LFSR BASED COUNTERS

BY

AVINASH AJANE, B.E

A technical report submitted to the Graduate School
in partial fulfillment of the requirements
for the degree
Master of Science in Electrical Engineering

New Mexico State University
Las Cruces, New Mexico
September 2005
“LFSR Based Counters” a technical report prepared by Avinash Ajane, in partial fulfillment of the requirements for the degree, Master of Science in Electrical Engineering, has been approved and accepted by the following:

Paul M. Furth  
Chair of the Examining Committee

Date

Committee in charge:

Dr. Paul M. Furth, Chair

Dr. Eric E. Johnson

Dr. Jeff Beasley
ACKNOWLEDGEMENTS

I would like to thank Dr. Paul Furth, Dr. Eric Johnson and Dr. Jeff Beasley for participating on my master’s defense committee. I am deeply grateful to Dr. Paul Furth for his guidance and advice throughout my master’s. A special thanks to Dr. Eric Johnson for developing an efficient algorithm to convert random LFSR count to a known binary count.

I also thank my parents, Mr. K. V Ajane and Mrs. Kalpana Ajane, for what I am today and my brother, Ankur, for his unconditional support in my life.
VITA

September 13, ‘1978  Born in Yeotmal, Maharashtra, INDIA

May 2001  Bachelor of Engineering in Electronics and Communications, University Institute of Technology, Barkatullah University, Bhopal, INDIA.

Spring 2004  Graduate Assistant, Electronics Technology Program, Dona Ana Branch Community College, NMSU, Las Cruces.

Fall 2003–Fall 2005  Master of Science in Electrical Engineering

Field of Study

Major  Electrical Engineering

Specialization  VLSI Design
This report explores the possibility of using Linear Feedback Shift Registers (LFSRs) as an alternative to the conventional binary counters. Two counters, one using LFSRs and the other using half adders with Static Manchester carry chain, were fabricated in AMI 0.5μm process. In addition, three Algorithms, which convert the random bit patterns of the LFSR counter to known binary counts, were implemented in the ‘C’ programming language. The assumption that LFSR counter leads to reduced area and higher speed, were validated using simulation and chip measurement results. For a 16-bit binary counter operating at a 1.1V power supply, the maximum operating speed was experimentally measured to be 2.7 MHz. In
comparison a 16-bit LFSR under the same conditions operated at 6.7 MHz. The area
of the conventional counter is $0.2108 \text{mm}^2$ whereas LFSR counter consumes
only $0.1248 \text{mm}^2$.

TABLE OF CONTENTS

| LIST OF TABLES | ix |
| LIST OF FIGURES | x |
| 1 INTRODUCTION | 1 |
| 2 COUNTERS | 4 |
| 2.1 Binary Counters | 4 |
| 2.1.1 Asynchronous Binary Counters | 4 |
| 2.1.2 Synchronous Binary Counters | 6 |
| 2.1.2.1 Adders | 8 |
| 2.1.2.1.1 Full Adder | 8 |
| 2.1.2.1.2 Carry-propagate Addition | 10 |
| Ripple Carry Adder with Inverting Full Adders | 10 |
| Manchester Carry Chain Adder | 11 |
| Carry-lookahead Adders | 14 |
| Carry-skip Adder | 15 |
| Carry Select Adder | 16 |
| 2.2 Linear Feedback Shift Registers | 18 |
| 2.2.1 LFSR Implementations | 19 |
2.2.2 LFSRs as Counters ................................................................. 21

3 COUNTERS: THEIR DESIGN, ARCHITECTURE and SIMULATIONS ................................................................. 23

3.1 Basic Building Blocks ................................................................ 23

Data Flip-flop (DFF) ................................................................. 23

Half Adder ................................................................................. 27

Tri-state Buffer ................................................................. 29

1:4 Decoder .................................................................................. 29

Gated Clock ................................................................................. 32

3.2 Design Specifications ................................................................. 33

3.3 Static Manchester Carry Chain Counter ....................................... 34

3.3.1 The 4-Bit Static Manchester Carry Chain Counter ............... 35

Static Manchester Carry Chain ............................................. 35

3.3.2 The 8 Bit Static Manchester Carry Chain Counter ............... 40

3.3.3 The 16-Bit Static Manchester Carry Chain Counter ............... 43

3.3.4 The 32-Bit Static Manchester Carry Chain Counter ............... 45

3.3.5 The complete static Manchester carry chain counter chip ....... 47

3.4 The LFSR Counter ................................................................. 48

3.4.1 The 4bit LFSR .................................................................... 49

3.4.2 The 8 bit LFSR ................................................................. 50

3.4.3 The 16-bit LFSR ................................................................. 52

3.4.4 The 32-bit LFSR ................................................................. 53
3.3.5 The complete LFSR counter chip

4 THE ALGORITHMS

4.1 Algorithm I

4.2 Algorithm II

4.3 Algorithm III

5 LAYOUT, TEST SETUP AND MEASUREMENT RESULTS

5.1 Test Setup: Conventional Binary Counter

5.2 Test Setup: LFSR Counter

5.3 Comparison of Simulation Results and Test Measurements

5.4 Comparison of the Conventional Counter and the LFSR Counter

6 CONCLUSIONS, APPLICATIONS AND RECOMMENDATIONS

6.1 Conclusions

6.2 Applications

6.3 Recommendations

APPENDICES

A Algorithm implementation in ‘C’ programming language

B Layouts

C Pin Configuration and Test Procedures

D Pictures of the Circuits, Test Setup and Lab Equipments

E Output Bits at Maximum Operating Frequency and Frequency at which the Counters Fail

REFERENCES
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Truth table for full adder</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Characteristic polynomials [Wes02]</td>
<td>19</td>
</tr>
<tr>
<td>2.3 LFSR sequence</td>
<td>21</td>
</tr>
<tr>
<td>3.1 Truth table for half adder circuit generating sum bit</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Truth table 1:4 decoder</td>
<td>29</td>
</tr>
<tr>
<td>3.3 Design Specifications</td>
<td>33</td>
</tr>
<tr>
<td>3.4 Truth table for half adder circuit generating carry bit</td>
<td>36</td>
</tr>
<tr>
<td>4.1 Table 4.1 Storage array for the 4 bit LFSR</td>
<td>58</td>
</tr>
<tr>
<td>5.1 Comparison of simulation and test results for the conventional counter</td>
<td>71</td>
</tr>
<tr>
<td>5.2 Comparison of simulation and test results for the LFSR counter</td>
<td>71</td>
</tr>
<tr>
<td>5.3 Summary of simulation results: Average power dissipation</td>
<td>72</td>
</tr>
<tr>
<td>5.4 Summary of simulation results: Maximum operating frequency</td>
<td>73</td>
</tr>
<tr>
<td>5.5 Summary of the test results and layout area</td>
<td>73</td>
</tr>
<tr>
<td>C.1 Pin Configuration of the conventional binary counter</td>
<td>100</td>
</tr>
<tr>
<td>C.2 Pin Configuration of the LFSR counter</td>
<td>104</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Asynchronous 4-bit counter</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Timing diagram Asynchronous 4-bit counter</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>(a) A 4-bit synchronous ripple up/down counter</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>(b) A 4-bit synchronous ripple up counter</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Full Adder [Wes02]</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>(a) A 4-bit ripple carry adder</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>(b) A 4-bit ripple carry adder with inverting adders</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>(a) Carry chain design [Wes02]</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>(b) Static implementation of complementary carry [Wes02]</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>(c) Dynamic implementation of complementary carry [Wes02]</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>(a) Static Manchester carry chain [Wes02]</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>(b) Dynamic Manchester carry chain [Wes02]</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>Dynamic Carry-Skip adder Manchester Stage [Wes02]</td>
<td>16</td>
</tr>
<tr>
<td>2.9</td>
<td>(a) Carry Select Adder</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>(b) Multiplexer</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>(c) AND-OR Logic</td>
<td>18</td>
</tr>
<tr>
<td>2.10</td>
<td>(a) Fibonacci implementation of 4-bit LFSR</td>
<td>20</td>
</tr>
</tbody>
</table>
(b) Galois implementation of 4-bit LFSR

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Rising edge triggered static DFF with active low reset signal</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Rising edge triggered static DFF with active high set signal</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Transient response of the DFF shown in figure 3.2</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Half adder generating the sum bit s</td>
<td>28</td>
</tr>
<tr>
<td>3.5</td>
<td>A tri-state buffer</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>A 1:4 decoder</td>
<td>30</td>
</tr>
<tr>
<td>3.7</td>
<td>AND gate with graded sizing of nMOS transistor</td>
<td>31</td>
</tr>
<tr>
<td>3.8</td>
<td>Gated clock circuit</td>
<td>32</td>
</tr>
<tr>
<td>3.9</td>
<td>Block Diagram of the counter</td>
<td>34</td>
</tr>
<tr>
<td>3.10</td>
<td>Block Diagram of the 4 bit counter</td>
<td>35</td>
</tr>
<tr>
<td>3.11</td>
<td>4 bit static Manchester carry chain</td>
<td>36</td>
</tr>
<tr>
<td>3.12</td>
<td>Carry skip path</td>
<td>37</td>
</tr>
<tr>
<td>3.13</td>
<td>Schematic of the 4 bit static Manchester carry chain counter</td>
<td>38</td>
</tr>
<tr>
<td>3.14</td>
<td>Transient response of the 4 bit static Manchester carry chain counter</td>
<td>39</td>
</tr>
<tr>
<td>3.15</td>
<td>Block Diagram of the 8 bit counter</td>
<td>40</td>
</tr>
<tr>
<td>3.16</td>
<td>8 bit static Manchester carry chain with carry skip path</td>
<td>41</td>
</tr>
<tr>
<td>3.17</td>
<td>Schematic of the 8 bit static Manchester carry chain counter</td>
<td>42</td>
</tr>
<tr>
<td>3.18</td>
<td>Transient response of the 8 bit static Manchester carry chain counter</td>
<td>43</td>
</tr>
<tr>
<td>3.19</td>
<td>16-bit static Manchester carry chain with carry skip path</td>
<td>44</td>
</tr>
<tr>
<td>3.20</td>
<td>Schematic of the 16 bit static Manchester carry chain counter</td>
<td>45</td>
</tr>
<tr>
<td>3.21</td>
<td>Schematic of the 32-bit static Manchester carry chain counter</td>
<td>46</td>
</tr>
</tbody>
</table>
3.22 Schematic of the static Manchester carry chain counter chip................. 47
3.23 Schematic of the 4 bit LFSR................................................................. 49
3.24 Transient response of the 4 bit LFSR at 404MHz........................... 49
3.25 Transient response of the 4 bit LFSR at 405MHz......................... 50
3.26 Schematic of the 8-bit LFSR............................................................... 51
3.27 Transient response of the 8 bit LFSR at 382.5 MHz ....................... 52
3.28 Transient response of the 16-bit LFSR at 378.5 MHz .................... 53
3.29 Transient response of the 32 bit LFSR at 373.5 MHz ................. 54
3.30 Schematic of the LFSR counter chip............................................... 55
3.31 Transient response LFSR counter when 8 bit LFSR is enabled........... 56
4.1 Hypothetical table for 4 bit LFSR (step 1)................................. 59
4.2 Hypothetical table for 4 bit LFSR (step 2)................................. 60
4.3 Stored column for 4-bit LFSR (step 3)........................................... 60
4.4 Hypothetical table with stored column for 4-bit LFSR (step 4)....... 61
5.1 Layout of the conventional binary counter ................................. 64
5.2 Layout of the LFSR counter............................................................... 64
5.3 Setup to generate the reset signal.................................................. 65
5.4 Output bits $Q_0 - Q_3$ at the maximum operating frequency (=5.6 MHz),
  when the 4-bit sub-counter is enabled. vdd = 1.1 V...................... 66
5.5 Output bits $Q_0 - Q_3$ at the frequency at which the counter
  fails (=5.7MHz), when the 4-bit sub-counter is enabled. vdd = 1.1V.... 67
5.6 Setup to generate the set signal....................................................... 68
5.7 Output bits $Q_0 - Q_3$ at the maximum operating frequency (=7MHz),
when the 4-bit LFSR is enabled. $v_{dd} = 1.1V$.

5.8 Output bits $Q_0 - Q_3$ at the frequency at which the LFSR counter fails ($=7.4MHz$), when the 4-bit LFSR is enabled. $v_{dd} = 1.1V$.

5.9 Transient response of the conventional counter when 4-bit sub-counter is enabled at $v_{dd} = 1.1v$ and maximum frequency of operation.

5.10 Transient response of the LFSR counter when the 4-bit LFSR is enabled at $v_{dd} = 1.1v$ and maximum frequency of operation.

B.1 Layout of the DFF.

B.2 Layout of the 1:4 Decoder.

B.3 Layout of the half adder or XOR gate.

B.4 Layout of the tri-state buffer.

B.5 Layout of the 4-bit counter.

B.6 Layout of the 4-bit LFSR.

B.7 Layout of the 8-bit counter.

B.8 Layout of the 8-bit LFSR.

B.9 Layout of the 16-bit counter.

B.10 Layout of the 16-bit LFSR.

B.11 Layout of the 32-bit counter.

B.12 Layout of the 32-bit LFSR.

B.13 Chip Layout of the conventional binary counter.

B.14 Chip Layout of the LFSR counter.

D.1 Picture of the function generator, Scope and power supply.

D.2 Picture of the Test Circuit.
D.3 Picture of the Test set-up

E.1 Bits $Q_0 - Q_3$ of the 8-bit counter at $vdd = 1.1V$ and $F_{max} = 4.1 MHz$...

