu m}$	$0.64\,\mu m$	$400\,\Omega$	2 nH	$2 \times \frac{2\,\mu m}{90\,nm}$	$0.50\,\mu m$	$200\,\Omega$	500 pH
Fig. 5.9(a)	$2 \times \frac{1\,\mu m}{0.13\,\mu m}$	$0.64\,\mu m$	$400\,\Omega$	2 nH	$2 \times \frac{1\,\mu m}{90\,nm}$	$0.50\,\mu m$	$200\,\Omega$	500 pH

Table 5.2: Device sizes for 1-mA latches in each configuration

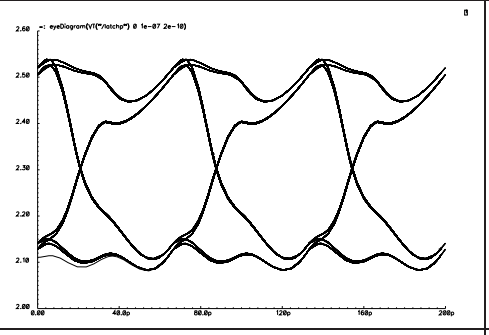
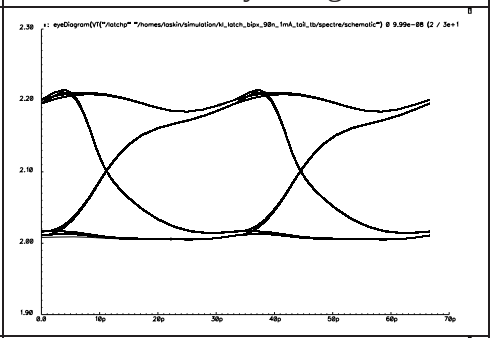
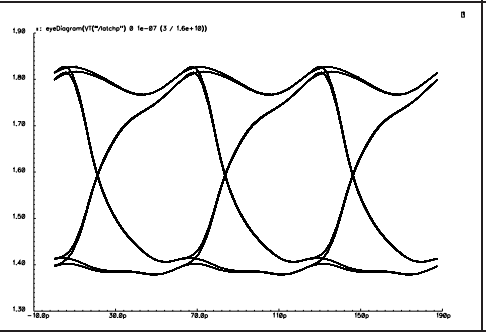
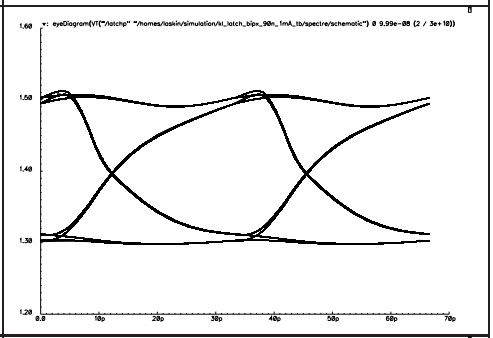
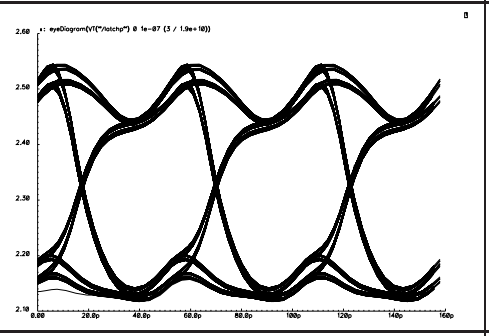
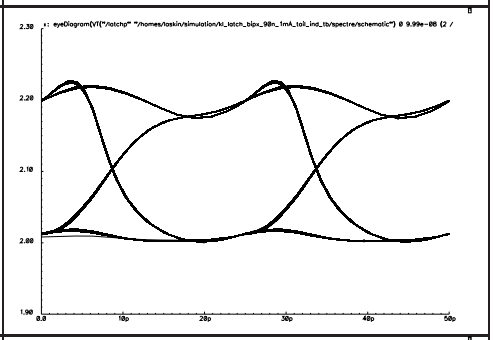
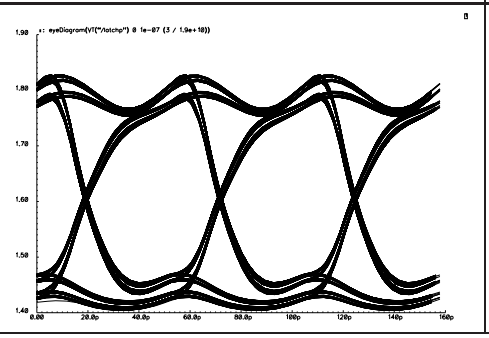
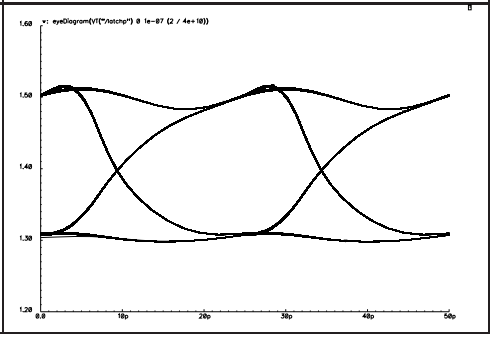
	$0.13\mu m$ SiGe BiCMOS	$90nm$ SiGe BiCMOS
Latch	Simulated Eye Diagrams	Simulated Eye Diagrams
Fig. 5.5(a)		
Fig. 5.7(a)		
Fig. 5.8(a)		
Fig. 5.9(a)		

Table 5.3: Simulated eye diagrams for 1-mA latches with different topologies. The corresponding bit rates are summarized in Table 5.4

	$0.13\,\mu m$ SiGe BiCMOS				$90\,nm$ SiGe BiCMOS			
Latch	τ	Power	ΔV	Bit Rate	τ	Power	ΔV	Bit Rate
Fig. 5.5(a)	14.1 ps	2.5 mW	400 mV	15 Gb/s	6.3 ps	2.2 mW	200 mV	30 Gb/s
Fig. 5.7(a)	13.9 ps	1.8 mW	400 mV	16 Gb/s	5.9 ps	1.5 mW	200 mV	30 Gb/s
Fig. 5.8(a)	9.1 ps	2.5 mW	400 mV	19 Gb/s	4.2 ps	2.2 mW	200 mV	40 Gb/s
Fig. 5.9(a)	8.9 ps	1.8 mW	400 mV	19 Gb/s	3.9 ps	1.5 mW	200 mV	40 Gb/s

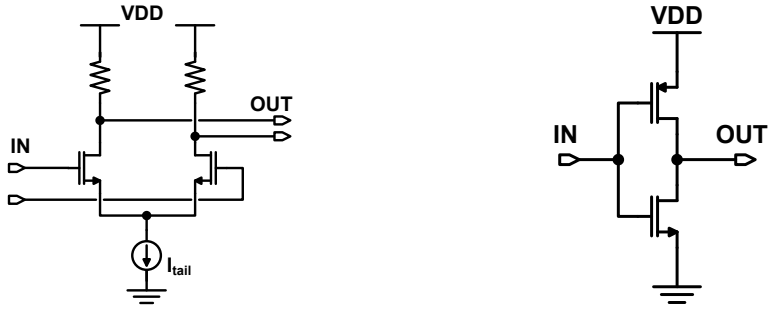
Table 5.4: Performance comparison of the different latch topologies

correctly up to 12 Gb/s. The performance of this latch after fabrication agrees closely with simulation results that include layout parasitics. This confirms the validity of the simulation results presented for the other latch topologies. The analysis presented in this chapter demonstrates that it is possible to reduce the supply voltage without increasing the tail current, thus saving power. Furthermore, it is possible to significantly increase the speed of a latch, with the sacrifice of some area, by adding peaking inductors.

5.5. Future Scaling and CMOS Logic

To further investigate the bit rates that will be possible with future technology scaling, it is useful to look at how CML logic compares with CMOS logic. This section will compare MOS-CML and CMOS ring oscillators built from inverters in a $65\,nm$ CMOS technology. Their oscillation speed is indicative of the logic speed attainable in this technology.

