State Machine Design Process

1. Determination of inputs and outputs.
2. Determination of machine states.
3. Create State/Bubble Diagram—should this be a Mealy or Moore machine?
4. State Assignment—assign each state a particular value.
5. Create Transition/Output Table
6. Derive Next State Logic for each state element—using K-maps as necessary.
7. Derive Output logic.
8. Implement in Xilinx.
Counters (DDPP 8.4)

• Ripple Counters gate the clock and so are not acceptable

Synchronous Counter

• Counting is simply adding “1” to previous value.
• Use CoreGen to create Counters
Modulo-n Counter

- If only the MSB of the Counter is considered, the counter creates a pulse every n seconds.
  - Used to create a slower periodic signal than core system clock.
  - Feed into synchronous enables to make state machines run slower
    - See Lab 5 Skeleton example.

Shift Registers (DDPP 8.5)

- Used to Convert from serial data stream to parallel data stream and vice versa
More Shift Registers

Parallel-In, Serial-Out

Parallel-In, Parallel-Out

Shift Register Applications

- Reduce Number of wires between components in a system
  - If high speed, call SerDes (Serializer, Deserializers)
  - Must make sure to synchronize signals....
LFSR (DDPP 8.5.8)

- Linear Feedback Shift Register
  - Non Binary Counter
  - Counts $2^n - 1$ states

LFSR Feedback Logic

<table>
<thead>
<tr>
<th>$n$</th>
<th>Feedback Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$X_2 = X_1 \oplus X_0$</td>
</tr>
<tr>
<td>3</td>
<td>$X_3 = X_1 \oplus X_0$</td>
</tr>
<tr>
<td>4</td>
<td>$X_4 = X_1 \oplus X_0$</td>
</tr>
<tr>
<td>5</td>
<td>$X_5 = X_2 \oplus X_0$</td>
</tr>
<tr>
<td>6</td>
<td>$X_6 = X_1 \oplus X_0$</td>
</tr>
<tr>
<td>7</td>
<td>$X_7 = X_3 \oplus X_0$</td>
</tr>
<tr>
<td>8</td>
<td>$X_8 = X_4 \oplus X_3 \oplus X_2 \oplus X_0$</td>
</tr>
<tr>
<td>12</td>
<td>$X_{12} = X_9 \oplus X_4 \oplus X_1 \oplus X_0$</td>
</tr>
<tr>
<td>16</td>
<td>$X_{16} = X_5 \oplus X_4 \oplus X_3 \oplus X_0$</td>
</tr>
<tr>
<td>20</td>
<td>$X_{20} = X_3 \oplus X_0$</td>
</tr>
<tr>
<td>24</td>
<td>$X_{24} = X_7 \oplus X_2 \oplus X_1 \oplus X_0$</td>
</tr>
<tr>
<td>28</td>
<td>$X_{28} = X_3 \oplus X_0$</td>
</tr>
<tr>
<td>32</td>
<td>$X_{32} = X_{22} \oplus X_2 \oplus X_1 \oplus X_0$</td>
</tr>
</tbody>
</table>
LFSR Applications

- LFSR’s used to create pseudo-random bit sequences.
- Also, very small and compact synchronous counter.
  - Very fast
  - Why faster than binary?

Iterative vs Sequential Circuits

- DDPP 8.6
- Space vs Time Tradeoff
Serial Synchronous Circuits

- Small but long latency

Another view of Decomposition

- DDPP 8.7
VGA Circuit

Figure 1: Digital-to-analog VGA monitor interface.

VGA Waveform

Figure 2: VGA signal timing.
Lab 5 Skeleton

- VGA Document at
  - Fair game for midterm #2
- Look at in Xilinx and demo it
  - VGA Sync module
  - Character ROM
  - Registered inputs (but really should register outputs)