E.2 Bits $Q_4 - Q_7$ of the 8-bit counter at $vdd = 1.1V$ and $F_{max} = 4.1 MHz$...

E.3 Bits $Q_4 - Q_7$ of the 8-bit counter at $vdd = 1.1V$ and $F_{fail} = 4.2 MHz$...

E.4 Bits $Q_0 - Q_3$ of the 16-bit counter at $vdd = 1.1V$ and $F_{max} = 2.7MHz$...

E.5 Bits $Q_4 - Q_7$ of the 16-bit counter at $vdd = 1.1V$ and $F_{max} = 2.7MHz$...

E.6 Bits $Q_8 - Q_{11}$ of the 16-bit counter at $vdd = 1.1V$ and $F_{max} = 2.7MHz$...

E.7 Bits $Q_4 - Q_7$ of the 16-bit counter at $vdd = 1.1V$ and $F_{max} = 2.7MHz$...

E.8 Bits $Q_{12} - Q_{15}$ of the 16-bit counter at $vdd = 1.1V$ and $F_{fail} = 2.8MHz$...

E.9 Bits $Q_0 - Q_3$ of the 8-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7 MHz$.......

E.10 Bits $Q_4 - Q_7$ of the 8-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7 MHz$.......

E.11 Bits $Q_4 - Q_7$ of the 8-bit LFSR at $vdd = 1.1V$ and $F_{fail} = 6.8 MHz$.......

E.12 Bits $Q_0 - Q_3$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

E.13 Bits $Q_4 - Q_7$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

E.14 Bits $Q_8 - Q_{11}$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

E.15 Bits $Q_{12} - Q_{15}$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

E.16 Bits $Q_0 - Q_4$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{fail} = 6.8MHz$.......

E.17 Bits $Q_0 - Q_3$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

E.18 Bits $Q_4 - Q_7$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.......

xiv
E.19  Bits $Q_8 - Q_{11}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.20  Bits $Q_{12} - Q_{15}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.21  Bits $Q_{16} - Q_{19}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.22  Bits $Q_{20} - Q_{23}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.23  Bits $Q_{24} - Q_{27}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.24  Bits $Q_{28} - Q_{31}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$.

E.25  Bit $Q_{31}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{fail} = 6.9MHz$. 


1. INTRODUCTION

Counters, as the name suggests, are used to count the occurrences of any event. Counting the cache hits and misses is among their more sophisticated applications. They are found in most digital systems and are considered standard items. The counters are classified as either decimal or binary counter, depending upon the count sequence they follow. They can also be classified as asynchronous or synchronous depending upon the way the registers are triggered. There have been many counter architectures proposed, but good silicon implementations have been a challenge since the early days of digital chip building.

The cycle time of the conventional binary counter is limited by the ripple carry delay. The minimum cycle time of these counters increases with \( N \), where \( N \) is the number of bits in the counter. These counters generally use flip-flops and full or half adders. Several fast adder techniques have been developed to reduce the ripple carry propagation delay.

This report explored the possibility of using Linear Feedback Shift Registers (LFSRs) as counters. LFSRs are pseudo-random pattern generators, finding applications in cryptography and wireless communications. To make the LFSR work as a counter, these pseudo-random patterns can be converted to a known binary count using an algorithm. The main advantage of LFSRs is that their minimum cycle time is independent of \( N \).

This work provides a direct comparison between a fast conventional binary counter built using a static Manchester carry chain and a counter built using an LFSR.
This comparison is focused on speed, power and area consumption. The two counters will be implemented in 0.5 μm process and sent to MOSIS fabrication foundry for fabrication. This technology was state-of-art in the mid 1990’s. The only reason to use this technology is that this is the only technology available under the MOSIS educational program for the academic institutions for chip fabrication.

“In 1995, Intel’s Pentium chip ran at 100 MHz” [Ans05]. To make a fair comparison between the two counters, we are targeting the same highest speed for the binary counter. A binary counter running at 100 MHz corresponds to a 10 nsec delay, and this delay depends on its length. Hence at this frequency the highest we intend to go is 32-bit. In contrast, the minimum cycle time of the LFSR counter, as discussed previously, is independent of its length and consequently its length could be way higher than 32-bits. But for the sake of comparison the maximum length of both the counters will be 32-bits.

In chapter 2, various types of counters are discussed. The fast adder techniques used in many counters are also discussed in detail. The last section describes the LFSRs and their two implementations.

In chapter 3, the design and simulations of the two counters in AMI 0.5μm technology is discussed. High speed operation and minimum silicon consumption were the two main objectives in their design. The circuits were optimized at various stages of design to achieve these goals.

Chapter 4 discusses the various algorithms that could be used to convert the pseudo-random LFSR count to a known binary count. The chapter also provides a
comparison between these algorithms, in terms of computer memory they consume and the average number of comparisons they perform for each conversion.

Chapter 5 describes the layout aspects of the conventional binary counter and the LFSR counter. The test setup is also discussed and measurements of the two counters are recorded and are compared with the simulation results.

In chapter 6, a summary of the results obtained in measurements and simulations are provided, along with some applications of the LFSR counters. Further extension of this work is also discussed in this chapter.
2. COUNTERS

A counter is a “sequential circuit that goes through a prescribed number of states when an input pulse is applied” [Mano72]. A vast majority of circuits designed using digital logic, employ counters. Their uses vary from counting the number of occurrences of a certain event to “generating timing sequences to control operations in a digital system” [Mano72]. Counters are of two main types: binary counters and linear feedback shift registers.

2.1 Binary Counters

Binary counters follow the binary number sequence. A \( N \) bit binary counter sequences through \( 2^N \) outputs before repeating, and its “minimum cycle time increases with \( N \)” [Wes02]. Binary counters can be classified as asynchronous and synchronous counters.

2.1.1 Asynchronous Binary Counters

The simplest asynchronous binary counter, shown in figure 2.1, is a series connection of flip-flops, where each flip-flop is clocked by the output of its preceding flip-flop. As shown in figure 2.2 the positive edge of each Data flip-flop (DFF) changes the output of the subsequent DFF. This leads to long delays. For a \( N \) bit counter, the worst case delay, i.e. when all the DFFs change their states, will be equal to \( N \) flip-flop transitions. Since these counters use only flip-flops, they have low cost and are preferable when the “speed of operation is not critical” [Mano72].
Figure 2.1: Asynchronous 4-bit counter

Figure 2.2: Timing diagram of asynchronous 4-bit counter
2.1.2 Synchronous Binary Counters

Unlike asynchronous binary counters, in which signals propagate or ripple through all flip-flops, in the synchronous binary counters “all the flip-flops are triggered simultaneously by the clock pulse” [Wes02]. This leads to higher speed of operation. There is a delay of a single flip-flop transition between the time the flip-flops are triggered by the clock pulse and the time they change their states.

Figure 2.3a: A 4-bit synchronous ripple up/down counter.
The simplest 4-bit carry ripple synchronous up/down counter, shown in figure 2.3a, uses full adders (FAs) and DFFs. The FAs can be reduced to half adders (HAs), if only up counting is required. The synchronous up counter is shown in figure 2.3b.

Figure 2.3b: A 4-bit synchronous ripple up counter.
These are called carry ripple counters because the carry-out for one bit position is simply connected as the carry in to the next. This ripple carry delay limits the cycle time. In order to reduce this delay several fast adder techniques can be employed. Some of these are discussed in the following section.

2.1.2.1 Adders

Bit addition is used for counting, multiplication and digital filtering [Wes 02].

2.1.2.1 Full Adder

A single bit adder with three inputs is called a full adder. The truth table for full adder with inputs \( A, B, C \) and outputs \( C_{out} \) and \( S \) along with signals Generate \((G)\), Propagate \((P)\) and Kill \((K)\), is shown in Table 2.1. The generate signal internally generates a carry high, so \( G = A \cdot B \). The propagate signal produces a \( C_{out} \) high only if \( C \) is true, so \( P = A \oplus B \). The kill signal kills a carry independent of \( C_{in} \) and hence \( K = \overline{A} \cdot \overline{B} \).

<table>
<thead>
<tr>
<th>( C )</th>
<th>( A )</th>
<th>( B )</th>
<th>( G = A \cdot B )</th>
<th>( P = A \oplus B )</th>
<th>( K = \overline{A} \cdot \overline{B} )</th>
<th>( C_{out} )</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.1: Truth table for full adder
From the truth table, the full adder logic is:

\[ S = ABC + \overline{A}BC + \overline{A}BC + ABC \]
\[ = (A \oplus B) \oplus C = P \oplus C \]  
\[ (2.1) \]

\[ C_{out} = AB + AC + BC \]
\[ = AB + C(A + B) \]  
\[ (2.2) \]

\[ \overline{C_{out}} = AB + C(A + B) \]  
\[ (2.3) \]

\[ \therefore S = ABC + (A + B + C)\overline{C_{out}} \]  
\[ (2.4) \]

\[ \overline{S} = ABC + (A + B + C)\overline{C_{out}} \]  
\[ (2.5) \]

The equations (2.4), (2.5), (2.6) and (2.7) signify that “the addition function is symmetric, i.e., the function of the complemented inputs is the complement of the function” [Wes02]. This leads to identical p-Metal-Oxide-Semiconductor (pMOS) and n-Metal-Oxide-Semiconductor (nMOS) networks, as shown in figure 2.4. This design “reduces the number of series transistors and makes the layout more uniform” [Wes02].

![Figure 2.4: Full Adder [Wes02.](image)
2.1.2.1.2 Carry-propagate Addition

In a carry propagate adder, the carry out of one stage is carry in to the next stage. These adders have \((A_0, A_1 \ldots A_N), (B_0, B_1 \ldots B_N),\) and carry-in \(C_0\) as the inputs and sum \((S_0, S_1 \ldots S_N)\) and carry-out of the last stage \(C_{out}\), as the outputs [Wes02]. The delay for these adders is dominated by the time it takes for the carries to ripple through each of \(N\) stages. This delay can reduced by using some of the techniques discussed below.

Ripple Carry Adder with Inverting Full Adders

The ripple carry adder used in the synchronous 4 bit ripple carry up/down counter of figure 2.3 is shown in figure 2.5a.

The problem with this adder is that the critical carry propagate path has inverters which lead to huge delays. These inverters can be omitted from the critical path by using inverting full adders every other stage. These inverting full adders take
inverted inputs and produces true outputs. Although there is delay in inverting adder inputs and sum outputs, they do not affect the critical path. The ripple carry adder with inverting full adders is shown in figure 2.5b. The ripple carry adders are good up to $N = 8$.

![4-bit ripple carry adder with inverting adders](image)

**Figure 2.5b: A 4-bit ripple carry adder with inverting adders.**

**Manchester Carry Chain Adder**

Unlike ripple carry adders which have AND-OR gates in their critical path, the “critical path of the Manchester carry chain adders has series propagate transistor for each bit of the adder” [Wes02]. This leads a significant amount of improvement in the speed of these adders [Wes02]. The $\overline{C_{out}}$ signal generated in figure 2.4 can also be generated using propagate, generate and kill signals as switch logic [Wes02]. The implementation of a complementary carry signal in figure 2.6a can be replaced with a static switch network as shown in figure 2.6b [Wes02]. A logic high on the propagate
signal $P$, will propagate the complementary carry signal through the transmission gate. The generate signal $G$, will generate the complementary carry using an nMOS transistor, whereas the kill signal $K$, will kill it using a pMOS transistor. The dynamic implementation of a complementary carry is shown in figure 2.6c. The dynamic implementation is faster and requires less hardware [Wes02].

![Figure 2.6a](image1)

**Figure 2.6a:** Carry chain design [Wes02].

![Figure 2.6b](image2)

**Figure 2.6b:** Static implementation of complementary carry [Wes02]
Figure 2.6c: Dynamic implementation of complementary carry [Wes02]

The static Manchester carry chain, shown in figure 2.7a, can be implemented by connecting multiple stages of the switch network shown in figure 2.6b.

Figure 2.7a: Static Manchester carry chain [Wes02]

Similarly, the dynamic Manchester carry chain, shown in figure 2.7b, connects multiple stages of the switch networks shown in figure 2.6c.
The number of stages that can be connected is limited by what is called the transmission line effect. This effect is based on the transmission line model which says that, for series connected transmission gates, the delay increases as the square of the number of transmission gates. Hence, after going through \( N \) transmission gates, where \( N = 3, 4, \) or \( 5 \) the logic level needs to be restored with a pair of inverters.

**Carry-lookahead Adders**

Carry-lookahead adders calculate carries to each stage in parallel. From equation 2.1 we have

\[
C_{\text{out}} = AB + C(A + B)
\]

For an \( N \)-bit adder the carry for \( (i+1) \) stage is given by:

\[
C_{i+1} = A_i B_i + C_i (A_i + B_i)
\]

\[
= G_i + C_i P_i
\]
where $G_i$ and $P_i$ are generate and propagate signals for the $i^{th}$ stage. Note that in equation 2.5 the propagate signal $P_i$ is changed from ‘XOR’ of the inputs to ‘OR’, without changing the outputs.