MOS-CML and conventional CMOS inverters are shown in Figure 5.10. For these circuits too, the time constants can be derived. They are given in equations 5.12 and 5.13. In the derivation of these equations, it is assumed that a unit-size inverter has a fanout of k. For the MOS-CML inverter the relation $R_L = \Delta V / I_{tail}$ is used. For the CMOS inverter, it is assumed



(a) MOS-CML inverter (b) CMOS inverter

Figure 5.10: MOS-CML and CMOS inverter schematics

that $W_p = 2W_n$, so that the NMOS and PMOS devices have the same g_m.

$$\tau_{MOS-CML-INV} \approx \frac{\Delta V}{I_{tail}} (C_{GD} + C_{DB}) + k\frac{\Delta V}{I_{tail}} \left(1 + \frac{R_G}{R_L}\right) [C_{GS} + (1 + g_m R_L) C_{GD}] \tag{5.12}$$

$$\tau_{CMOS-INV} \approx \frac{3}{2} r_o (C_{GD} + C_{DB}) + \frac{3}{2} k r_o \left(1 + \frac{R_G}{r_o}\right) [C_{GS} + (1 + g_m r_o) C_{GD}] \tag{5.13}$$

These equations have almost the same form if $r_o = R_L$. Since, usually, r_o is larger than R_L, pure CMOS inverters are slower (have larger τ) than CML inverters. This statement also applies to the speeds of general CMOS gates and MOS-CML / BiCMOS-CML gates, because the inverters are basic units of the logic gates.

The average dynamic power consumption of a CMOS inverter can be approximated by [31]

$$P_{dyn-avg} = C_L V_{DD}^2 f_{osc} \approx V_{DD} I_{peak} \left(\frac{t_{rise} + t_{fall}}{2}\right) f_{osc} \tag{5.14}$$

If the maximum operating frequency $f_{osc} = 1/T$ occurs when $T = (t_{rise} + t_{fall})$ then the CMOS power consumption is approximated by

$$P_{dyn-avg} \approx \frac{V_{DD} I_{peak}}{2} \tag{5.15}$$

Where I_{peak} is directly proportional to the transistor width. This figure is much smaller than the power consumption of MOS-CML inverters, latches, and other gates. However, the achievable speed in CMOS logic is also much smaller than that of MOS-CML circuits. As shown in [32], with the benefit of scaling and using the design methodology presented earlier, 65-nm latches can operate at bit rates up to 60 Gb/s.

5.5.1. CMOS and MOS-CML Ring Oscillators

The PRBS generator and checker chip was based entirely on BiCMOS CML logic. This logic topology was chosen to achieve the required speed, and designed to minimize power. As part of this thesis I have looked at the speed and power of other logic topologies. To verify the predictions that are made regarding the power and speed performance of CML and CMOS logic, test chips were designed with both logic topologies. The test chips were designed in TSMC's 65-nm CMOS process technology. They include CMOS and MOS-CML ring oscillator circuits that can be easily used to evaluate the maximum logic speed that can be achieved.

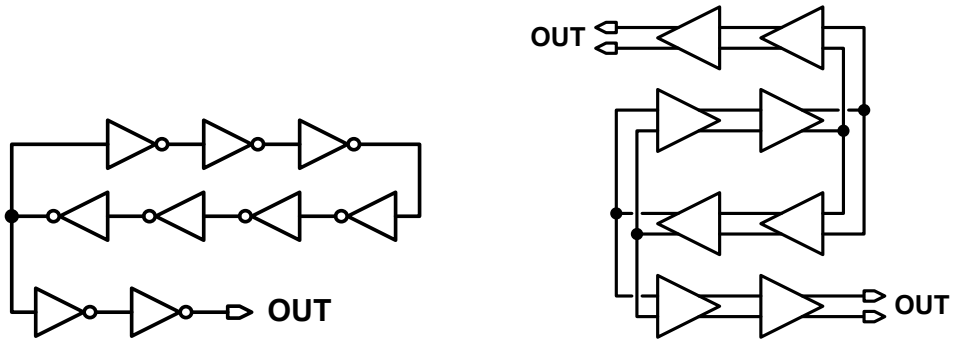
In total, 5 different ring oscillator circuits were designed. Their schematics, simulation results, and layouts are summarized in Table 5.5. For all 5 circuits $VDD = 1V$ was used.

The inverters in the 7-stage CMOS ring oscillators were designed with $W_p = 2 \times W_n$, which reflects the measured I_{on} ratio of 90-nm PMOS and NMOS transistors. Two different inverter sizes were used in the two CMOS ring oscillators to explore the effect of transistor width on speed and power. One with $\frac{W_n}{L} = \frac{0.5\,\mu m}{65\,nm}$ (Figure 5.12(a)) and the other with $\frac{W_n}{L} = \frac{1\,\mu m}{65\,nm}$ (Figure 5.12(b)).

Also, two different MOS-CML 7-stage ring oscillators were designed. Both oscillators consist of inverters with a tail current of 1 mA, with transistors sized for peak-f_T current density (Figure 5.12(c)). One of the designs includes 700-pH peaking inductors (Figure 5.12(d)). Notice that, because the quality factor of the inductors is not important in this case, very high inductance per unit area can be achieved by using narrow metal width and spacing. Thus the 700-pH, 3-turn, 3-layer stacked inductor occupies an area of only $10\,\mu m \times 10\,\mu m$.

Finally, a 60-GHz quadrature oscillator, capable of generating 8 different output phases (because it is differential), was created. It is built from the MOS-CML inverters with inductive peaking (Figure 5.12(d)). The layouts of all the designed 65-nm circuits are shown in Figure 5.13.

These 65-nm CMOS circuits did not come back from fabrication yet.

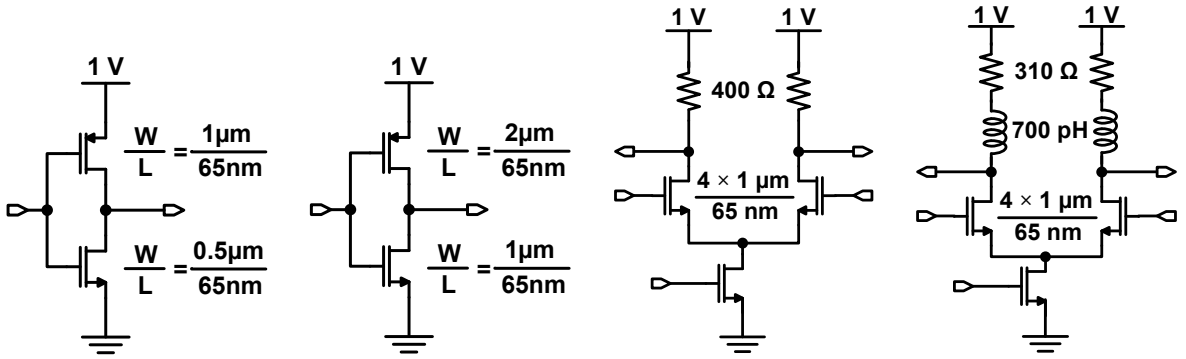


(a) 7-Stage ring oscillator (b) Quadrature ring oscillator

Figure 5.11: Top level schematics of the designed ring oscillators and quadrature oscillator

