A Configurable Asynchronous Pseudorandom Bit Sequence Generator

Alex Chow, Bill Coates, David Hopkins
VLSI Research Group
Sun Microsystems Laboratories
Outline

• Pseudorandom bit sequences (PRBSs)
• Traditional PRBS generators
• An asynchronous implementation
  > Design
  > Properties
• Extension to other types of circuits
Properties & Applications of PRBSs

• A statistically “random” bit sequence
  > Behaves like noise
  > Carries an equal number of 1's and 0's
  > Has near zero correlation with other PRBSs

• Useful for
  > Testing I/O channels
  > Cryptography
  > Spread-spectrum communication systems
Generating a PRBS
Using a Linear Feedback Shift Register (LFSR)

• Next input bit is a linear function of the current state
Generating a PRBS

Using a Linear Feedback Shift Register (LFSR)

- Next input bit is a linear function of the current state
- Feedback logic is a sum (XOR) of a subset of bits in the register
Feedback Tap Sequence

- Not every tap sequence generates a PRBS
- A PRBS must be a maximal-length sequence
  - An $N$-stage LFSR produces a PRBS of length $2^N - 1$
  - The PRBS contains every $N$-bit pattern (except all 0's)
- Maximal tap sequence found using finite-field math
Feedback Tap Sequence

Notation

• We list the positions of all tapped bits
• Example: 5-stage LFSR with taps at positions 3, 5
  > Notation: \( N = 5 \), Taps = \([5,3]\)
  > Produces a PRBS with length \( 2^5 - 1 = 31 \)
LFSR Logical Constraints

- Properties of the PRBS strictly defined by LFSR topology, i.e.
  - Number of LFSR stages
  - Tap sequence (location and number of taps)
- Each LFSR circuit can produce only one PRBS
An Asynchronous PRBS Generator

- Started as a “Friday project”
- Replace shift register with asynchronous FIFO stages
- We present a GasP implementation
Two-Tap PRBS Generator

The Circuit

![Diagram of a two-tap PRBS generator circuit with data path and control path indicated.]
Two-Tap PRBS Generator
GasP Control Elements

Data path
Control path

Control branch

Feedback Logic

DATA OUT
CTL OUT

Data path
Control path
Enforcing Proper Bit Sequence

• LFSR uses global clock
  > Every stage contains valid data
  > Data moves in lock-step
  > Bit sequencing and synchronization implicitly enforced

• Async implementation requires explicit control
  > Not every stage contains valid data
  > Data bits may propagate autonomously
  > Must explicitly synchronize pairs of control tokens to guarantee correct data sequencing
Two-Tap PRBS Generator

Example

• Emulate 5-stage LFSR with taps = [5,3]
Two-Tap PRBS Generator

Example

- Emulate 5-stage LFSR with taps = [5,3]
Two-Tap PRBS Generator

Example

- Emulate 5-stage LFSR with taps = [5,3]
Two-Tap PRBS Generator

Example

- Emulate 5-stage LFSR with taps = [5,3]
Two-Tap PRBS Generator

Example

- Emulate 5-stage LFSR with taps = [5,3]

[Diagram of a two-tap PRBS generator with logical taps [5,3] and data bits labeled 1 to 5, along with control tokens and output ports.]
Physical & Logical Independence

- **LFSR:**
  - Physical tap sequence = logical tap sequence

- **Asynchronous implementation:**
  - Logical tap sequence depends only on token distribution
  - Physical circuit structure need only **enable** logical tap sequence, not **match** it
Configurability

• Async control decouples logical bit distribution from physical circuit structure
• One circuit can produce many different PRBSs
Generating Different PRBSs

Example

- Construct a circuit to generate all two-tap maximal-length patterns of order $N \leq 7$

<table>
<thead>
<tr>
<th>Order $N$</th>
<th>Logical Taps</th>
<th># Tokens Seg 1</th>
<th># Tokens Seg 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>[3,2]</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>[4,3]</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>[5,3]</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>[6,5]</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>[7,4]</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>[7,6]</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
Generating All Patterns for $N \leq 7$

- $N = 3$
- Taps = [3,2]
Generating All Patterns for $N \leq 7$

- $N = 4$
- Taps = $[4,3]$
Generating All Patterns for $N \leq 7$

- $N = 5$
- Taps = [5,3]
Generating All Patterns for $N \leq 7$

- $N = 6$
- Taps = [6,5]
Generating All Patterns for $N \leq 7$

- $N = 7_A$
- Taps = [7,4]
Generating All Patterns for $N \leq 7$

- $N = 7_B$
- Taps = [7,6]

![Diagram showing a circuit with labels and connections for generating patterns.](image)
Throughput and I/O

• Throughput varies for different patterns
  > Because token occupancy is different for each pattern

• May use the generator in synchronous systems
  > Take the output through async-to-clocked interfaces
  > Clock rate must be lower than minimum async throughput
Summary

- Async design decouples number and distribution of logical bits from physical shift register structure.
- Exploit this orthogonality to let one FIFO circuit adopt multiple logical configurations.
- PRBS generator serves as an illustrative example.
  > Can apply this idea to other circuits.
  > Other examples: CRC generators, encoders, filters.
Extension

- Example: Cyclic Redundancy Checksum (CRC) generator
Questions
Alex Chow
alex.chow@sun.com