Recursively, we can substitute:

$$C_i = G_{i-1} + C_{i-1}P_{i-1}$$  \hspace{1cm} (2.10)

To get

$$C_{i+1} = G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + \ldots + P_1\ldots P_0C_0$$  \hspace{1cm} (2.11)

The sum bits are given by:

$$S_i = P_i \oplus C_i$$  \hspace{1cm} (2.12)

The number of stages in the carry-lookahead adder is usually limited to 3-5, so that the gates don’t get overly complex. The equations for the first three carry bits are given below:

$$C_1 = G_0 + P_0C_0$$  \hspace{1cm} (2.13)

$$C_2 = G_1 + P_1G_0 + P_1P_0C_0$$  \hspace{1cm} (2.14)

$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$  \hspace{1cm} (2.15)

The implementation of a carry-lookahead adder carry chain is similar to that of the dynamic Manchester carry chain shown in figure 2.7b.

**Carry-skip Adder**

The carry-skip or carry-bypass adder was proposed by Charles Babbage in the 19th century. This adder generates a group of propagate signal to skip over long carry
ripples. A dynamic carry-skip adder Manchester stage is shown in figure 2.8. The
group propagate signal is generated using an AND gate. It “selects the group carry in
if the group propagate is true or the ripple carry-out otherwise” [Wes02]. For a 32-bit
adder, arranging blocks or stages of length 4-4-5-6-7-6 can reduce the overall delay of
the carry chain and carry-select adders, discussed below.

Figure 2.8: Dynamic Carry-Skip adder Manchester Stage [Wes02]

**Carry-Select Adder**

The carry-select adder, shown in figure 2.9a, pre-computes outputs for carry-
in low and carry-in high, for each block of adders (except the first). Then using
multiplexers (figure 2.9b) or AND-OR (figure 2.9c) gates, selects the correct outputs.
Carry select adders are fast but require more hardware.
Figure 2.9a: Carry Select Adder

Figure 2.9b: Multiplexer
2.2 Linear Feedback Shift Registers

Linear Feedback Shift Registers (LFSRs) are found in most digital systems. They are frequently used to generate pseudo-random bit sequences. These sequences are based on specific mathematical algorithms and thus are repetitive and predictable. However, they appear to be random and non-repetitive if the cycle time is large. Hence they are called pseudo-random sequences. Apart from high-speed counters, LFSRs are being used for bit error rate measurements, wireless communication using CDMA techniques and cryptography [Nwi02].

An $N$ bit LFSR consists of $N$ registers and “at each clock edge, the contents of the registers are shifted right by one position” [George01]. The left most register gets feed back from predefined registers or taps through an exclusive-NOR (XNOR) or exclusive-OR (XOR) gate [George01]. A value of all 1’s, in the case of XNOR feedback and a value of all 0’s, in the case of XOR feedback are illegal because the LFSR would remain locked-up in this state [George01].
An LFSR of given size $N$ (number of registers) sequences through $2^N - 1$ outputs before repeating itself. This is the maximum achievable period and it depends on the feedback taps [Nwi02]. These feedback taps are often specified with a characteristic polynomial. For example, the taps for the 4-bit LFSR with characteristic polynomial $1 + x^3 + x^4$ will come after the 3rd and the 4th registers. Table 2.2 lists characteristic polynomials for some commonly used maximal-length LFSRs [Wes02].

<table>
<thead>
<tr>
<th>N</th>
<th>Characteristic polynomials</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$1 + x^2 + x^3$</td>
</tr>
<tr>
<td>4</td>
<td>$1 + x^3 + x^4$</td>
</tr>
<tr>
<td>5</td>
<td>$1 + x^3 + x^5$</td>
</tr>
<tr>
<td>6</td>
<td>$1 + x^5 + x^6$</td>
</tr>
<tr>
<td>7</td>
<td>$1 + x^6 + x^7$</td>
</tr>
<tr>
<td>8</td>
<td>$1 + x^1 + x^6 + x^7 + x^8$</td>
</tr>
<tr>
<td>9</td>
<td>$1 + x^5 + x^9$</td>
</tr>
<tr>
<td>15</td>
<td>$1 + x^{14} + x^{15}$</td>
</tr>
<tr>
<td>16</td>
<td>$1 + x^4 + x^{13} + x^{15} + x^{16}$</td>
</tr>
<tr>
<td>23</td>
<td>$1 + x^{18} + x^{23}$</td>
</tr>
<tr>
<td>24</td>
<td>$1 + x^{17} + x^{22} + x^{23} + x^{24}$</td>
</tr>
<tr>
<td>31</td>
<td>$1 + x^{28} + x^{31}$</td>
</tr>
<tr>
<td>32</td>
<td>$1 + x^{10} + x^{30} + x^{31} + x^{32}$</td>
</tr>
</tbody>
</table>

Table 2.2: Characteristic polynomials [Wes02]

### 2.2.1 LFSR Implementations

LFSRs can be implemented in two ways: Fibonacci implementation and Galois implementation [Nwi02]. The two implementations are shown in figure 2.10a and 2.10b. The implementations differ in the feedback loop.
The Fibonacci implementation has XOR gates in the feedback path, whereas the Galois implementation has it in the feed forward path. The reduced number of gates in the feedback loop makes the Galois implementation faster and the favored one [Nwi02]. The two implementations result in the same output sequence if they have the same feedback taps. The feedback taps in the figures shown above use XOR.
gates; hence the all 0’s state is illegal. Table 2.3 shows the output sequence of both implementations. As shown in the table, the sequence repeats itself after $2^4 - 1 = 15$ cycles.

<table>
<thead>
<tr>
<th>Cycle</th>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15(repeat)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.3: LFSR sequence

**2.2.2 LFSRs as Counters**

The LFSR can be very efficiently implemented in hardware. An $N$-bit LFSR requires only $N$ registers and a few XOR gates. This results in a very small circuit with very high operating speed. Moreover the “cycle time is set by the register and XOR delays, independent of $N$” [Wes02]. This results in extremely fast LFSR counters as there are no carry propagation delays. “LFSRs can replace conventional
binary counters in performance critical applications where the count sequence is not important” [George01].

Less logic leads to reduced area. Minimal routing complexity leads to higher speed. These are the two distinct advantages of an LFSR over a conventional binary counter, and are also the motivating factors behind this work.
3 COUNTERS: THEIR DESIGN, ARCHITECTURE and SIMULATIONS

The primary objectives in the design of the counters, a conventional binary counter using static Manchester carry chain and an LFSR counter, are high speed of operation with minimum silicon consumption. Theses goals can be achieved by optimizing the circuits at various stages of design. Both the counters are implemented using static logic gates. This is because dynamic logic gates suffer from the problem of sub-threshold and diode leakage and require periodic refreshing for “long lasting data retention” [Wes02]. This can be a major drawback for applications involving computer performance analysis, where in cache hits and misses are to be counted.

As described in the previous chapter, flip-flops, half-adders and XOR gates are the basic building blocks of the counters. Apart from these, there is a decoder and gated clock circuit in each counter, and a tri-state buffer at each output. The following section explains the design of these basic building blocks.

3.1 Basic Building Blocks

Data Flip-flop (DFF)

The DFF is a single bit storage circuit that switches its state on the rising or falling edge of the clock pulse. Figure 3.1 shows the schematic of a rising edge triggered static DFF. When the clock is low, Transmission Gates (TGs) T1 and T4 are on, whereas T2 and T3 are off. In this case the complemented D input reaches node C. When the clock goes high T1 and T4 are off, while T2 and T3 are on. This causes the true value of D to pass to Q. As TGs pass rail to rail voltages, the noise margin of the
circuit improves. Moreover, the input does have to fight the output of the feedback inverter and also the delay is reduced.

The transistors in the feedback path, shown by the two circles, are made minimum feature size i.e. each is $1.5\mu m / 0.6\mu m$. This reduces the capacitance at node $x2, x3$ and $x4$ and $Q$. It also increases the resistance of the transistors. But this increase in the delay in the feedback path is not critical.

The transistors in the feed forward path, i.e. between node $x1$ and $Q$ are unit sized. They are made stronger, with each having a multiplicity $2$. This reduces the resistance by a factor of 2, whereas the capacitance at node $x3$ and $Q$ remains the same. This is because the inverters in the feedback path, as explained in the previous paragraph, have a multiplicity of $1$ with the width of pMOS and nMOS transistors being $1.5\mu m / 0.6\mu m$. The overall effect is an increase in signal propagation speed.

Figure 3.1: Rising edge triggered static DFF with active low reset signal.
The delay from node D to node x3 is given by the following equations:

\[ t_{PHL} = \text{delay from } x2 \text{ to } x3 + \text{delay from } x1 \text{ to } x2 + \text{delay from } D \text{ to } x1. \]

\[ t_{PHL} = 0.7R_n^7 \left( C_{out n7} + C_{out p6} + C_{in n8} + C_{in p7} + \frac{C_{ox n10}}{2} + \frac{C_{ox p9}}{2} \right) \]

\[ + 0.7R_p^5 \left( \frac{C_{ox n6}}{2} + \frac{C_{ox p5}}{2} + C_{in n7} + C_{in p6} + \frac{C_{ox n9}}{2} + \frac{C_{ox p8}}{2} \right) \]

\[ + 0.7R_p^4 \left( \frac{C_{ox n4}}{2} + \frac{C_{ox p4}}{2} + C_{ox n5} + \frac{C_{ox n6}}{2} + \frac{C_{ox p5}}{2} \right) \]

(3.1)

where,
\[ t_{PHL} \] - high to low propagation delay.
\[ C_{out n} = Cox_n \text{ and } C_{out p} = Cox_p \]
\[ C_{in n} = \frac{3}{2}Cox_n \text{ and } C_{in p} = \frac{3}{2}Cox_p \]
\[ Cox_n = Cox'W_nL_n \text{ and } Cox_p = Cox'W_pL_p \]
\[ Cox' = 2.4 \mu F / \mu m^2 \]

In the term for the delay from \( x2 \) to \( x3 \), the resistance \( R_{n7} \) is decreased by a factor of 2. The increase in capacitances \( C_{out n7} \) and \( C_{out p6} \) is compensated by the decrease in other capacitance term, resulting in an overall decrease in the delay.

The reset signal used in this flip-flop is active low. This flip-flop will be used in the binary counter built using a static Manchester carry chain. The logic AND of \( D \) and the reset signal is implemented using a TG between node \( D \) and \( x1 \). This is because \( D \) will be the latest arriving signal, and in such cases the TG implementation
is much faster than the AND-OR-INVERT. All the transistors in this AND gate implementation are unit sized.

The clock signal is buffered through two inverters to make the transitions more abrupt; otherwise, all the TGs could be partially on at the same time, causing the logic levels to be intermediate. Moreover, it presents a low input capacitance to the global clock signal and also generates a local complementary clock signal.

Since the LFSR counter using XOR gates for the feedback taps have all 0s as an illegal state, they require DFFs with active-high ‘set’ signal. The set signal sets the states of all DFFs to 1s. Apart from this, their design is the same as DFFs with an active low reset signal. The schematic of the DFF using an active-high set signal is shown in figure 3.2

Figure 3.2: Rising edge triggered static DFF with active high set signal.
The transient response for the static rising edge triggered DFF with active high set signal is shown in figure 3.3. As can be seen in the figure, the output is high when the set signal is high and changes only on the rising edge of the clock pulse.

Figure 3.3: Transient response of the DFF shown in figure 3.2.

**Half Adder**

The truth table for half adder circuit generating sum bit, \( s \), is shown below.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1: Truth table for half adder circuit generating sum bit.
From the truth table, the half adder logic is:

\[ s = \overline{a}b + \overline{a}b = a \oplus b \quad (3.2) \]

The XOR logic is implemented using TGs as shown in figure 3.4. All the transistors are unit-sized with all pMOS transistors being 3.0\(\mu m\)/0.6\(\mu m\) and all the nMOS transistors being 1.5\(\mu m\)/0.6\(\mu m\). In the binary counter using the static Manchester carry chain the carry of the previous stage will go into ‘a’ and the other input will act as a select line ‘b’. For the LFSR counter, the XOR gates in the feedback taps will be implemented in a similar fashion, with the feedback going into \(a\), and \(b\) taking the feed forward value.

Figure 3.4: Half adder generating the sum bit \(s\).
**Tri-state Buffer**

A tri-state buffer is shown in figure 3.5. It is used at each output of the counters.

![Tri-state Buffer Diagram](image)

**Figure 3.5: A tri-state buffer.**

It enables only those outputs for which the enable signal $en$, is high. Again all the pMOS transistors are $3.0\mu m/0.6\mu m$ and all the nMOS transistors are $1.5\mu m/0.6\mu m$.

**1:4 Decoder**

<table>
<thead>
<tr>
<th>$s1$</th>
<th>$s0$</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$en4$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$en8$</td>
</tr>
</tbody>
</table>
The truth table of the 1:4 decoder is given in table 3.2. Each counter has 4 sub counters and the decoder generates the ‘enable’ signal to enable one of the 4 counters.
The decoder is implemented using AND gates as shown in figure 3.6. The AND gate used for the decoder is shown in figure 3.7.

The nMOS transistors of the AND gate have graded sizing, wherein the nMOS transistor closest to the output node has the smallest (W/L) ratio. In this design N1 is $1.5\mu m/0.6\mu m$ with multiplicity 1, while as N2 is $1.5\mu m/0.6\mu m$ with multiplicity 2.

As a result, the equivalent resistance of the nMOS transistor closest to the output node increases, whereas its parasitic drain capacitance decreases. This decrease in drain capacitance reduces the delay significantly. The increase in resistance has very
little impact on the combined RC delay. The pMOS transistors are made minimum feature size, with $P1$ and $P2$ both being $1.5\mu m/0.6\mu m$, further decreasing the node capacitance.