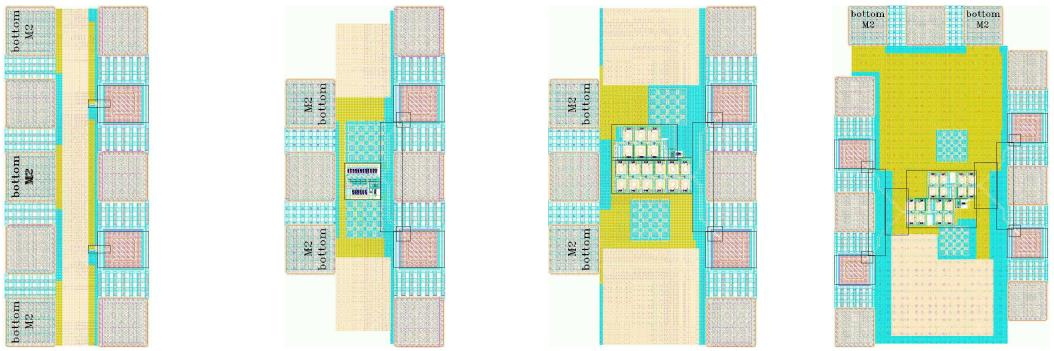
	Oscillator Schematic	Inverter Schematic	Simulated Oscillation Frequency	Simulated Inverter Power	Layout Size
1	Fig. 5.11(a)	Fig. 5.12(a)	$f_{osc} = 8.7\,GHz$	$P_{inv} = 37\,\mu W$	$8.8\,\mu m \times 4.4\,\mu m$
2	Fig. 5.11(a)	Fig. 5.12(b)	$f_{osc} = 8.8\,GHz$	$P_{inv} = 65\,\mu W$	$8.8\,\mu m \times 5.4\,\mu m$
3	Fig. 5.11(a)	Fig. 5.12(c)	$f_{osc} = 28\,GHz$	$P_{inv} = 1\,mW$	$48\,\mu m \times 44\,\mu m$
4	Fig. 5.11(a)	Fig. 5.12(d)	$f_{osc} = 51\,GHz$	$P_{inv} = 1\,mW$	$108\,\mu m \times 93\,\mu m$
5	Fig. 5.11(b)	Fig. 5.12(d)	$f_{osc} = 66\,GHz$	$P_{inv} = 1\,mW$	$120\,\mu m \times 92\,\mu m$

Table 5.5: Summary of the designed 65-nm oscillators



(a) CMOS inverter (b) CMOS inverter (c) CML inverter (d) CML inverter with inductors

Figure 5.12: Schematics of blocks used in the ring oscillators



(a) 2 CMOS Ring Oscillators (b) CML Ring Oscillator (c) CML Ring Oscillator with Inductors (d) CML Quadrature Oscillator

Figure 5.13: Layouts of the Ring Oscillators

Several observations can be made from the simulation results of Table 5.5. First, CMOS ring oscillators produce an output signal with a smaller frequency, but also consume a lot less power and occupy a very small area. Conversely, CML ring oscillators operate much faster with a higher power consumption. By comparing rows 3 and 4 of Table 5.5, it can be seen that inductive peaking significantly improves the operation speed with the expense of area.

5.6. Summary

This chapter has introduced two transistor-level design techniques for improving the performance of high-speed digital gates. First, the power consumption of the basic CML gates can be reduces significantly by removing the current source transistor and halving the sizes of the other transistors. Second, the operation speed of these gates can be increased while maintain-

ing the same power consumption by introducing inductive peaking; this increases the circuit area. In addition, this chapter has compared the speed, power and area characteristics of CML and CMOS inverters in 65-nm CMOS technology.

6 Conclusion

6.1. Summary

In this thesis, the development process of a self-test IP block for high-speed applications was described. Self-test blocks consist of PRBS generators and error checkers. To be able to place these self-test blocks on the same chip as the circuit to be tested, the blocks have to be small in size and consume low power. Furthermore, they must operate at a higher, or comparable, speed to the circuit being tested.

To achieve the above goals, the design was optimized both at the system level, and at the transistor level. At the system level an extensive comparison was performed between series and parallel PRBS generator architectures. It was found that, for operation above 80 Gb/s, the parallel generator architecture is much more suitable. This is due to the constant and low fanout of the flip-flops in the parallel generator, and due to the readily available re-timed outputs. Another very important property of parallel PRBS generators is that all outputs are appropriately delayed one with respect to the other such that direct multiplexing is possible. When multiplexing is used, the core generator can operate at a fraction of the final bit rate, thus saving power.

At the transistor level, a procedure for low-power latch design was described. This procedure uses the CML latch topology to avoid spending extra current in emitter or source followers. The CML latch is based on the BiCMOS cascode configuration [6]. With zero differential input to the latch, the MOS transistors are sized to be biased at half peak-f_T current density. The HBT devices are sized at 1.5 peak-f_T current density when fully switched. Using this procedure, a latch was designed that operated at 12 Gb/s while consuming only 2.5 mW of power. To the best of my knowledge, this is the lowest power latch operating above 10 Gb/s.

Several improvements to the basic CML latch topology have been presented. It is possible to further reduce the power used by the latch without sacrificing speed by removing the tail

current transistor, thus lowering the supply voltage and still drawing the same current. Also, it is possible to increase the speed of the latch, while consuming the same power as before, by adding peaking inductors.

A test chip that employs the above concepts was designed and fabricated in STMicroelectronics' $0.13\,\mu m$ SiGe BiCMOS technology. The chip included a 2^7-1 PRBS generator, PRBS error checker, and a 5-bit error counter. The PRBS generator worked correctly up to 23 Gb/s. It had four parallel, appropriately shifted, outputs, which can be directly multiplexed to 92 Gb/s. This makes the generator suitable for testing 80 Gb/s circuits. The PRBS generator consumes 235 mW, resulting in only 60 mW per 23-Gb/s output lane.

6.2. Future Work

The work presented in this thesis can be continued in several directions. These include further circuit development together with an investigation of power reduction in high-speed digital circuits.

For example, the $2^7 - 1$ PRBS generator presented in this thesis was designed to be used together with a 4:1 multiplexer, to produce a final pseudo-random sequence at a bit rate higher than 80 Gb/s. The multiplexer was not part of the chip described previously. Using the parallel generator approach combined with multiplexing, generators with sequence lengths longer than $2^7 - 1$ can be designed at very high bit rates, without consuming excessive power.

The new latch topologies presented in Chapter 5 appear to have very promising performance according to simulations. Further investigation of these topologies and fabrication of test chips that utilize these latches is required. This will enable the development of low power broadband systems operating above 80 Gb/s and open the way for the next generation 100 Gb/s Ethernet [33].

Bibliography

[1] A. Koyama, T. Harada, H. Yamashita, R. Takeyari, N. Shiramizu, K. Ishikawa, M. Ito, S. Suzuki, T. Yamashita, S. Yabuki, H. Ando, T. Aida, K. Watanabe, K. Ohhata, S. Takeuchi, H. Chiba, A. Ito, H. Yoshioka, A. Kubota, T. Takahashi, and H. Nii, "43Gb/s Full-Rate-Clock 16:1 Multiplexer and 1:16 Demultiplexer with SFI-5 Interface in SiGe BiCMOS Technology," in *2003 IEEE ISSCC Digest of Technical Papers*, San Francisco, CA, Feb. 2003, pp. 232–233.

[2] M. Meghelli, "A 108Gb/s 4:1 Multiplexer in 0.13μm SiGe-Bipolar Technology," in *2004 IEEE ISSCC Digest of Technical Papers*, San Francisco, CA, Feb. 2004, pp. 236–237.

[3] A. Rylyakov and T. Zwick, "96 GHz static frequency divider in SiGe bipolar technology," in *IEEE GaAs IC Symposium Technical Digest*, San Diego, CA, Feb. 2003, pp. 288–290.

[4] T. O. Dickson, E. Laskin, I. Khalid, R. Beerkens, J. Xie, B. Karajica, and S. P. Voinigescu, "A 72 Gb/s $2^{31} - 1$ PRBS generator in SiGe BiCMOS technology," in *2005 IEEE ISSCC Digest of Technical Papers*, San Francisco, CA, Feb. 2005, pp. 342–345.

[5] ——, "An 80-Gb/s $2^{31} - 1$ Pseudorandom Binary Sequence Generator in SiGe BiCMOS Technology," *IEEE Journal of Solid State Circuits*, vol. 40, pp. 2735 – 2745, Dec. 2005.

[6] T. O. Dickson, R. Beerkens, and S. P. Voinigescu, "A 2.5-V 45-Gb/s Decision Circuit Using SiGe BiCMOS Logic," *IEEE Journal of Solid State Circuits*, vol. 40, pp. 994–1003, Apr. 2005.