**Gated Clock**

The clock is gated to stop the outputs of those counters from switching states which are not enabled. This reduces dynamic power consumption and also makes it possible to measure the average power dissipation of each counter individually during testing. The gated clock circuit is shown in figure 3.8.

![Gated Clock Diagram](image)

Figure 3.8: Gated clock circuit.
The clock signal is passed through 2 input AND gates before being applied to the counters. The other inputs to these AND gates are the outputs of the 1:4 decoder, which also act as enable signal for the counters. In the above figure the number besides the inverter label gives it multiplicity. For example the label invx4 signifies an inverter with multiplicity 4. The multiplicity of these inverters is approximately chosen depending upon the number of inverters and TGs the clock signal is driving. As a rule of thumb, a scale factor of 2 to 4 is used in the design of these buffers.

With these basic building blocks the two counters will be designed. The following section gives the design specifications for the two counters.

### 3.2 Design Specifications

<table>
<thead>
<tr>
<th>Technology</th>
<th>AMI 0.5 µM technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>A conventional binary up counter using static Manchester carry chain.</td>
</tr>
<tr>
<td></td>
<td>A counter using LFSR.</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Vdd = 3.3 V; Vss = 0V</td>
</tr>
<tr>
<td>Input Signals</td>
<td>Set/reset, clock, select lines (s0, s1)</td>
</tr>
<tr>
<td>Output Signals</td>
<td>Bits (Q0, Qb1 ......., Q31)</td>
</tr>
<tr>
<td>Output load</td>
<td>20f F at each output</td>
</tr>
<tr>
<td>Targeted maximum operating Frequency</td>
<td>400 MHz</td>
</tr>
</tbody>
</table>

Table 3.3: Design Specifications
3.3 Static Manchester Carry Chain Counter

As the name suggests, this counter uses a static Manchester carry chain for carry propagation. The top level block diagram of the counter is shown in figure 3.9. It has four sub-counters of 4, 8, 16, and 32 bits respectively. A 1:4 decoder is used to enable the outputs of only one sub-counter at a time. Then there is a gated clock circuit to give the clock signal to only the sub counter which is enabled, so that other sub-counters do not switch states. The following section explains the design of each individual block.

![Block Diagram of the counter.](image)

Figure 3.9: Block Diagram of the counter.
3.3.1 The 4-Bit Static Manchester Carry Chain Counter

The block diagram of the 4-bit counter is shown in figure 3.10. It has a static Manchester carry chain and an array of half adders, DFFs, and tri-state buffers. The designs of half adders (for calculating sum bits), DFFs, and tri-state buffers have already been explained in section 3.1. The design of static Manchester carry chain is explained below.

**Static Manchester Carry Chain**

As shown in figure 3.10, the carry chain takes carry $C_0$ and bits $(A_0 : A_3)$ as inputs and produces carry bits $(C_1 : C_3)$ as outputs. Apart from these $CC_0$ and $\overline{CC}_0$ are also generated as outputs, for use when we go to higher hierarchal levels.
To understand the working of the Manchester carry chain let’s look at the truth table of half adder with the previous carry as one of the inputs and the present carry as the output. The truth table is shown in table 3.4.

<table>
<thead>
<tr>
<th>A</th>
<th>$C_i$</th>
<th>$C_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.4: Truth table for half adder circuit generating carry bit

As seen from the truth table, the previous carry is propagated if the input $A$ is high. If $A$ is low, it kills the carry. So effectively, a single stage of the static Manchester carry chain is a TG implementation of an AND gate with input $A$ as the select line. If we connect multiple stages of these AND gates we get the carry chain schematic shown in the figure 3.11. All the TGs in this figure have pMOS transistors of

Figure 3.11: 4 bit static Manchester carry chain.
size $3.0\mu m/0.6\mu m$ with multiplicity 2, and nMOS transistors of size $1.5\mu m/0.6\mu m$ and the same multiplicity. This leads to decreased propagation delay due to a decrease in the equivalent resistance of each stage. The pull-down nMOS transistors are sized $1.5\mu m/0.6\mu m$ with multiplicity 1, whereas inverters have unit size transistors with a multiplicity of 1.

The counter has carry $C_0$ at logic high. So the carry to the next hierarchal level will be high only if all the bits $(A_0 : A_3)$ are high. This provides a carry skip path as shown in the figure 3.12. It is a 4-input AND gate with graded sizing of nMOS transistors. All the nMOS and pMOS transistors except $P_5$ are sized $1.5\mu m/0.6\mu m$.

![Figure 3.12: Carry skip path.](image-url)
The nMOS transistors \( N1, N2, N3, \) and \( N4 \) have multiplicity of \( 1, 2, 3, \) and \( 4 \) respectively. This graded sizing reduces the propagation delay by the same explanation as given for graded AND used for 1:4 decoder in section 3.1. The size of \( P5 \) is \( 3.0 \mu m / 0.6 \mu m \).

With the design of all the blocks ready, let’s look at the complete schematic of the 4-bit counter shown in figure 3.13.

![Figure 3.13: Schematic of the 4 bit static Manchester carry chain counter.](image-url)
The XOR gates or half adder circuits for the calculation of sum bits use bits \((A_i : A_3)\) are the select lines. This is because the carry bits are the latest arriving signals are never used as select lines. Also, notice that the complementary outputs of the DFFs are fed to a tri-state buffer. This reduces the node capacitances on the propagation path and hence increases speed. The reset signal is active low and resets all the outputs to logic low. The maximum operating frequency of this 4 bit static Manchester carry chain counter is 294.5 MHz. The transient response at the maximum operating frequency is shown in figure 3.12 below.

Figure 3.14: Transient response of the 4 bit static Manchester carry chain counter.
Before all the size and multiplicity optimizations were done, the maximum operating speed of this counter was approximately 225 MHz. Hence there is a about a 30% improvement in its performance, with only a small area and power penalty.

3.3.2 The 8 Bit Static Manchester Carry Chain Counter

The architecture of the 8 bit static Manchester carry chain counter is shown in figure 3.15.

![Block Diagram of the 8 bit counter.](image)

Figure 3.15: Block Diagram of the 8 bit counter.
It has two 2 blocks of 4 bit static Manchester carry chain counter and one block of a 2-bit static Manchester carry chain. The design of the 4 bit counter has been explained in the previous section. The 2 bit Manchester carry chain with carry skip path is shown in figure 3.16. This chain and sizing of the transistors is the same as the carry chain of the 4 bit counter, except that it does not have the inverters. This is because both true and inverted block carry-outs of the 4 bit blocks are passed to the carry chain.

Figure 3.16: 8 bit static Manchester carry chain with carry skip path.
The schematic of the 8-bit counter is shown in figure 3.17. The input signals are clock, enable, reset, carry-in \( C_0 \) and the outputs are bits \( Q_0 : Q_7 \). The outputs \( C_\neg CC_0 \) and \( C_\neg CC_0 \) will be used for the 32 bit counter.

![Figure 3.17: Schematic of the 8 bit static Manchester carry chain counter.](image)

The maximum operating frequency of the static 8 bit Manchester carry chain counter is 213 MHz. The transient response of this counter at the maximum operation
frequency is shown in figure 3.18. The clock pulses are crowded because very many cycles (around 256 cycles) are required to show all the states of this counter.

![Figure 3.18: Transient response of the 8 bit static Manchester carry chain counter.]

3.3.3 The 16-Bit Static Manchester Carry Chain Counter

The 16-bit static Manchester carry chain counter has 4 blocks of 4-bit counters and one block of a 4-bit static Manchester carry chain. The 4-bit counter is the same as that given in section 3.3.1. The 4 bit Manchester carry is similar to the 4-bit chain used in the 4-bit counter in terms of the transistor sizing of TGs and inverters. The
The only difference is that in this case the inverted carry is propagated. The helps in negating the transmission line effect as explained in section 2.1, under the static Manchester carry chain adders. The static Manchester carry chain for this counter is shown in figure 3.19. The adding of the inverters breaks the transmission line and helps in restoring the logic to appropriate levels. Also, since the inverted carry is being propagated, the AND gate in the single stage of the carry chain in figure 3.11 is replaced by a TG implementation of an OR gate. The pull up pMOS transistors in these OR gates have sizes of $1.5 \mu m / 0.6 \mu m$. Since this is the minimum size possible for a pMOS transistor it further reduces the node capacitance in the propagation path, thereby increasing the speed. Moreover, we do not need inverters for inverting the signals for the select lines as both, the true and inverted block carry-outs of the 4 bit counters are passed to the carry chain. This saves some hardware. Also the carry skip path is only inside the 4 bit counters. There is not separate carry skip path, as the highest we intend to go is 32 bits.

Figure 3.19: 16-bit static Manchester carry chain with carry skip path.
The schematic of the 16-bit counter is shown in the figure 3.20. The input signals are clock, enable, reset and carry-in $C_0$ and the outputs are bits ($Q_0 : Q_{15}$). Due to limitations in the Cadence design environment, the transient response of this counter could not be obtained.

![Figure 3.20: Schematic of the 16 bit static Manchester carry chain counter.](image)

**3.3.4 The 32-Bit Static Manchester Carry Chain Counter**

The 32-bit static Manchester carry chain counter has 4 blocks of 8 bit counters and one block of a 4-bit static Manchester carry chain. The 8-bit counter is the same
as that given in section 3.3.2. The static Manchester carry chain used for the 32-bit counter is exactly the same, as described in the previous section. The schematic of the 32-bit counter shown in figure 3.21, has clock, enable, reset, and carry-in $C_0$ as the input signals and the bits ($Q_0 : Q_{31}$) as the outputs.

Figure 3.21: Schematic of the 32-bit static Manchester carry chain counter.
3.3.5 The complete static Manchester carry chain counter chip

The schematic of the complete Manchester carry chain counter chip is shown in figure 3.22. The decoder has two select lines to generate enable signals to enable one of four sub counters. The clock is also gated using these enable signals. As only an up counter is designed, the carry-in to all the sub counters is high.

Figure 3.22: Schematic of the static Manchester carry chain counter chip.
The transient response of this counter when the 4 bit and the 8 bit sub counters are enabled is similar to figure 3.14 and figure 3.18, respectively. The outputs which do not belong to the counter that is enabled are held low. Thus, if the enabled output is the 8-bit counter, outputs \((Q_8 : Q_{31})\) are held low, while outputs \((Q_0 : Q_7)\) go through the normal count sequences. The maximum operating frequency of the complete counter when the 4 bit counter is enabled is 292.5 MHz and when the 8 bit counter is enabled it is 213 MHz.

3.4 The LFSR Counter

The basic building blocks of this counter are the same as shown in figure 3.9. The only difference is that the sub counters use LFSRs, instead of half adder and the Manchester carry chain, and the reset signal is replaced by a set signal. A Galois mode feedback shift register is used in all the sub counters because of the reasons explained in section 2.2, under LFSR Implementations. The DFF used in this counter is shown in figure 3.2 and the design is explained in section 3.1. The XOR gates used in the feedback taps are the same as the half adder circuit explained in section 3.2. Since XORs gates are being used, the all 0’s state is illegal and every time the set signal goes high all the outputs are set high. All the sub counters are basically LFSRs and henceforth we will call them just LFSRs. The design of these LFSRs is discussed in the following section. Periodicity in the transient response tells us that the counter has failed and the frequency just before it gives its maximum frequency of operation.
3.4.1 The 4bit LFSR

The schematic of the 4 bit LFSR is shown in 3.23. The characteristic polynomial used for this LFSR is $1 + x^3 + x^4$. This means there is a feedback tap between the 3rd and 4th stage. The schematic of the 4-bit LFSR is shown in figure 3.23.

![Schematic of the 4 bit LFSR](image)

Figure 3.23: Schematic of the 4 bit LFSR.

![Transient response of the 4 bit LFSR at 404MHz](image)

Figure 3.24: Transient response of the 4 bit LFSR at 404MHz.
The feed forward signal is chosen as the select line input in the transmission gate implementation of the XOR gate. It reduces the propagation delay. This applies to all the feedback taps in each LFSR. The transient responses of this LFSR at its maximum frequency of operation, 404 MHz, and the frequency at which it fails, 405MHz are shown in figure 3.24 and figure 3.25 respectively.

Figure 3.25: Transient response of the 4 bit LFSR at 405MHz.

3.4.2 The 8 bit LFSR

The characteristic polynomial used for the 8 bit LFSR is \(1 + x^1 + x^6 + x^7 + x^8\). It results in feed back taps after the 1st, 6th and 7th stage. The schematic and transient response of this LFSR at its maximum operating frequency 382.5 MHz is shown in figure 3.26 and figure 3.27 respectively.
Figure 3.26: Schematic of the 8-bit LFSR
3.4.3 The 16-bit LFSR

The schematic of the 16 bit LFSR is similar to the 8 bit one, except that it has 16 stages or DFFs and the feedback taps are after stage 1, 12 and 15 corresponding the characteristic polynomial $1 + x^1 + x^{12} + x^{15} + x^{16}$. The maximum operating frequency of this LFSR is 378.5 MHz. The transient response is shown in figure 3.28.
3.4.4 The 32-bit LFSR

The 32 bit LFSR has feedback taps after the 1\textsuperscript{st}, 29\textsuperscript{th} and the 31\textsuperscript{st} stage. Its characteristic polynomial is $1 + x^1 + x^{29} + x^{31} + x^{32}$. The transient response of this LFSR at the maximum operating frequency of 373.5 MHz is shown in figure 3.28.
Figure 3.29: Transient response of the 32 bit LFSR at 373.5 MHz.
3.3.5 The complete LFSR counter chip

The schematic the complete LFSR counter chip is shown in figure 3.30. It is similar to the schematic of the static Manchester carry chain counter chip, except that the 4 sub counters are now the counters developed using LFSRs and the reset signal is replaced by the set signal.

Figure 3.30: Schematic of the LFSR counter chip.
The transient response of this counter when the 8 bit LFSR is enabled is shown in figure 3.31. The maximum operating frequency of the complete counter when the 4-bit LFSR is enabled is 380 MHz, is 361 MHz when the 8 bit LFSR is enabled, is 351 MHz when 16-bit LFSR is enabled and is 340 MHz when 32-bit counter is enabled.

![Figure 3.31: Transient response LFSR counter when 8 bit LFSR is enabled.](image)

The simulation results show that the LFSR counter is faster than conventional binary counter and presumably occupies less area. These results and presumptions will be validated in chapter 5, which deals with the test setup and measurement results. The next chapter discusses the algorithms to convert random output patterns of the LFSR to known binary count.
4 THE ALGORITHMS

In order for the LFSR to work as a counter, its pseudo random pattern needs to be converted to a known binary count. For example the 4-bit LFSR has the state 1111, when all the outputs are set high. At the next clock cycle its state is 0111. So the actual count when all outputs are set high is 15 (decimal equivalent of binary 1111). But for the LFSR to work as a counter all 1’s should represent count 0, because it is the first state of LFSR. We denote this count as true count. Similarly, the true count at the next clock cycle is 1, and so on.

There exist several algorithms to do this. Three of these algorithms are discussed below. Their respective codes are included in Appendix A. These algorithms are overhead, that is, are extra costs involved with the implementation of an LFSR counter.

4.1 Algorithm 1

This algorithm entails generating LFSR outputs and comparing them with the input (the LFSR state for which the true count is desired) at each loop iteration. The true count in this case is given by the loop iteration number in which the match was found.

The average number of comparisons, for an \( N \) bit LFSR are \( 2^N/2 \). The memory consumed by this algorithm is negligible, as there is no storage involved. But it is extremely slow, because the LFSR outputs need to be generated every time a new value is to be searched.
4.2 Algorithm II

This algorithm uses the decimal equivalent of the LFSR state as an index to an array. The loop iteration number in which a particular LFSR state is generated, is stored in the array indexed by that LFSR state. Table 4.1 gives the storage array for the 4-bit LFSR. For example the LFSR state 0001 (decimal equivalent = 1) occurs in the 9\textsuperscript{th} loop iteration, hence 9 is stored at the array location indexed by 1 -1 = 0. In this algorithm, the input value for which we need to find the true count is used as the index to a particular location in the array and the value stored at that location will be its true count.

<table>
<thead>
<tr>
<th>LFSR output - 1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop iteration #</td>
<td>9</td>
<td>10</td>
<td>6</td>
<td>11</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>12</td>
<td>13</td>
<td>4</td>
<td>14</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1 Storage array for the 4 bit LFSR

The look up time in this algorithm is negligible and is constant for any input value. The major drawback is that it hogs a lot of memory. For an N bit LFSR, it occupies about $2^N - 1$ bytes of memory which amounts to 16 GBytes for a 32 bit LFSR.
4.3 Algorithm III

This report uses algorithm III. It was developed by Dr. Eric E. Johnson [John]. The steps involved in this algorithm are explained using the 4-bit LFSR, as follows:

1. A hypothetical 4 x 4 table is created. For $N$ bit LFSR, it is a table of $2^{N/2} \times 2^{N/2}$. Each block of the table is labeled from 0 to 14, leaving the first block blank since the all 0’s state is not permissible. This is shown in figure 4.1.

![Figure 4.1 Hypothetical table for 4 bit LFSR (step 1).](image)

2. The LFSR outputs or states are entered into the hypothetical table, in the order in which they appear. This is shown in figure 4.2.
3. Only the last column is stored and its contents are sorted based on the LFSR state or outputs. Sorting is done in order to implement the binary search.
4. The input is searched in the stored column. If a match is found the block label gives the true count of the input. If not, the LFSR output is cycled until a match is found. The block label minus the number of cycles the LFSR needs to be run before a match is found gives the true count. For example in figure 4.4, if 0011 is the input, it will be searched in the stored column and a match will be found at block label 6, which is the true count. Now, if the input is 1100, it will not be found in the stored column. The LFSR will have to be run 2 cycles, taking 1100 as its initial state, to get to a state of 0010 which is present in stored column. Now the match will be found at block label 10. In this case $10 - 2 = 8$ (block label - # of LFSR cycles) is the true count of the input.

<table>
<thead>
<tr>
<th>Unsorted</th>
<th>Sorted</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1111</td>
<td>0111</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>1010</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>0110</td>
<td>1100</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>0100</td>
<td>1000</td>
</tr>
<tr>
<td>1100</td>
<td>1011</td>
</tr>
</tbody>
</table>

Figure 4.4 Hypothetical table with stored column for 4-bit LFSR (step 4).
The memory consumed for an $N$-bit LFSR in this case is $2^{N/2+1}$ bytes, whereas the average number of comparisons before a match is found is $2^{N/2-1} \log_2 2^{N/2}$. The main advantage of this algorithm is that it is extremely fast as compared to the algorithm I and consumes minimal memory as compared to algorithm II. In particular, for the 32-bit LFSR it consumes only 512 KBytes as compared to 16 GBytes for algorithm II.

In the next chapter, which deals with the test setup and measurement schemes, all the observations made during simulations are validated.
5. LAYOUT, TEST SETUP AND MEASUREMENT RESULTS

As described in chapter 3, two counters were designed in AMI 0.5\(\mu m\) technology. Both chips have been fabricated and tested. The two chips also include on-chip digital buffers to drive external load capacitances of up to 30 pF. In the layout of the two counters, the metal3 layer is not used until the topmost hierarchal level and in that it runs only vertically. The power rails and clock are the global signals; hence, they were made thick. Moreover, the clock is not routed over the power rails. Apart from this, an array of contacts was used, instead of a single contact, every time two lines were connected. This reduced the contact resistance. The Poly layer was not used for long interconnects, because it has high sheet resistance.

The area occupied by the conventional binary counter, developed using half adders and static Manchester carry chain, is about 362.4\(\mu m\) x 581.85\(\mu m\). Whereas the area occupied by the LFSR counter is 745.50\(\mu m\) x 167.50\(\mu m\). The counters are shown in figure 5.1 and 5.2 respectively. Clearly the LFSR counter consumes less area. The layouts of the basic building blocks and sub counters are shown in Appendix B. The area consumed by the sub-counters and LFSRs are given in section 5.1 in table 5.1 and 5.2 respectively.

A design rule check (DRC) and layout versus schematic (LVS) were performed on the layout of each block and the final chips before being submitted for fabrication. The fabricated chips were packaged in a DIP (Dual Inline Package) with 40 pins each. The following sections explain the test setup and measurement results.
for the two counters. The pin configuration and testing procedures for the two chips are included in Appendix C.

Figure 5.1: Layout of the conventional binary counter.

Figure 5.2: Layout of the LFSR counter.
5.1 Test Setup: Conventional Binary Counter

The maximum frequency of the waveform generated by the function generator is 15 MHz, whereas the maximum operating frequency of the circuits shown in chapter 3 is above 150 MHz. Reducing the power supply reduces the maximum frequency of operation of a circuit. Also the maximum difference between \( vdd \) and \( pad\_vdd \) should not be more than a threshold voltage drop, in order to have proper switching of the outputs. Hence a \( vdd \) of 1.1V and \( pad\_vdd \) of 1.8V is chosen. A by-pass capacitor is connected between \( vdd \) and \( ground \) and also between \( pad\_vdd \) and \( ground \). This reduces the high frequency noise. All the wires going to ground are hooked up to a single wire. The setup for the generation of the active low reset signal is shown in figure 5.3 below. It uses a 1M\( \Omega \) resistor, a 1\( \mu \)F capacitor and 3 inverters. Pictures of the complete test setup are shown in Appendix D.

![Figure 5.3: Setup to generate the reset signal.](image)

The select lines are used to enable one of the four sub counters. The frequencies of the output bits are compared with the clock frequency. For example,
the frequency of bit $Q_0$ should be half the clock frequency, whereas it is $1/4^{th}$ for bit $Q_1$ and so on. When both the select lines are low, the 4-bit sub-counter is enabled. The outputs (bits $Q_0 - Q_3$) of the conventional binary counter in this state and at its maximum frequency of operation are shown in figure 5.4. The bits $Q_4 - Q_{31}$ are measured using Digital Multi Meter (DMM) and are at logic low, as expected. Figure 5.5 shows the outputs at the frequency at which the 4-bit sub-counter fails. The remaining figures, showing the outputs of the counter at the maximum frequency of its operation and also at the frequency at which it fails are given in Appendix E.

![Figure 5.4: Output bits $Q_0 - Q_3$ at the maximum operating frequency (≈5.6 MHz), when the 4-bit sub-counter is enabled. vdd = 1.1 V](image-url)
Figure 5.5: Output bits $Q_0 - Q_3$ at the frequency at which the counter fails (=5.7MHz), when the 4-bit sub-counter is enabled. $vdd = 1.1V$

5.2 Test Setup: LFSR Counter

This counter too uses a $vdd$ of 1.1V and $pad_vdd$ of 1.8V as explained in the previous section. The test setup for the LFSR counter is similar to that used by the conventional binary counter, except the setup for the generation of the active high set signal, which is shown in figure 5.6. It uses one inverter less as compared to the setup for the generation of reset signal. As in the conventional binary counter, the select lines are used to enable one of the four LFSRs. The frequency at which the outputs appear to be periodic gives a fair idea of the maximum frequency of operation of the
counter. When both the select lines are low, the 4-bit LFSR is enabled and its outputs (bits $Q_3 - Q_0$) at its maximum frequency of operation are shown in figure 5.7.

Figure 5.6: Setup to generate the set signal.

Figure 5.7: Output bits $Q_0 - Q_3$ at the maximum operating frequency (=7MHz), when the 4-bit LFSR is enabled. $vdd = 1.1V$
The bits $Q_4 - Q_{31}$ measured using DMM and are at logic low. Figure 5.8 shows the outputs at the frequency at which the 4-bit LFSR fails. Observe the periodicity in the outputs in this figure. The remaining figures are shown in Appendix D.

Figure 5.8: Output bits $Q_0 - Q_3$ at the frequency at which the LFSR counter fails (=7.4MHz), when the 4-bit LFSR is enabled. $vdd = 1.1V$

5.3 Comparison of Simulation Results and Test Measurements

Simulations were carried out at $vdd = 1.1V$ for the two counters. The transient responses at the maximum operating frequency, when the 4-bit sub-counter and the 4-bit LFSR were enabled are shown in figure 5.9 and 5.10 respectively. The maximum operating frequency for each sub-counter and LFSR were noted for comparison with the actual test results.
Figure 5.9: Transient response of the conventional counter when 4-bit sub-counter is enabled at \( \text{vdd} = 1.1v \) and maximum frequency of operation.

Figure 5.10: Transient response of the LFSR counter when the 4-bit LFSR is enabled at \( \text{vdd} = 1.1v \) and maximum frequency of operation.
The table 5.1 shows and table 5.2 shows the simulation results and test results for the conventional binary counter and the LFSR counter respectively. In the table some circuits are not tested and some are not simulated. The reasons for these circuits not being tested are explained in section 5.1. Some circuits could not be simulated because of the limitations of the Cadence design environment. The 32-bit sub-counter is not testable at vdd = 1.1 V because the time for it to make the first transition is beyond the limitation of the scopes available in labs.

<table>
<thead>
<tr>
<th>Sub-counter enabled</th>
<th>Fmax at vdd = 3.3 V</th>
<th>Fmax at vdd = 1.1 V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simulation results (in MHz)</td>
<td>Test results (in MHz)</td>
</tr>
<tr>
<td>4-bit</td>
<td>294.5</td>
<td>Not tested</td>
</tr>
<tr>
<td>8-bit</td>
<td>213</td>
<td>Not tested</td>
</tr>
<tr>
<td>16-bit</td>
<td>Not simulated</td>
<td>Not tested</td>
</tr>
<tr>
<td>32-bit</td>
<td>Not simulated</td>
<td>Not tested</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of simulation and test results for the conventional counter

<table>
<thead>
<tr>
<th>Sub-counter enabled</th>
<th>Fmax at vdd = 3.3 V</th>
<th>Fmax at vdd = 1.1 V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simulation results (in MHz)</td>
<td>Test results (in MHz)</td>
</tr>
<tr>
<td>4-bit LFSR</td>
<td>404</td>
<td>Not tested</td>
</tr>
<tr>
<td>8-bit LFSR</td>
<td>382.5</td>
<td>Not tested</td>
</tr>
<tr>
<td>16-bit LFSR</td>
<td>378.5</td>
<td>Not tested</td>
</tr>
<tr>
<td>32-bit LFSR</td>
<td>373.5</td>
<td>Not tested</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of simulation and test results for the LFSR counter
In both the above tables the test results are 30 to 35% lower than the simulation results. There are several reasons for this disparity. First is that simulations do not include parasitics associated with metal layers and metal contacts in the layout. The layout also suffers with cross talk between metal conductors. Second, is that the clock signals cannot be distributed with perfect synchronization and there is a difference in arrival time between two stages. This is called clock skew. Clock signal also deviates from its ideal position leading to what is called clock jitter. The third reason is that the power supply is not ideal. It has a finite source resistance due to which the voltage reaching the circuits is less than the desired. Then there is thermal noise, which affects the mobility of electrons. Chip process variations also affect circuit delays.