[7] S. W. Golomb, *Shift Register Sequences*. San Francisco, CA: Holden-Day, Inc., 1967.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, The Art of Scientific Computing, Second Edition*. New York, NY: Cambridge University Press, 1992.

[9] *Error Performance Measuring Equipment Operating at the Primary Rate and Above*, International Telecommunications Union CCITT Series O Recommendation, O.151, Rev. 1, 1992.

[10] E. W. Weisstein, "Totient Function," *From MathWorld–A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/TotientFunction.html

[11] F. Sinnesbichler, A. Ebberg, A. Felder, and R. Weigel, "Generation of High-Speed Pseudorandom Sequences Using Multiplex Techniques," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 2738–2742, Dec. 1996.

[12] W.-Z. Chen and G.-S. Huang, "A Parallel Multi-pattern PRBS Generator and BER Tester for 40+ Gbps Serdes Applications," in *Proceedings of the 2004 IEEE Asia-Pacific Conference on Advanced Systen Integrated Circuits*, Aug. 2004, pp. 318–321.

[13] A. N. Van-Luyn, "Shift Register Connections for Delayed Versions of m-Sequences," *Electronics Letters*, vol. 14, pp. 713–715, Oct. 1978.

[14] J. J. O'Reilly, "Series-Parallel Generation of m-Sequences," *The Radio and Electronic Engineer*, vol. 45, pp. 171–176, Apr. 1975.

[15] S. Kim, M. Kapur, M. Meghelli, A. Pylyakov, Y. Kwark, and D. Friedman, "45-Gb/s SiGe BiCMOS PRBS Generator and PRBS Checker," in *Proc. IEEE Custom Integrated Circuits Conference (CICC'2003)*, 2003, pp. 313–316.

[16] R. Westcott, "Testing Digital Data Transmission Systems," United Kingdom Patent 1 281 390, 12, 1972.

[17] B. Razavi, *Design of Integrated Circuits for Optical Communications*. New York, NY: McGraw-Hill, 2003.

[18] M. Laurens, B. Martinet, O. Kermarrec, Y. Campidelli, F. Deleglise, D. Dutarte, G. Troillard, D. Gloria, J. Bonnouvrier, R. Beerkens, V. Rousset, F. Leverd, A. Chantre, and A. Monroy, "A 150GHz f_T/f_{max} 0.13μm SiGe:C BiCMOS technology," in *Bipolar/BiCMOS Circuits and Technology Meeting, 2003. Proceedings of the 2003 Meeting*, Toulouse, 2003, pp. 199–202.

[19] E. Laskin and S. P. Voinigescu, "A 60 mW per Lane, $4 \times$ 23-Gb/s 2^7-1 PRBS Generator," in *2005 IEEE Compound Semiconductor Integrated Circuit Symposium*, Palm Springs, CA, Oct. 2005, pp. 192–195.

[20] Y. Amamiya, Y. Suzuki, J. Yamaraki, A. Fujihara, S. Tanaka, and H. Hida, "1.5-V Low Supply Voltage 43-Gb/s Delayed Flip-Flop Circuit," in *Gallium Arsenide Integrated Circuit (GaAs IC) Symposium, 2003. 25th Annual Technical Digest 2003. IEEE*, Nov. 2003, pp. 169–172.

[21] D. Kucharski and K. Kornegay, "A 40Gb/s 2.5V $2^7 - 1$ PRBS Generator in SiGe Using a Low-Voltage Logic Family," in *2005 IEEE ISSCC Digest of Technical Papers*, San Francisco, CA, Feb. 2005, pp. 340–341.

[22] H. Knapp, M. Wurzer, W. Perndl, K. Aufinger, T. F. Meister, and J. Bock, "100-Gb/s 2^7-1 and 54-Gb/s $2^{11} - 1$ PRBS generators in SiGe bipolar technology," in *IEEE Compound Semiconductor IC Symposium*, Monterey, CA, Oct. 2004, pp. 219–222.

[23] K. Kanda, D. Yamazaki, T. Yamamoto, M. Horinaka, J. Ogawa, H. Tamura, and H. Onodera, "40Gb/s 4:1 MUX/1:4 DEMUX in 90nm Standard CMOS," in *2005 IEEE ISSCC Digest of Technical Papers*, San Francisco, CA, Feb. 2005, pp. 152–153.

[24] S. P. Voinigescu, D. S. McPherson, F. Pera, S. Szilagyi, M. Tazlauanu, and H. Tran, "A Comparison of Silicon and III-V Technology Performance and Building Block Implementations for 10 and 40 Gb/s Optical Networking ICs," *International Journal of High Speed Electronics and Systems*, vol. 13, Mar. 2003.

[25] M. Green, "Current-controlled CMOS circuits with inductive broadbanding," U.S. Patent 06 525 571, Feb. 25, 2003.

[26] H.-M. Rein, "Si and SiGe Bipolar ICs for 10 to 40 Gb/s Optical-Fiber TDM Links," *International Journal of High Speed Electronics and Systems*, vol. 9, pp. 347–384, 1998.

[27] S. P. Voinigescu, T. Dickson, R. Beerkens, I. Khalid, and P. Westergaard, "A Comparison of Si CMOS, SiGe BiCMOS, and InP HBTs Technologies for High-Speed and Millimeter-wave ICs," in *Si Monolithic Integrated Circuits in RF Systems*, Atlanta, GA, 2004, pp. 111–114.

[28] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 5th ed. New York, NY: Oxford Press.

[29] T. O. Dickson, M.-A. LaCroix, S. Boret, D. Gloria, R. Beerkens, and S. P. Voinigescu, "30-100-GHz Inductors and Transformers for Millimeter-Wave (Bi)CMOS Integrated Circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, pp. 123–133, Jan. 2005.

[30] P. Chevalier, C. Fellous, L. Rubaldo, D. Dutartre, M. L. T. Jagueneau, F. Leverd, S. Bord, C. Richard, D. Lenoble, J. Bonnouvrier, M. Marty, A. Perrotin, D. Gloria, F. Saguin, B. Barbalat, R. Beerkens, N. Zerounian, F. Aniel, and A. Chantre, "230 GHz self-aligned SiGeC HBT for 90 nm BiCMOS technology," in *Bipolar/BiCMOS Circuits and Technology, 2004. Proceedings of the 2004 Meeting*, 2004, pp. 225–228.

[31] K. Martin, *Digital Integrated Circuit Design*. New York, NY: Oxford University Press, Inc., 2000.

[32] S. P. Voinigescu, T. O. Dickson, T. Chalvatzis, A. Hazneci, E. Laskin, R. Beerkens, and I. Khalid, "Algorithmic design methodologies and design porting of wireline transceiver ic building blocks between technology nodes," in *IEEE Custom Integrated Circuits Conference*, 2005.

[33] M. Duelk, "Next Generation 100G Ethernet," in *31 European Conference on Optical Communication*, Glasgow, Scotland, 2005.