5.4 Comparison of the Conventional Counter and the LFSR Counter

The table 5.3 and table 5.4 show the comparison of the conventional counter and the binary counter, based on simulation results. Table 5.3 shows their average power dissipation whereas table 5.4 shows the maximum operating frequency of each counter with relative speed-up taking the conventional binary counter as the base. Table 5.5 summarizes the test results along with the layout area of each counter.

<table>
<thead>
<tr>
<th>4-bit LFSR @ Fmax = 404 MHz</th>
<th>4-bit LFSR @ F = 294MHz</th>
<th>4-bit binary counter @ Fmax = 294 MHz</th>
<th>% change (LFSR counter as base)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.71mW</td>
<td>1.36mW</td>
<td>1.18mW</td>
<td>-13%</td>
</tr>
</tbody>
</table>

72
**Table 5.3: Summary of simulation results: Average power dissipation**

<table>
<thead>
<tr>
<th>Length of the counter</th>
<th>Fmax (in MHz) at vdd = 3.3V</th>
<th></th>
<th>Fmax (in MHz) at vdd = 1.1V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-bit LFSR @ Fmax = 382.5 MHz</td>
<td></td>
<td>8-bit binary counter @ Fmax = 213 MHz</td>
<td>% change (LFSR counter as base)</td>
</tr>
<tr>
<td></td>
<td>3.608mW</td>
<td>2.003mW</td>
<td>1.314mW</td>
<td>-34%</td>
</tr>
</tbody>
</table>

**Table 5.4: Summary of simulation results: Maximum operating frequency**

<table>
<thead>
<tr>
<th>Length of the counter</th>
<th>Fmax (in MHz) at vdd = 3.3V</th>
<th></th>
<th>Fmax (in MHz) at vdd = 1.1V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conventional Counter</td>
<td>LFSR Counter</td>
<td>Speed-up (with conventional counter as base)</td>
<td>Conventional Counter</td>
</tr>
<tr>
<td>4-bit</td>
<td>294.5</td>
<td>404</td>
<td>37%</td>
<td>8.5</td>
</tr>
<tr>
<td>8-bit</td>
<td>213</td>
<td>382.5</td>
<td>80%</td>
<td>6</td>
</tr>
<tr>
<td>16-bit</td>
<td>Not simulated</td>
<td>378.5</td>
<td></td>
<td>NotSimulated</td>
</tr>
<tr>
<td>32-bit</td>
<td>Not simulated</td>
<td>373.5</td>
<td></td>
<td>NotSimulated</td>
</tr>
</tbody>
</table>

**Table 5.5: Summary of the test results and layout area**

<table>
<thead>
<tr>
<th>Length of the counter</th>
<th>Fmax (in MHz) at vdd = 1.1V</th>
<th></th>
<th>Layout Area (in µm²)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conventional Counter</td>
<td>LFSR Counter</td>
<td>Speed-up (with conventional counter as base)</td>
<td>Conventional Counter</td>
</tr>
<tr>
<td>4-bit</td>
<td>5.6</td>
<td>7</td>
<td>25%</td>
<td>9980</td>
</tr>
<tr>
<td>8-bit</td>
<td>4.1</td>
<td>6.7</td>
<td>63%</td>
<td>21103</td>
</tr>
<tr>
<td>16-bit</td>
<td>2.7</td>
<td>6.7</td>
<td>148%</td>
<td>43144</td>
</tr>
<tr>
<td>32-bit</td>
<td>Not tested</td>
<td>6.7</td>
<td></td>
<td>97339</td>
</tr>
<tr>
<td>Chip</td>
<td></td>
<td></td>
<td></td>
<td>210862</td>
</tr>
</tbody>
</table>
Test results validate the simulations, wherein the LFSR counter was found to be faster than the conventional binary counter. Test results also show that as the number of bit $N$, increases the maximum operating frequency of the conventional counter decreases. We can estimate the maximum operating frequency of 32-bit conventional counter to be about 1.5 MHz at a power supply of 1.1V. On the other hand increasing the number of bits has a negligible effect on the LFSR counter. Its maximum operating frequency remains almost constant at about 6.7 MHz at a power supply of 1.1V, independent of $N$. Also, as discussed at the beginning of this chapter, the LFSR counter has a distinct advantage in terms of low silicon consumption.
6. CONCLUSIONS, APPLICATIONS AND RECOMMENDATIONS

6.1 Conclusions

This report has dealt with the comparison of the conventional binary counter and the LFSR counter. These two counters were fabricated and the measurement results were compared with simulations. This work assumed the availability of resources to convert the LFSR count to a binary count in software, after the benchmark test simulation is complete. The Algorithm to do the same was implemented in the ‘C’ programming language. It is an overhead to the LFSR counter.

Both counters were designed for high speed and lower area consumption. As seen from the simulation results in chapter 3 and test measurements in chapter 5, the LFSR counter has higher speed of operation. For 16 and 32 bit counters the speed of the LFSR counter was more than double the speed of the conventional binary counter. This is because it has minimum routing complexity. As seen in chapter 5 the area consumed by the LFSR counter is about $0.1248 \text{mm}^2$, whereas the area consumed by the counter using the static Manchester carry chain is about $0.2108 \text{mm}^2$. The LFSR counter consumes only about $3/5\text{th}$ the area consumed by the conventional binary counter. This can be attributed to less required logic for the LFSR counter. Hence, reduced area and higher speed are the two distinct advantages of the LFSR counter.

The conventional binary counter has the advantage of providing direct results. Also it has low average power dissipation compared to the LFSR counter. The is due the fact that the bits in LFSR counter switch states more frequently and randomly than the binary counter.
6.2 Applications

LFSR counters can be used for developing extremely high speed counters. Since this work builds the LFSR counter using static logic gates, it is suitable for applications involving computer performance analysis. In particular it can be used to count cache hits and misses, which are vital statistics in analyzing the performance of a computer.

6.3 Recommendations

At a power supply of 3.3V the counters operate at frequencies which cannot be tested by the equipments in the lab. A scheme to make the power supply programmable can be developed on chip. Moreover, a dedicated clock trunk could be provided on the chip. Clock skew is suppressed by providing branch lines from the trunk line and supplying the clock to each flip-flop from the branch lines. Also post layout simulations could be carried out to measure the effect of layout on test results. Apart from this exhaustive simulation and test measurements could be carried out. The Cadence design environment could be interfaced with MATLAB to obtain the simulation results of the 16-bit conventional binary counter. Initializing the 16 or 32-bit binary sub-counter to the state where only the most significant bit (MSB) is 0 while others are all 1’s, and then running the counter for 2 clock cycles to see if the MSB switches properly, can be another way of testing or simulating the two binary sub-counters. Finally the possibility of decoding LFSR outputs in hardware could be explored.
APPENDICES
APPENDIX A: Algorithm implementation in ‘C’ programming language

Code 1: The following code implements a function to generate LFSR outputs for 4, 8, 16 and 32 bits.

/* Linear feedback Shift Register */
int lfsr(unsigned long int dff, int BITS)
{
    long unsigned int temp,temp0, temp1, temp3, temp6;
    long unsigned int temp7, temp12, temp15, temp29, temp31, tempall;

    /* 4-bit Linear feedback Shift Register */
    if (BITS == 4)
    {
        temp = dff;
        temp0 = (temp >> 3) & 1;
        tempall = (temp << 1) & 6;
        temp3 = ((temp << 1)^ temp);
        temp3 = temp3 & 8;
        dff   = temp0 | tempall |temp3;
    }

    /* 8-bit Linear feedback Shift Register */
    else if (BITS == 8)
    {
        temp = dff;
    
...
temp0 = temp >> 7;
temp1 = ((temp >> 7) ^ temp);
temp1 = ((temp1 << 1) & 2);
tempall = ((temp << 1) & 60);
temp6 = ((temp >> 2) ^ temp);
temp6 = (temp6 << 1) & 64;
temp7 = ((temp >> 1) ^ temp);
temp7 = (temp7 << 1) & 128;
dff = temp0 | temp1 | tempall | temp6 | temp7;
}

/* 16-bit Linear feedback Shift Register */

else if (BITS == 16)
{

temp = dff;
temp0 = temp >> 15;
temp1 = ((temp >> 15) ^ temp);
temp1 = ((temp1 << 1) & 2);
tempall = (temp << 1) & 28668;
temp12 = ((temp >> 4) ^ temp);
temp12 = (temp12 << 1) & 4096;
temp15 = ((temp >> 1) ^ temp);
temp15 = (temp15 << 1) & 32768;
```c
    dff = temp0 | temp1 | tempall | temp12 | temp15;

    /* 32-bit Linear feedback Shift Register */
    else if (BITS == 32)
    {
        temp = dff;
        temp0 = temp >> 31;
        temp1 = ((temp >> 31) ^ temp);
        temp12 = ((temp1 << 1) & 2);
        tempall = (temp << 1) & 1610612732;
        temp29 = ((temp >> 3) ^ temp);
        temp29 = (temp29 << 1) & 536870912;
        temp31 = ((temp >> 1) ^ temp);
        temp31 = (temp31 << 1) & 2147483648ul;
        dff = temp0 | temp1 | tempall | temp29 | temp31;
    } 
    return dff;
    }
```
Code 2: The following code implements a function to display the LFSR output bits.

/*Function to print binary bits of a number*/

int showbits (unsigned long int n, int size)
{
    int i;
    unsigned long int k, andmask;
    for (i = size-1; i >= 0; i--)
    {
        andmask = 1 << i;
        k = n & andmask;
        if (k == 0)
            printf ("0");
        else
            printf ("1");
    }
    printf ("
");
    return 0;
}
Code 3: The following code implements a function to return an index such that only the last column of the hypothetical table in Algorithm 111 is stored.

/*Function to convert blockable to index*/

int ndx(unsigned long int n, unsigned long int len)
{
    int i = 0;
    i = n / len;
    return i;
}
Code 4: The following code implements Algorithm 1

/* Complete Linear feedback Shift Register with no memory storage */

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include "showbits.c"
#include "lfsr.c"

#define BITS 4 /* define BITS according to the LFSR to be enabled e.g. 4,8,16,32*/

int main()
{
    long unsigned int dff;
    long unsigned int j;
    unsigned char answer, ch[0];
    int decimalch;
    int more = 1;
    while (more)
    {
        dff = (pow(2,BITS)-1);
        printf("Enter the %d-bit value to search:\n", BITS);
        scanf("%s",ch);
        decimalch = strtol(ch, NULL, 2); /*converts string to long int i.e
binary i/p to decimal*/
for (j = 0 ; j <= pow(2,BITS)-2; j++)
{
    showbits(dff, BITS);
    if(decimalch == dff) /*searches for a match in each loop
iteration*/
    break;
    dff = lfsr(dff, BITS);
}
printf("\n");
printf("count = %lu \n", j);
printf("Do you want to run it again? (Y/N) ");
scanf("%c",&answer);
scanf("%c",&answer);
if('Y'==answer || 'y'== answer)
    more=1;
else
    more=0;
}
return 0;
Code 5: The following code implements Algorithm 11

/*LFSR counter with memory storage n lookup*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "showbits.c"
#include "lfsr.c"

#define BITS 8 /*define BITS according to the LFSR to be enabled e.g. 4,8,16,32*/

int main()
{
    int decimalch, match;
    long unsigned int dff;
    long unsigned int j;
    unsigned char answer, ch[200];
    long unsigned int truecount[(1 << BITS)-1];
    int more=1;
    dff = pow(2,BITS)-1;
    for (j = 0 ; j <= (pow(2,BITS)-2); j++)
    {
        truecount[dff-1] = j; /*dff outputs are used as index to store the loop
        iteration number in which they occur*/
        showbits(dff, BITS);
dff = lfsr(dff, BITS);

}
while (more)
{

dprintf("Enter the %d-bit value to search:\n", BITS);
scanf("%s",ch);

decimalch = strtol(ch, NULL, 2); /*converts string to long int i.e
binary i/p to decimal*/
match = truecount[decimalch-1]; /* just loop using the i/p "decimalch"
as index */
dprintf("\n");
dprintf("count = %d \n", match);
dprintf("Do you want to look up another value? (Y/N) ");
scanf("%c",&answer);
scanf("%c",&answer);
if('Y'==answer || 'y'== answer)
    more=1;
else
    more=0;
}
return 0 ;
}
Code 6: The following code implements Algorithm 111

/*LFSR counter with low memory storage and less # of comparisons*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "showbits.c"
#include "index1.c"
#include "lfsr.c"