A Appendix

A.1. MATLAB Code

```
function mux_delays(m, phi, fname)
% mux_delays
% uses the function srconns() to find all possible shift register
% tap combinations to generate m equally-spaced phases of a PRBS
% that can be used as inputs to a m-to-(m/2) mux.
% it also writes the results to file.
% m   - the number of equally spaced phase shifts to find
% phi - an array that contains the numbers of the taps of the
%       shift register that generate the feedback polynomial,
%       e.g., for 2^7-1 PRBS: phi = [0, 6, 7].
% fname - name of the file where to write the results
%

% find the total number of possible delays
delays = 2^max(phi);

fid = fopen(fname, 'w');

% loop over the possible combinations
for d = 0:(delays/m)
    disp(sprintf('combination # %d', d));
    fprintf(fid, 'combination # %d\n', d);
    % find the m delays for that combination
    for i = 0:(delays/m):delays-1
        [b, Rd] = srconns(phi, d+i);
        disp(sprintf('\t%d delay (%d):\t%s', i, d+i, char(Rd)));
        fprintf(fid, '\t%d delay (%d):\t%s\n', i, d+i, char(Rd));
```

```
    end
end

fclose(fid);

function [b, Rd] = srconns(phi, d)
% implementation of the algorithm described in
% A. N. Van Luyn, "Shift Register Connections for
% Delayed Versions of m-Sequences", Electronics
% Letters, Vol 14, Oct 1978.
%
% phi - an array that contains the numbers of the switches
%       to generate the feedback polynomial, e.g.,
%       for 2^7-1 PRBS: phi = [0, 6, 7].
% d   - the required phase shift to be generated, e.g.,
%       for 2^7-1 PRBS, d is a number between 1 and 127.
% b   - an array that indicates the shift register connections
%       required to produce a sequence with the delay d.
% Rd  - returns the same information as b, but in polunomial
%       form

% form the feedback polynomial:
x = sym('x');
p = sum(x.^phi);

% find s - the highest power of 2 in the binary
%          representation of d
a = fliplr(dec2bin(d));
s = length(a);

% initialization:
i = s;
Rd = x^0;

% execution
while i > 0
    Rd = x^a(i) * Rd^2;
    Rd = expand(Rd);
    Rd = poly2sym(mod(sym2poly(Rd), 2)) % mod2 addition
    [Q, R] = deconv(sym2poly(Rd), sym2poly(p));
    Rd = poly2sym(mod(R, 2))
    i = i - 1
end
```

```
% correction 1
[Q, R] = deconv(sym2poly(Rd), sym2poly(p));
Rd1 = poly2sym(mod(R, 2));
b1 = mod(sym2poly(Rd1), 2);

% correction 2
b2 = mod(sym2poly(Rd + p), 2);
Rd2 = expand(poly2sym(b2));

b = b1;
Rd = Rd1;
if sum(b1) > sum(b2)
    b = b2;
    Rd = Rd2;
end
```

A.2. Simulink Schematics

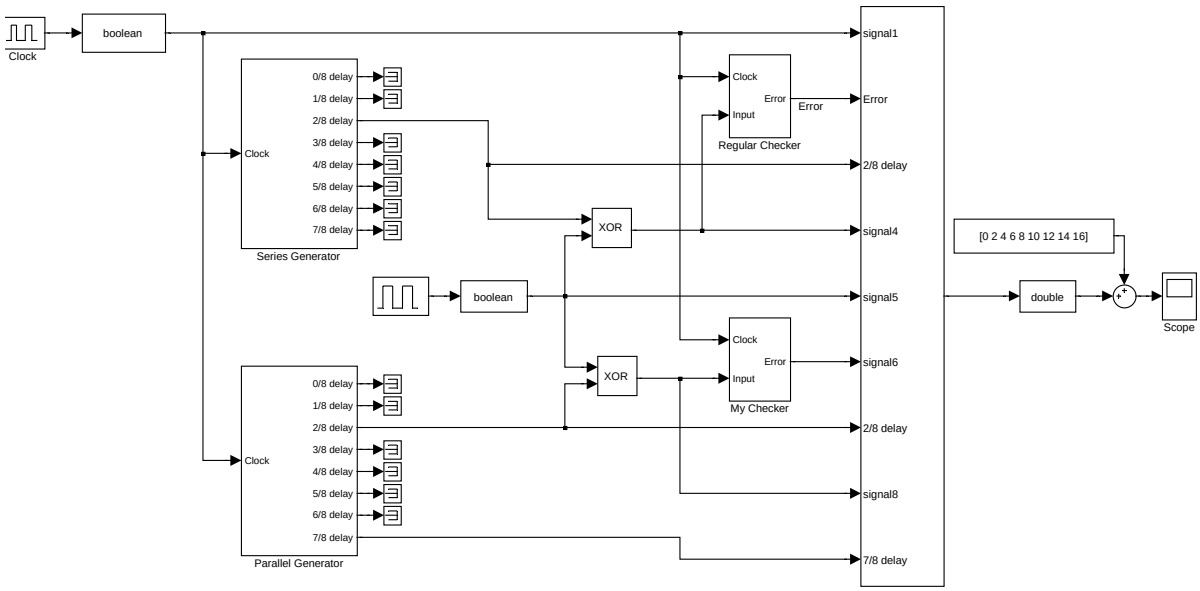


Figure A.1: Simulink test bench for the PRBS generator and checker

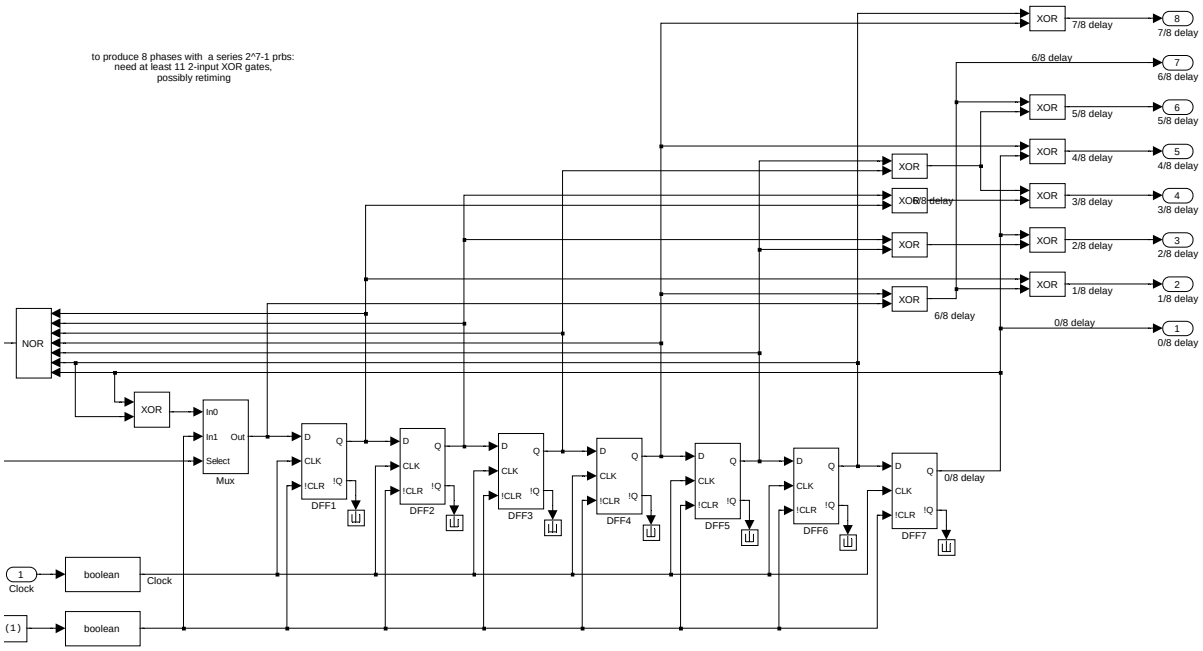


Figure A.2: A series $2^7 - 1$ PRBS generator in Simulink

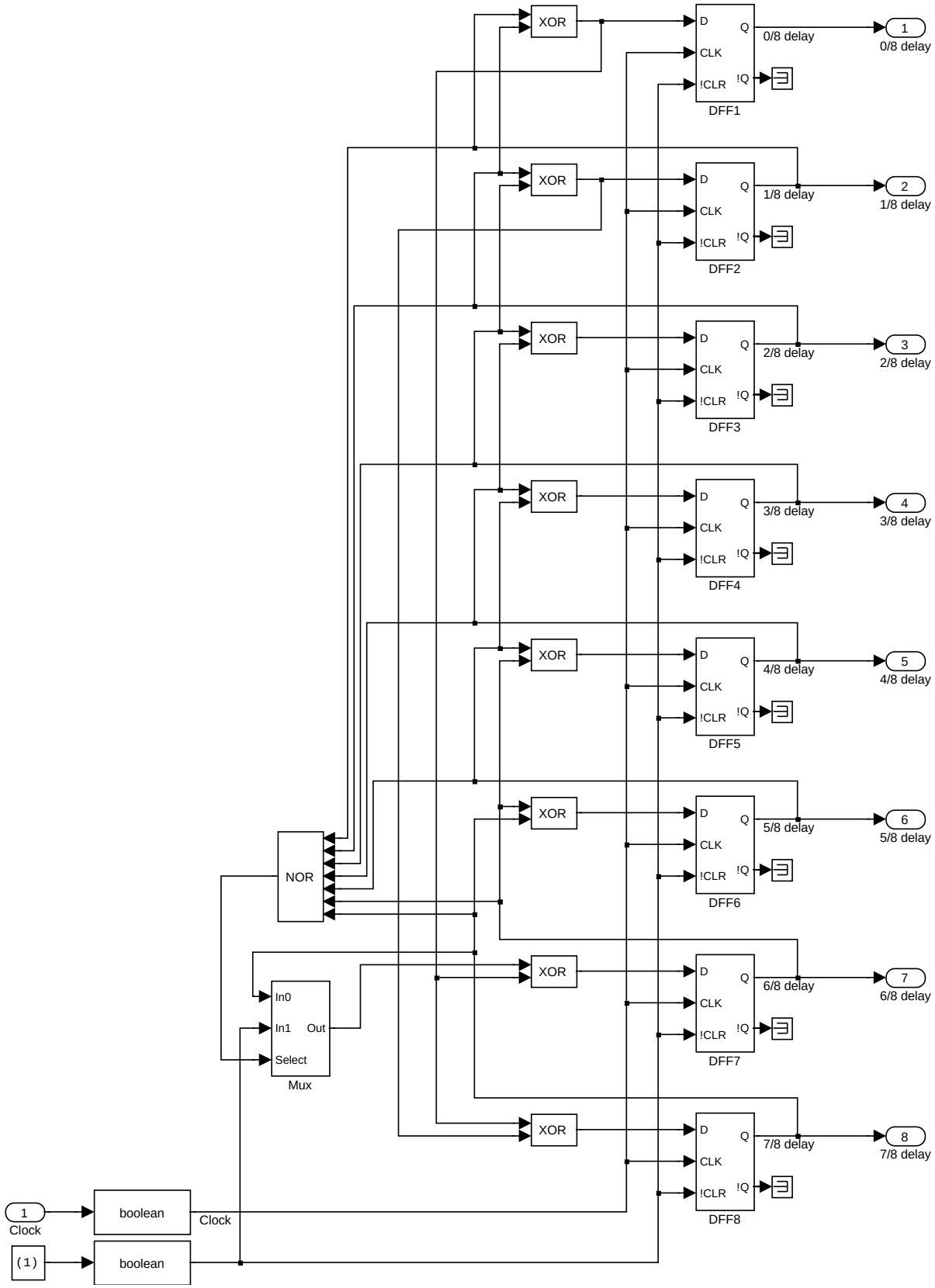


Figure A.3: A parallel $2^7 - 1$ PRBS generator in Simulink

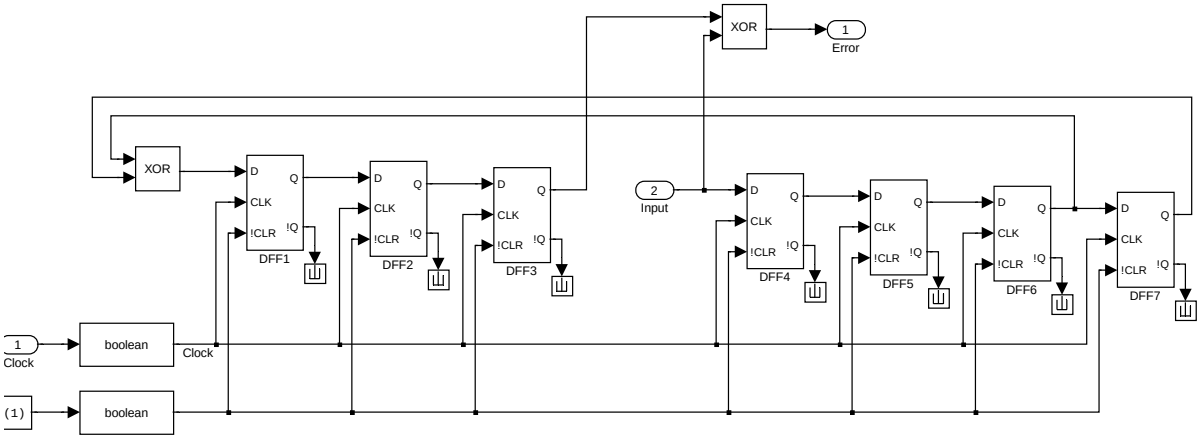


Figure A.4: A regular $2^7 - 1$ PRBS checker in Simulink

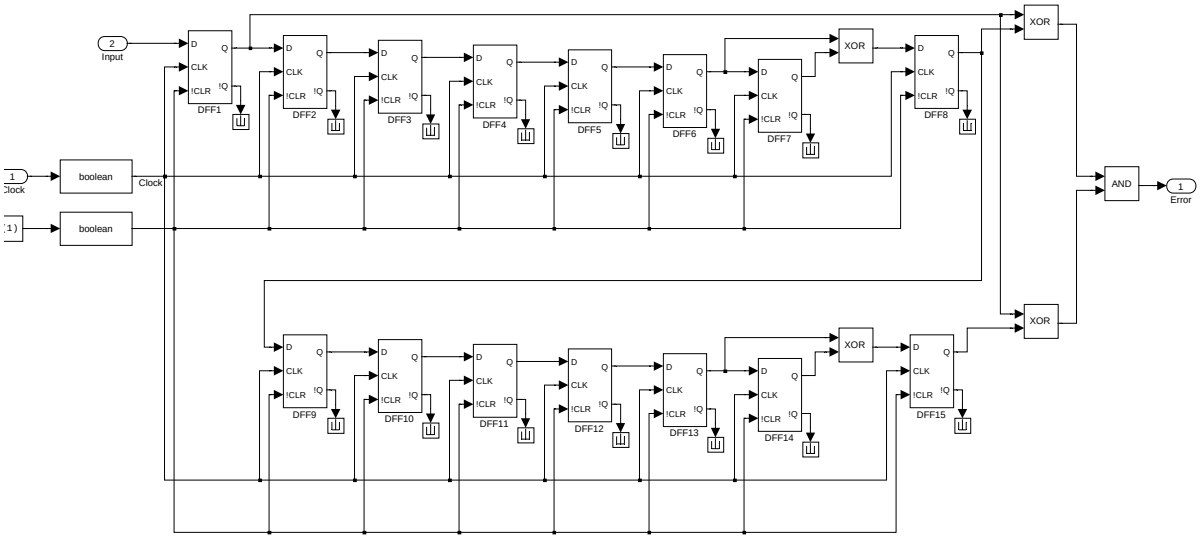


Figure A.5: The implemented $2^7 - 1$ PRBS checker in Simulink

A.3. Verilog Code

```
‘timescale 1ps/10fs

/*
parallel architecture with 8 DFFs and 8 XORs (all outputs are retimed)
All xor gates have fanout of 1, 2 of the DFFs have fanout of 3
includes gate delays
*/

//////////////////// random testing stuff ////////////////////////////

///////////////////////////// generator ////////////////////////////////

module gen;
    reg Clock, Clock2, Clock4, vdd, gnd;    // inputs to the circuit
    wire [2:0] out; // the fast output
    wire [15:0] q;  // outputs of latches
    wire [6:0] mux_out; // outputs of multiplexers
    wire [7:0] xor_out; // outputs of xor gates
    //wire [1:0] buf_out;   // outputs from the delay buffers
    wire [14:0] clk_buf;    // outputs of the clock buffers (the clock tree)
    wire [7:0] clk_buf2;
    wire [7:0] not_out; // outputs of not gates to invert the clock (no delay)
    wire       not_out2;
    wire [7:0] temp;    // inverters
    wire [9:0] retimer; // retiming latches after first stage of multiplexing

    assign out[2] = mux_out[6]; // the fastest output
    assign out[1] = mux_out[4]; // the medium output
    assign out[0] = /*mux_out[0]*/retimer[1];   // the slowest output

    // instantiation of components

    // clock tree :
    BUF_d0 cb0(clk_buf[0], Clock);
    BUF_d0 cb1(clk_buf[1], Clock);
    BUF_d0 cb2(clk_buf[2], Clock);
    BUF_d0 cb3(clk_buf[3], Clock);
    BUF_d0 cb4(clk_buf[4], Clock);
    BUF_d0 cb5(clk_buf[5], Clock);
    BUF_d0 cb6(clk_buf[6], Clock);
    BUF_d0 cb7(clk_buf[7], Clock);
```

```
/*BUF_d0 cb0(clk_buf[0], clk_buf[8]);
BUF_d0 cb1(clk_buf[1], clk_buf[8]);
BUF_d0 cb2(clk_buf[2], clk_buf[9]);
BUF_d0 cb3(clk_buf[3], clk_buf[9]);
BUF_d0 cb4(clk_buf[4], clk_buf[10]);
BUF_d0 cb5(clk_buf[5], clk_buf[10]);
BUF_d0 cb6(clk_buf[6], clk_buf[11]);
BUF_d0 cb7(clk_buf[7], clk_buf[11]);
BUF_d0 cb8(clk_buf[8], clk_buf[12]);
BUF_d0 cb9(clk_buf[9], clk_buf[12]);
BUF_d0 cb10(clk_buf[10], clk_buf[13]);
BUF_d0 cb11(clk_buf[11], clk_buf[13]);
BUF_d0 cb12(clk_buf[12], clk_buf[14]);
BUF_d0 cb13(clk_buf[13], clk_buf[14]);
BUF_d0 cb14(clk_buf[14], Clock);*/

not(not_out[0], clk_buf[0]);
not(not_out[1], clk_buf[1]);
not(not_out[2], clk_buf[2]);
not(not_out[3], clk_buf[3]);
not(not_out[4], clk_buf[4]);
not(not_out[5], clk_buf[5]);
not(not_out[6], clk_buf[6]);
not(not_out[7], clk_buf[7]);

not(not_out2, Clock2);

// the shift register :
LCH_d1 latch0(q[0], not_out[0], xor_out[0]);
LCH_d1 latch1(q[1], Clock, q[0]);
LCH_d1 latch2(q[2], not_out[1], xor_out[1]);
LCH_d2 latch3(q[3], Clock, q[2]);
LCH_d1 latch4(q[4], not_out[2], xor_out[2]);
LCH_d3 latch5(q[5], Clock, q[4]);
LCH_d1 latch6(q[6], not_out[3], xor_out[3]);
LCH_d3 latch7(q[7], Clock, q[6]);
LCH_d1 latch8(q[8], not_out[4], xor_out[4]);
LCH_d3 latch9(q[9], Clock, q[8]);
LCH_d1 latch10(q[10], not_out[5], xor_out[5]);
LCH_d3 latch11(q[11], Clock, q[10]);
LCH_d1 latch12(q[12], not_out[6], xor_out[6]);
LCH_d3 latch13(q[13], Clock, q[12]);
```

```
LCH_d1 latch14(q[14], not_out[7], xor_out[7]);
LCH_d3 latch15(q[15], Clock, q[14]);

/*LCH_d0 latch0(q[0], not_out[0], xor_out[0]);
LCH_d0 latch1(q[1], clk_buf[0], q[0]);
LCH_d0 latch2(q[2], not_out[1], xor_out[1]);
LCH_d0 latch3(q[3], clk_buf[1], q[2]);
LCH_d0 latch4(q[4], not_out[2], xor_out[2]);
LCH_d0 latch5(q[5], clk_buf[2], q[4]);
LCH_d0 latch6(q[6], not_out[3], xor_out[3]);
LCH_d0 latch7(q[7], clk_buf[3], q[6]);
LCH_d0 latch8(q[8], not_out[4], xor_out[4]);
LCH_d0 latch9(q[9], clk_buf[4], q[8]);
LCH_d0 latch10(q[10], not_out[5], xor_out[5]);
LCH_d0 latch11(q[11], clk_buf[5], q[10]);
LCH_d0 latch12(q[12], not_out[6], xor_out[6]);
LCH_d0 latch13(q[13], clk_buf[6], q[12]);
LCH_d0 latch14(q[14], not_out[7], xor_out[7]);
LCH_d0 latch15(q[15], clk_buf[7], q[14]);*/

// xor wiring :