#define BITS 16 /* define BITS according to the LFSR to be enabled e.g. 4,8,16,32*/

int main()
{

    int match = 0;
    long unsigned int dff, len, b, count, mid, low, hi;
    long unsigned int i, j, k, min;
    unsigned char answer, ch[200];
    long unsigned int blocklable[(1 << BITS/2)], blockno[(1 << BITS/2)],
    decimalch, t, c;
    int more = 1;
    dff = pow(2,BITS)-1;
    len = pow(2,BITS)-1;
    printf("dff = %lu\n", dff);
    b = pow(2,BITS/2);
printf("b = \%lu\n", b);

for (i = 0 ; i < len; i++)
{
    if ((i + 2) % b == 0)
    {
        blocklable[ndx(i, b)] = dff; /*dff outputs are inserted in the
        stored column */
        blockno[ndx(i, b)] = i; /* block number is inserted in another
        stored column*/
    }

    showbits(dff, BITS);

    dff = lfsr(dff, BITS);
}

/* Following for loop implements selection sorting
based on blocklable or stored LFSR outputs. Then
the blockno. are also sorted accordingly.
Sorting is done only to implement binary search*/
for ( j = 0; j < b ; j++ )
{
    min = j;

    for (k = j+1; k < pow(2,(BITS/2)); k++)
    {

if (blocklable[k] < blocklable[min])
    min = k;
}
t = blocklable[min];
blocklable[min] = blocklable[j];
blocklable[j]= t;
t = blockno[min];
blockno[min] = blockno[j];
blockno[j]= t;
}
while (more)
{
    printf("Enter the %d-bit value to search:\n", BITS);
    scanf("%s",ch);
    decimalch = strtol(ch, NULL, 2);
    match = 0;
    c = 0;
    /* Following while loop implements binary search*/
    while ( match != 1)
    {
        low = 0;
        hi = b-1;
while (low <= hi)
{
    mid = (low + hi)/2;

    if ((hi == 0 && decimalch < blocklable[mid]) ||
        (low == b-1 && decimalch > blocklable[mid]))
    {break;
     /* this if statement looks for a match*/

    if (decimalch == blocklable[mid])
    {

        count = blockno[mid] - c;
        printf("count = %lu\n",count);
        match = 1;
        break;
    }

    if (decimalch < blocklable[mid])

        hi = mid - 1;

    else

        low = mid + 1;

    }

    /*if match is not found LFSR is run one
     cycle and counter is increased by one*/
if (match != 1)
{
    decimalch = lfsr(decimalch, BITS);
    c = c + 1;
}

printf("Do you want to look up another value? (Y/N) ");
scanf("%c",&answer);
scanf("%c",&answer);
scanf("%c",&answer);
if('Y'==answer || 'y'== answer)
    more=1;
else
    more=0;
}

return 0 ;
APPENDIX B: Layouts

Layouts of the basic building blocks, sub-counters and LFSRs and the two chips are shown. The ruler is placed along the length and width of the layout. The units shown on the ruler are in lambda ($\lambda$) where $\lambda=0.3\mu$m.

Figure B.1: Layout of the DFF

Figure B.2: Layout of the 1:4 decoder
Figure B.3: Layout of the half adder or XOR gate

Figure B.4: Layout of the tri-state buffer
Figure B.5: Layout of the 4-bit counter

Figure B.6: Layout of the 4-bit LFSR
Figure B.7: Layout of the 8-bit counter

Figure B.8: Layout of the 8-bit LFSR
Figure B.9: Layout of the 16-bit counter

Figure B.10: Layout of the 16-bit LFSR
Figure B.11: Layout of the 32-bit counter

Figure B.12: Layout of the 32-bit LFSR
Figure B.13: Chip Layout of the conventional binary counter
Figure B.14: Chip Layout of the LFSR counter
### APPENDIX C: Pin Configurations and Test Procedures

#### C.1 Conventional binary Counter

<table>
<thead>
<tr>
<th>P #</th>
<th>Name</th>
<th>Pad Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b15</td>
<td>DigBuf</td>
<td>Bit b15 output of the counter</td>
</tr>
<tr>
<td>2</td>
<td>b14</td>
<td>DigBuf</td>
<td>Bit b14 output of the counter</td>
</tr>
<tr>
<td>3</td>
<td>b13</td>
<td>DigBuf</td>
<td>Bit b13 output of the counter</td>
</tr>
<tr>
<td>4</td>
<td>b12</td>
<td>DigBuf</td>
<td>Bit b12 output of the counter</td>
</tr>
<tr>
<td>5</td>
<td>s1</td>
<td>Protect</td>
<td>+1.1 or 0V depending upon the counter to be enabled</td>
</tr>
<tr>
<td>6</td>
<td>s0</td>
<td>Protect</td>
<td>+1.1 or 0V depending upon the counter to be enabled</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td>DigBuf</td>
<td>Bit b7 output of the counter</td>
</tr>
<tr>
<td>8</td>
<td>b6</td>
<td>DigBuf</td>
<td>Bit b6 output of the counter</td>
</tr>
<tr>
<td>9</td>
<td>b5</td>
<td>DigBuf</td>
<td>Bit b5 output of the counter</td>
</tr>
<tr>
<td>10</td>
<td>b4</td>
<td>DigBuf</td>
<td>Bit b4 output of the counter</td>
</tr>
<tr>
<td>11</td>
<td>b0</td>
<td>DigBuf</td>
<td>Bit b0 output of the counter</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
<td>DigBuf</td>
<td>Bit b1 output of the counter</td>
</tr>
<tr>
<td>13</td>
<td>b2</td>
<td>DigBuf</td>
<td>Bit b2 output of the counter</td>
</tr>
<tr>
<td>14</td>
<td>b3</td>
<td>DigBuf</td>
<td>Bit b3 output of the counter</td>
</tr>
<tr>
<td>15</td>
<td>vdd</td>
<td>Protect</td>
<td>+1.1V, to power counters, decoder and gated clock circuits</td>
</tr>
<tr>
<td>16</td>
<td>Vss</td>
<td>Protect</td>
<td>0V, to power counters, decoder and gated clock circuits</td>
</tr>
<tr>
<td>17</td>
<td>b11</td>
<td>DigBuf</td>
<td>Bit b11 output of the counter</td>
</tr>
<tr>
<td>18</td>
<td>b10</td>
<td>DigBuf</td>
<td>Bit b10 output of the counter</td>
</tr>
<tr>
<td>19</td>
<td>b9</td>
<td>DigBuf</td>
<td>Bit b9 output of the counter</td>
</tr>
<tr>
<td>20</td>
<td>b8</td>
<td>DigBuf</td>
<td>Bit b8 output of the counter</td>
</tr>
<tr>
<td>21</td>
<td>b19</td>
<td>DigBuf</td>
<td>Bit b19 output of the counter</td>
</tr>
<tr>
<td>22</td>
<td>b18</td>
<td>DigBuf</td>
<td>Bit b18 output of the counter</td>
</tr>
<tr>
<td>23</td>
<td>b17</td>
<td>DigBuf</td>
<td>Bit b17 output of the counter</td>
</tr>
<tr>
<td>24</td>
<td>b16</td>
<td>DigBuf</td>
<td>Bit b16 output of the counter</td>
</tr>
<tr>
<td>25</td>
<td>reset</td>
<td>Protect</td>
<td>Input to the counters. It is active low.</td>
</tr>
<tr>
<td>26</td>
<td>Vss</td>
<td>vss</td>
<td>0V, to Digital PADFRAME for digital output buffers and protection.</td>
</tr>
<tr>
<td>27</td>
<td>b27</td>
<td>DigBuf</td>
<td>Bit b27 output of the counter</td>
</tr>
<tr>
<td>28</td>
<td>b26</td>
<td>DigBuf</td>
<td>Bit b26 output of the counter</td>
</tr>
<tr>
<td>29</td>
<td>b25</td>
<td>DigBuf</td>
<td>Bit b25 output of the counter</td>
</tr>
<tr>
<td>30</td>
<td>b24</td>
<td>DigBuf</td>
<td>Bit b24 output of the counter</td>
</tr>
<tr>
<td>31</td>
<td>b28</td>
<td>DigBuf</td>
<td>Bit b28 output of the counter</td>
</tr>
<tr>
<td>32</td>
<td>b29</td>
<td>DigBuf</td>
<td>Bit b29 output of the counter</td>
</tr>
<tr>
<td>33</td>
<td>b30</td>
<td>DigBuf</td>
<td>Bit b30 output of the counter</td>
</tr>
<tr>
<td>34</td>
<td>b31</td>
<td>DigBuf</td>
<td>Bit b31 output of the counter</td>
</tr>
<tr>
<td>35</td>
<td>Pad_vdd</td>
<td>vdd</td>
<td>+1.8 V, to Digital PADFRAME for digital output buffers and</td>
</tr>
</tbody>
</table>

100
Table C.1: Pin Configuration of the conventional binary counter

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>clk Protect 1.1 Vpp 10MHz square wave, input to gated clock circuits.</td>
</tr>
<tr>
<td>37</td>
<td>b20 DigBuf Bit b20 output of the counter</td>
</tr>
<tr>
<td>38</td>
<td>b21 DigBuf Bit b21 output of the counter</td>
</tr>
<tr>
<td>39</td>
<td>b22 DigBuf Bit b22 output of the counter</td>
</tr>
<tr>
<td>40</td>
<td>b23 DigBuf Bit b23 output of the counter</td>
</tr>
</tbody>
</table>

Suggested Test Procedure for the conventional binary counter

Vdd = +1.1 V (logic 1), Vss = 0 V (logic 0), pad_vdd = +1.8 V

1. Attach the following inputs to Vss: s1 (pin 5), s0 (pin 6), reset (pin 25), clk (pin 36), vss (pin 26) and pad_vss (pin 16).

2. Attach Vdd to vdd (pin 15) and pad_vdd to pad_vdd (pin 35). Measure the DC current through each of these connections. It should be zero, meaning, nothing in the digital circuitry should be switching or consuming any power.

3. Remove the Vss connection to clk (pin 36). Apply a 1MHz, 1.1 Vpp square wave to clk (pin 36). Make sure the DC offset of the function generator equals 0v. Attach 10x probes to clk (pin 36). Using either the DMM or the scope, measure the input and output clock frequencies, which should be 1 MHz:

   clock Freq:  

4. Remove the Vss connection to reset (pin 25). Use the set up given in figure 5.3 under section 5.1 to generate the reset signal. Give this signal to reset (pin 25).

5. Attach 10x probes to b0 (pin 11) and b1 (pin 12). Using either the scope or logic analyzer, measure the frequencies, which should be 1/2 x clk freq and 1/4 x clk freq, respectively:

   b0 Freq:  b1 Freq:  

6. Attach 10x probes to b2 (pin 13) and b3 (pin 14). Using either the scope or logic analyzer, measure the frequencies, which should be 1/8 x clk freq and 1/16 x clk freq, respectively:

   b2 Freq:  b3 Freq:  
7. Observe the maximum frequency of operation.

\[ F_{\text{max}}_{\text{4bit}} = \] 

8. Using DMM measure the outputs b4 to b32. They should be low as s0 and s1 are low and only the 4 bit counter is enabled.

9. Remove Vss connection to s0 (pin 6) and connect it to Vdd. This should enable the 8 bit counter.

10. Attach 10x probes to b0 (pin 11), b1 (pin 12), b2 (pin 13), b3 (pin 14), b4 (pin10), b5 (pin 9), b6 (pin 8), b7 (pin 7) sequentially. Using either the scope or logic analyzer, measure the frequencies, which should decrease by factor of two compared to the frequency of the previous output. Observe the maximum frequency of operation.

\[ F_{\text{max}}_{\text{8bit}} = \] 

11. Using DMM measure the outputs b8 to b32. They should be low as s0 is high and s1 is low and so only the 8 bit counter is enabled.

12. Remove Vdd connection to s0 (pin 6) and connect it to Vss. Also, remove Vss connection to s1 (pin 5) and connect it to Vdd. This should enable the 16bit counter.

13. Attach 10x probes to b0 (pin 11), b1 (pin 12), b2 (pin 13), b3 (pin 14), b4 (pin10), b5 (pin 9), b6 (pin 8), b7 (pin 7), b8 (pin 20), b9 (pin 19), b10 (pin 18), b11 (pin 17), b12 (pin 4), b13 (pin 3), b14 (pin 2), b15 (pin 1) sequentially. Using either the scope or logic analyzer, measure the frequencies, which should decrease by factor of two compared to the frequency of the previous output. Calculate the maximum frequency of operation.

\[ F_{\text{max}}_{\text{16bit}} = \] 

14. Using DMM measure the outputs b16 to b32. They should be low as s0 is low and s1 is high and so only the 16 bit counter is enabled.