not(temp[0], q[1]);
not(temp[1], q[3]);
not(temp[2], q[5]);
not(temp[3], q[7]);
not(temp[4], q[9]);
not(temp[5], q[11]);
not(temp[6], q[13]);
not(temp[7], q[15]);

XNOR_d3 xnor0(xor_out[0], q[3], q[5]);
XNOR_d2 xnor1(xor_out[1], q[5], q[7]);
XNOR_d1 xnor2(xor_out[2], q[7], q[9]);
XNOR_d1 xnor3(xor_out[3], q[9], q[11]);
XNOR_d1 xnor4(xor_out[4], q[11], q[13]);
XNOR_d1 xnor5(xor_out[5], q[13], q[15]);
XNOR_d1 xnor6(xor_out[6], q[15], xor_out[0]);
XNOR_d1 xnor7(xor_out[7], xor_out[0], xor_out[1]);

/*XNOR_d3 xnor0(xor_out[0], q[3], q[5]);
XNOR_d2 xnor1(xor_out[1], q[5], q[7]);
XNOR_d1 xnor2(xor_out[2], q[7], q[9]);
XNOR_d1 xnor3(xor_out[3], q[9], q[11]);
```

```
XNOR_d1 xnor4(xor_out[4], q[11], q[13]);
XNOR_d1 xnor5(xor_out[5], q[13], q[15]);
XNOR_d1 xnor6(xor_out[6], q[1], q[5]);
XNOR_d1 xnor7(xor_out[7], q[3], q[7]);*/

// mux wiring :
SEL_d1 mux0(mux_out[0], temp[0], temp[4], Clock);
SEL_d1 mux1(mux_out[1], temp[2], temp[6], Clock);
SEL_d1 mux2(mux_out[2], temp[1], temp[5], Clock);
SEL_d1 mux3(mux_out[3], temp[3], temp[7], Clock);
SEL_d0 mux4(mux_out[4], retimer[1], retimer[4], Clock2);
SEL_d0 mux5(mux_out[5], retimer[6], retimer[9], Clock2);
SEL_d0 mux6(mux_out[6], mux_out[4], mux_out[5], Clock4);

LCH_d1 latch16(retimer[0], Clock2, mux_out[0]);
LCH_d1 latch17(retimer[1], not_out2, retimer[0]);   // MS
LCH_d1 latch18(retimer[2], Clock2, mux_out[1]);
LCH_d1 latch19(retimer[3], not_out2, retimer[2]);
LCH_d1 latch20(retimer[4], Clock2, retimer[3]);     // MSM
LCH_d1 latch21(retimer[5], Clock2, mux_out[2]);
LCH_d1 latch22(retimer[6], not_out2, retimer[5]);   // MS
LCH_d1 latch23(retimer[7], Clock2, mux_out[3]);
LCH_d1 latch24(retimer[8], not_out2, retimer[7]);
LCH_d1 latch25(retimer[9], Clock2, retimer[8]);     // MSM

initial begin
    Clock = 1;
    Clock2 = 0;
    Clock4 = 0;
    vdd = 1;
    gnd = 0;
end
always #60 Clock = !Clock;  // generate the clock
always #30 Clock2 = !Clock2;    // double the speed
always #15 Clock4 = !Clock4;    // 4 times the speed
initial begin
    $shm_open("signals.shm");   // waveforms
    $shm_probe(Clock, out, q, mux_out, xor_out, temp, retimer, not_out);
    $shm_probe(clk_buf, Clock2, Clock4, clk_buf2);
end
initial begin
    $display("Time Clock in0 in1 sel m0 q1 q2 q3 q4 q5 q6 q7 slow1 slow2 fast");
    $monitor("%4g",$time,,,,Clock,,,,,xor_out[0],,,,vdd,,,,
```

```
            mux_out[0],,,q[1],,,q[2],,,q[3],,,q[4],,,q[5],,,
            q[6],,,q[7],,,,q[8],,,,,,q[3],,,,,,out);
        $dumpvars;
        #10000 $shm_close();
        $finish;
    end
endmodule

//////////////////////////// LATCH ////////////////////////////////////////

module LCH_d3(Q, clk, D);   // latch with delays (fanout=3)
    output Q;
    input clk, D;

    latch(Q, clk, D);   // instantiate a latch

    specify     // timing specifications
        specparam tPLH_clk_Q = 0;//22;  // rising Q after clk edge
        specparam tPHL_clk_Q = 0;//22;  // falling Q after clk edge
        specparam tPLH_D_Q = 0;//22;
        specparam tPHL_D_Q = 0;//22;
        (clk => Q) = (tPLH_clk_Q, tPHL_clk_Q);
        (D => Q) = (tPLH_D_Q, tPHL_D_Q);
    endspecify
endmodule

module LCH_d2(Q, clk, D);   // latch with delays (fanout=2)
    output Q;
    input clk, D;

    latch(Q, clk, D);   // instantiate a latch

    specify     // timing specifications
        specparam tPLH_clk_Q = 0;//19;  // rising Q after clk edge
        specparam tPHL_clk_Q = 0;//19;  // falling Q after clk edge
        specparam tPLH_D_Q = 0;//19;
        specparam tPHL_D_Q = 0;//19;
        (clk => Q) = (tPLH_clk_Q, tPHL_clk_Q);
        (D => Q) = (tPLH_D_Q, tPHL_D_Q);
    endspecify
endmodule

module LCH_d1(Q, clk, D);   // latch with delays (fanout=1)
```

```
    output Q;
    input clk, D;

    latch(Q, clk, D);   // instantiate a latch

    specify     // timing specifications
        specparam tPLH_clk_Q = 0;//16;  // rising Q after clk edge
        specparam tPHL_clk_Q = 0;//16;  // falling Q after clk edge
        specparam tPLH_D_Q = 0;//16;
        specparam tPHL_D_Q = 0;//16;
        (clk => Q) = (tPLH_clk_Q, tPHL_clk_Q);
        (D => Q) = (tPLH_D_Q, tPHL_D_Q);
    endspecify
endmodule

module LCH_d0(Q, clk, D);   // latch without delays
    output Q;
    input clk, D;
    latch(Q, clk, D);   // instantiate a latch
endmodule

primitive latch(Q, clk, D) ;
    output Q; reg Q;
    input clk, D;

    initial Q = 0;  // <-------- !!!

    table
        // clock data : q : q+
             0    1   : ? : 1 ;
             0    0   : ? : 0 ;
             1    ?   : ? : - ; // - = no change
    endtable
endprimitive

//////////////////////////// XNOR ////////////////////////////////////////

module XNOR_d1(c, a, b);    // XNOR gate with delays (fanout=1)
    output c;
    input a, b;

    xnor(c, a, b);
```

```
    specify
        specparam tPLH_a_c = 0;//16;
        specparam tPHL_a_c = 0;//16;
        specparam tPLH_b_c = 0;//16;
        specparam tPHL_b_c = 0;//16;
        (a => c) = (tPLH_a_c, tPHL_a_c);
        (b => c) = (tPLH_b_c, tPHL_b_c);
    endspecify
endmodule
module XNOR_d2(c, a, b);    // XNOR gate with delays (fanout=2)
    output c;
    input a, b;

    xnor(c, a, b);

    specify
        specparam tPLH_a_c = 0;//19;
        specparam tPHL_a_c = 0;//19;
        specparam tPLH_b_c = 0;//19;
        specparam tPHL_b_c = 0;//19;
        (a => c) = (tPLH_a_c, tPHL_a_c);
        (b => c) = (tPLH_b_c, tPHL_b_c);
    endspecify
endmodule
module XNOR_d3(c, a, b);    // XNOR gate with delays (fanout=3)
    output c;
    input a, b;

    xnor(c, a, b);

    specify
        specparam tPLH_a_c = 0;//22;
        specparam tPHL_a_c = 0;//22;
        specparam tPLH_b_c = 0;//22;
        specparam tPHL_b_c = 0;//22;
        (a => c) = (tPLH_a_c, tPHL_a_c);
        (b => c) = (tPLH_b_c, tPHL_b_c);
    endspecify
endmodule
module XNOR_d0(c, a, b);    // XNOR gate without delays
    output c;
    input a, b;
    xnor(c, a, b);
```

```
endmodule

////////////////////////////// SEL ////////////////////////////////////////

module SEL_d1(out, in0, in1, sel);  // SEL with delays (fanout=1)
    output out;
    input in0, in1, sel;

    MUX(out, in0, in1, sel);

    specify
        specparam tPLH_in0_out = 0;//16;
        specparam tPHL_in0_out = 0;//16;
        specparam tPLH_in1_out = 0;//16;
        specparam tPHL_in1_out = 0;//16;
        specparam tPLH_sel_out = 0;//16;
        specparam tPHL_sel_out = 0;//16;
        (in0 => out) = (tPLH_in0_out, tPHL_in0_out);
        (in1 => out) = (tPLH_in1_out, tPHL_in1_out);
        (sel => out) = (tPLH_sel_out, tPHL_sel_out);
    endspecify
endmodule

module SEL_d2(out, in0, in1, sel);  // SEL with delays (fanout=2)
    output out;
    input in0, in1, sel;

    MUX(out, in0, in1, sel);

    specify
        specparam tPLH_in0_out = 0;//8;
        specparam tPHL_in0_out = 0;//8;
        specparam tPLH_in1_out = 0;//8;
        specparam tPHL_in1_out = 0;//8;
        specparam tPLH_sel_out = 0;//8;
        specparam tPHL_sel_out = 0;//8;
        (in0 => out) = (tPLH_in0_out, tPHL_in0_out);
        (in1 => out) = (tPLH_in1_out, tPHL_in1_out);
        (sel => out) = (tPLH_sel_out, tPHL_sel_out);
    endspecify
endmodule

module SEL_d0(out, in0, in1, sel);  // SEL without delays
```

```
    output out;
    input in0, in1, sel;

    MUX(out, in0, in1, sel);

endmodule

primitive MUX(out, in0, in1, sel);
    output out; //reg out;  // <-------- !!!
    input in0, in1, sel;

    //initial out = 0;  // <-------- !!!

    table
        // in0 in1 sel : out
            1   ?   0  :  1 ; // ? = 0,1,x
            0   ?   0  :  0 ;
            ?   1   1  :  1 ;
            ?   0   1  :  0 ;
            0   0   x  :  0 ;
            1   1   x  :  1 ;
    endtable
endprimitive

////////////////////////////// BUF //////////////////////////////////////

module BUF_d1(out, in); // buffer with delay (has same delay as mux) (fanout=1)
    output out;
    input in;

    buf(out, in);

    specify
        specparam tPLH_in_out = 0;//6;
        specparam tPHL_in_out = 0;//6;
        (in => out) = (tPLH_in_out, tPHL_in_out);
    endspecify
endmodule

module BUF_d2(out, in); // buffer with delay (has same delay as mux) (fanout=2)
    output out;
    input in;
```

```
    buf(out, in);

    specify
        specparam tPLH_in_out = 0;//2.0;
        specparam tPHL_in_out = 0;//2.0;
        (in => out) = (tPLH_in_out, tPHL_in_out);
    endspecify
endmodule

module BUF_d0(out, in); // buffer without delays
    output out;
    input in;

    //assign out = in;
    buf(out, in);
endmodule
```