15. Remove Vss connection to s0 (pin 6) and connect it to Vdd. This should enable the 32 bit counter.

16. Attach 10x probes to b0 (pin 11), b1 (pin 12), b2 (pin 13), b3 (pin 14), b4 (pin10), b5 (pin 9), b6 (pin 8), b7 (pin 7), b8 (pin 20), b9 (pin 19), b10 (pin 18), b11 (pin 17), b12 (pin 4), b13 (pin 3), b14 (pin 2), b15 (pin 1), b16 (pin 24), b17 (pin 23), b18 (pin 22), b19 (pin 21), b20 (pin 37), b21 (pin 38), b22 (pin 39), b23 (pin 40), b24 (pin 30), b25 (pin 29), b26 (pin 28), b27 (pin 27), b28 (pin 31), b29 (pin 32), b29 (pin 32), b30 (pin 31), b31 (pin 30), b32 (pin 29), b33 (pin 28), b34 (pin 27), b35 (pin 26), b36 (pin 25), b37 (pin 24), b38 (pin 23), b39 (pin 22), b40 (pin 21), b41 (pin 20), b42 (pin 19), b43 (pin 18), b44 (pin 17), b45 (pin 16), b46 (pin 15), b47 (pin 14), b48 (pin 13), b49 (pin 12), b50 (pin 11), b51 (pin 10), b52 (pin 9), b53 (pin 8), b54 (pin 7), b55 (pin 6), b56 (pin 5), b57 (pin 4), b58 (pin 3), b59 (pin 2), b60 (pin 1), b61 (pin 0).
b30 (pin 33), b31 (pin 34) sequentially. Using either the scope or logic analyzer, measure the frequencies, which should decrease by factor of two compared to the frequency of the previous output. Calculate the maximum frequency of operation.

\[ F_{\text{max, 32bit}} = \quad \]
## C.2 LFSR Counter

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Pad Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b11</td>
<td>DigBuf</td>
<td>Bit b11 output of the LFSR</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td>DigBuf</td>
<td>Bit b10 output of the LFSR</td>
</tr>
<tr>
<td>3</td>
<td>b9</td>
<td>DigBuf</td>
<td>Bit b9 output of the LFSR</td>
</tr>
<tr>
<td>4</td>
<td>b8</td>
<td>DigBuf</td>
<td>Bit b8 output of the LFSR</td>
</tr>
<tr>
<td>5</td>
<td>s0</td>
<td>Protect</td>
<td>+1.1 or 0V depending upon the LFSR to be enabled</td>
</tr>
<tr>
<td>6</td>
<td>s1</td>
<td>Protect</td>
<td>+1.1 or 0V depending upon the LFSR to be enabled</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td>DigBuf</td>
<td>Bit b7 output of the LFSR</td>
</tr>
<tr>
<td>8</td>
<td>b6</td>
<td>DigBuf</td>
<td>Bit b6 output of the LFSR</td>
</tr>
<tr>
<td>9</td>
<td>b5</td>
<td>DigBuf</td>
<td>Bit b5 output of the LFSR</td>
</tr>
<tr>
<td>10</td>
<td>b4</td>
<td>DigBuf</td>
<td>Bit b4 output of the LFSR</td>
</tr>
<tr>
<td>11</td>
<td>b3</td>
<td>DigBuf</td>
<td>Bit b3 output of the LFSR</td>
</tr>
<tr>
<td>12</td>
<td>b2</td>
<td>DigBuf</td>
<td>Bit b2 output of the LFSR</td>
</tr>
<tr>
<td>13</td>
<td>b1</td>
<td>DigBuf</td>
<td>Bit b1 output of the LFSR</td>
</tr>
<tr>
<td>14</td>
<td>b0</td>
<td>DigBuf</td>
<td>Bit b0 output of the LFSR</td>
</tr>
<tr>
<td>15</td>
<td>vdd</td>
<td>Protect</td>
<td>+1.1 V, to power LFSRs, decoder and gated clock circuits</td>
</tr>
<tr>
<td>16</td>
<td>vss</td>
<td>Protect</td>
<td>0V, to power LFSRs, decoder and gated clock circuits</td>
</tr>
<tr>
<td>17</td>
<td>b15</td>
<td>DigBuf</td>
<td>Bit b15 output of the LFSR</td>
</tr>
<tr>
<td>18</td>
<td>b14</td>
<td>DigBuf</td>
<td>Bit b14 output of the LFSR</td>
</tr>
<tr>
<td>19</td>
<td>b13</td>
<td>DigBuf</td>
<td>Bit b13 output of the LFSR</td>
</tr>
<tr>
<td>20</td>
<td>b12</td>
<td>DigBuf</td>
<td>Bit b12 output of the LFSR</td>
</tr>
<tr>
<td>21</td>
<td>b31</td>
<td>DigBuf</td>
<td>Bit b31 output of the LFSR</td>
</tr>
<tr>
<td>22</td>
<td>b16</td>
<td>DigBuf</td>
<td>Bit b16 output of the LFSR</td>
</tr>
<tr>
<td>23</td>
<td>b30</td>
<td>DigBuf</td>
<td>Bit b30 output of the LFSR</td>
</tr>
<tr>
<td>24</td>
<td>b17</td>
<td>DigBuf</td>
<td>Bit b17 output of the LFSR</td>
</tr>
<tr>
<td>25</td>
<td>set</td>
<td>Protect</td>
<td>Input to LFSRs. It is active high.</td>
</tr>
<tr>
<td>26</td>
<td>vss</td>
<td>vss</td>
<td>0V, to Digital PADFRAME for digital output buffers and protection.</td>
</tr>
<tr>
<td>27</td>
<td>b29</td>
<td>DigBuf</td>
<td>Bit b29 output of the LFSR</td>
</tr>
<tr>
<td>28</td>
<td>b18</td>
<td>DigBuf</td>
<td>Bit b18 output of the LFSR</td>
</tr>
<tr>
<td>29</td>
<td>b28</td>
<td>DigBuf</td>
<td>Bit b28 output of the LFSR</td>
</tr>
<tr>
<td>30</td>
<td>b19</td>
<td>DigBuf</td>
<td>Bit b19 output of the LFSR</td>
</tr>
<tr>
<td>31</td>
<td>b27</td>
<td>DigBuf</td>
<td>Bit b27 output of the LFSR</td>
</tr>
<tr>
<td>32</td>
<td>b20</td>
<td>DigBuf</td>
<td>Bit b20 output of the LFSR</td>
</tr>
<tr>
<td>33</td>
<td>b26</td>
<td>DigBuf</td>
<td>Bit b26 output of the LFSR</td>
</tr>
<tr>
<td>34</td>
<td>b21</td>
<td>DigBuf</td>
<td>Bit b21 output of the LFSR</td>
</tr>
<tr>
<td>35</td>
<td>pad_vdd</td>
<td>vdd</td>
<td>+1.8 V, to Digital PADFRAME for digital output buffers and</td>
</tr>
</tbody>
</table>
Table C.2: Pin Configuration of the LFSR counter

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>clk</td>
</tr>
<tr>
<td>37</td>
<td>b25 DigBuf</td>
</tr>
<tr>
<td>38</td>
<td>b22 DigBuf</td>
</tr>
<tr>
<td>39</td>
<td>b24 DigBuf</td>
</tr>
<tr>
<td>40</td>
<td>b23 DigBuf</td>
</tr>
</tbody>
</table>

Suggested Test Procedure for the LFSR counter

Vdd = +1.1 V (logic 1), Vss = 0 V (logic 0), pad_vdd = +1.8 V

1. Attach the following inputs to Vss: s1 (pin 6), s0 (pin 5), set (pin 25), clk (pin 36), vss (pin 26) and pad_vss (pin 16).

2. Attach Vdd to vdd (pin 15) and pad_vdd to pad_vdd (pin 35). Measure the DC current through each of these connections. It should be zero, meaning, nothing in the digital circuitry should be switching or consuming any power.

3. Remove the Vss connection to clk (pin 36). Apply a 1 MHz, 1.1 Vpp square wave to clk (pin 36). Make sure the DC offset of the function generator equals 0v. Attach 10x probes to clk (pin 36). Using either the DMM or the scope, measure the input and output clock frequencies, which should be 1 MHz:

   clock Freq: __________

4. Remove the Vss connection to set (pin 25). Use the setup given in figure 5.6 under section 5.2 to generate the set signal. Give this signal to set (pin 25).

5. Using the 10x probes look at outputs b0 (pin 14), b1 (pin 13), b2 (pin 12), b3 (pin 11) on the scope or logic analyzer.

6. Observe these outputs for 15 clock cycles. Observe the outputs for periodicity. The frequency at which outputs become periodic gives a fair idea of the maximum frequency of operation.

   Fmax_4bit = __________

7. Using DMM measure the outputs b4 to b32. They should be low as s0 and s1 are low and only the 4 bit LFSR is enabled.
8. Remove Vss connection to s0 (pin 5) and connect it to Vdd. This will enable the 8 bit LFSR.

9. Using the 10x probes look at outputs b0 (pin14), b1 (pin13), b2 (pin 12), b3 (pin 11), b4 (pin 10), b5 (pin 9), b6 (pin 8), b7 (pin 7) on the scope or logic analyzer. Repeat step 6.

   \[ F_{max\_8bit} = \quad \] 

10. Using DMM measure the outputs b8 to b32. They should be low as s0 is high and s1 is low and so only the 8 bit counter is enabled.

11. Remove Vdd connection to s0 (pin 5) and connect it to Vss. Also, remove Vss connection to s1 (pin 6) and connect it to Vdd. This should enable the 16 bit LFSR.

12. Using the 10x probes look at outputs b0 (pin14), b1 (pin13), b2 (pin 12), b3 (pin 11), b4 (pin 10), b5 (pin 9), b6 (pin 8), b7 (pin 7), b8 (pin4), b9 (pin3), b10 (pin2), b11 (pin1), b12 (pin 20), b13 (pin 19), b14 (pin 18), b15 (pin 17) on the scope or logic analyzer. Repeat step 6.

   \[ F_{max\_16bit} = \quad \] 

13. Using DMM measure the outputs b16 to b32. They should be low as s0 is low and s1 is high and so only the 16 bit counter is enabled.

14. Remove Vss connection to s0 (pin 5) and connect it to Vdd. This will enable the 32 bit LFSR.

15. Using the 10x probes look at outputs b0 (pin14), b1 (pin13), b2 (pin 12), b3 (pin 11), b4 (pin 10), b5 (pin 9), b6 (pin 8), b7 (pin 7), b8 (pin4), b9 (pin3), b10 (pin2), b11 (pin1), b12 (pin 20), b13 (pin 19), b14 (pin 18), b15 (pin 17), b16 (pin22), b17 (pin24), b18 (pin 28), b19 (pin 27), b20 (pin 32), b21 (pin 34), b22 (pin 38), b23 (pin 40), b24 (pin39), b25 (pin37), b26 (pin33), b27 (pin31), b28 (pin 29), b29 (pin 27), b30 (pin 23), b31 (pin 21) on the scope or logic analyzer. Repeat step 6.

   \[ F_{max\_32bit} = \quad \]
APPENDIX D: Pictures of the Circuit, Test Setup and Lab Equipments

Figure D.1: Picture of the function generator, Scope and power supply.

Figure D.2: Picture of the Test Circuit
Figure D.3: Picture of the Test set-up
APPENDIX E: Output Bits at Maximum Operating Frequency and Frequency at which the Counters Fail.

The following figures show the outputs of the two counters at their maximum operating frequency (Fmax) and also at frequency at which they fail (Ffail). In each figure the least significant bit is at the bottom of the figure and the next significant bits follow going towards the top.

8-Bit Conventional Counter

Figure E.1 Bits $Q_0 - Q_3$ of the 8-bit counter at vdd = 1.1V and Fmax = 4.1 MHz

Figure E.2 Bits $Q_4 - Q_7$ of the 8-bit counter at vdd = 1.1V and Fmax = 4.1 MHz
Figure E.3 Bits $Q_4 - Q_7$ of the 8-bit counter at $vdd = 1.1V$ and $F_{fail} = 4.2$ MHz
16-Bit Conventional Counter

Figure E.4 Bits $Q_0 - Q_3$ of the 16-bit counter at $vdd = 1.1V$ and $Fmax = 2.7MHz$

Figure E.5 Bits $Q_4 - Q_7$ of the 16-bit counter at $vdd = 1.1V$ and $Fmax = 2.7MHz$

Figure E.6 Bits $Q_8 - Q_{11}$ of the 16-bit counter at $vdd = 1.1V$ and $Fmax = 2.7MHz$
Figure E.7 Bits $Q_{12} - Q_{15}$ of the 16-bit counter at $vdd = 1.1V$ and $F_{max} = 2.7MHz$

Figure E.8 Bits $Q_{12} - Q_{15}$ of the 16-bit counter at $vdd = 1.1V$ and $F_{fail} = 2.8MHz$
In the each of following figures the waveform at the center of each figure is the clock with its frequency showing at the bottom.

**8-Bit LFSR**

Figure E.9 Bits $Q_0 - Q_3$ of the 8-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7\ MHz$

Figure E.10 Bits $Q_4 - Q_7$ of the 8-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7\ MHz$

Figure E.11 Bits $Q_4 - Q_7$ of the 8-bit counter at $vdd = 1.1V$ and $F_{fail} = 6.8\ MHz$
16-Bit LFSR

Figure E.12 Bits $Q_0 - Q_3$ of the 16-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz

Figure E.13 Bits $Q_4 - Q_7$ of the 16-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz

Figure E.14 Bits $Q_8 - Q_{11}$ of the 16-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz
Figure E.15 Bits $Q_{12} - Q_{15}$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{max} = 6.7MHz$

Figure E.16 Bits $Q_0 - Q_1$ of the 16-bit LFSR at $vdd = 1.1V$ and $F_{fail} = 6.8MHz$
32-Bit LFSR

Figure E.17 Bits $Q_0 - Q_3$ of the 32-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz

Figure E.18 Bits $Q_4 - Q_7$ of the 32-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz

Figure E.19 Bits $Q_8 - Q_{11}$ of the 32-bit LFSR at vdd = 1.1V and Fmax = 6.7MHz
Figure E.20 Bits $Q_{12} - Q_{15}$ of the 32-bit LFSR at $V_{dd} = 1.1\text{V}$ and $F_{max} = 6.7\text{MHz}$.

Figure E.21 Bits $Q_{16} - Q_{19}$ of the 32-bit LFSR at $V_{dd} = 1.1\text{V}$ and $F_{max} = 6.7\text{MHz}$.

Figure E.22 Bits $Q_{20} - Q_{23}$ of the 32-bit LFSR at $V_{dd} = 1.1\text{V}$ and $F_{max} = 6.7\text{MHz}$.
Figure E.23 Bits $Q_{24} - Q_{27}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{\text{max}} = 6.7MHz$

Figure E.24 Bits $Q_{28} - Q_{31}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{\text{max}} = 6.7MHz$

Figure E.25 Bit $Q_{31}$ of the 32-bit LFSR at $vdd = 1.1V$ and $F_{\text{fail}} = 6.9MHz$
REFERENCES


[John] E.E. Johnson, Professor, Department of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM.

