CSE477
VLSI Digital Circuits
Fall 2003

Lecture 22: Shifters, Decoders, Muxes

Mary Jane Irwin (www.cse.psu.edu/~mji)
www.cse.psu.edu/~cg477

Review: Basic Building Blocks

- Datapath
  - Execution units
    - Adder, multiplier, divider, shifter, etc.
  - Register file and pipeline registers
  - Multiplexers, decoders

- Control
  - Finite state machines (PLA, ROM, random logic)

- Interconnect
  - Switches, arbiters, buses

- Memory
  - Caches (SRAMs), TLBs, DRAMs, buffers
Parallel Programmable Shifters

Control = 

Shift amount ($Sh_2Sh_1Sh_0$)
Shift direction (left, right)
Shift type (logical, arithmetic, circular)

Data In $\rightarrow$ Data Out

Shifters used in multipliers, floating point units

Consume lots of area if done in random logic gates
**A Programmable Binary Shifter**

A diagram illustrating the operation of a programmable binary shifter. The diagram shows the inputs and outputs for different shift operations. The table provides the values for each shift operation.

<table>
<thead>
<tr>
<th>(A_i)</th>
<th>(A_{i-1})</th>
<th>(B_i)</th>
<th>(B_{i-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_1)</td>
<td>(A_0)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(A_1)</td>
<td>(A_0)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(A_1)</td>
<td>(A_0)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
A Programmable Binary Shifter

<table>
<thead>
<tr>
<th>A_i</th>
<th>A_{i-1}</th>
<th>rgt</th>
<th>nop</th>
<th>left</th>
<th>B_i</th>
<th>B_{i-1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_1</td>
<td>A_0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A_1</td>
<td>A_0</td>
</tr>
<tr>
<td>A_1</td>
<td>A_0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A_1</td>
</tr>
<tr>
<td>A_1</td>
<td>A_0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A_0</td>
<td>0</td>
</tr>
</tbody>
</table>
4-bit Barrel Shifter

Example:

\[ \overline{\text{Sh}_1} \overline{\text{Sh}_0} = 1 \]
\[ B_3 B_2 B_1 B_0 = A_3 A_2 A_1 A_0 \]
\[ \text{Sh}_1 \text{Sh}_0 = 1 \]
\[ B_3 B_2 B_1 B_0 = A_3 A_3 A_2 A_1 \]
\[ \text{Sh}_1 \text{Sh}_0 = 1 \]
\[ B_3 B_2 B_1 B_0 = A_3 A_3 A_3 A_2 \]
\[ \text{Sh}_1 \text{Sh}_0 = 1 \]
\[ B_3 B_2 B_1 B_0 = A_3 A_3 A_3 A_3 \]

Area dominated by wiring
4-bit Barrel Shifter

Example:

!Sh₁!Sh₀ = 1
B₃B₂B₁B₀ = A₃A₂A₁A₀

!Sh₁Sh₀ = 1
B₃B₂B₁B₀ = A₃A₃A₂A₁

Sh₁!Sh₀ = 1
B₃B₂B₁B₀ = A₃A₃A₃A₂

Sh₁Sh₀ = 1
B₃B₂B₁B₀ = A₃A₃A₃A₃

Area dominated by wiring
4-bit Barrel Shifter Layout

Only one Sh# active at a time

\[ \text{Width}_{\text{barrel}} \sim 2 \ p_m \ N \]

\[ N = \text{max shift distance}, \ p_m = \text{metal pitch} \]

Delay \sim 1 \ \text{fet} + N \ \text{diff caps}
Logarithmic Shifter Structure

<table>
<thead>
<tr>
<th>Sh₀ !Sh₀</th>
<th>Sh₁ !Sh₁</th>
<th>Sh₂ !Sh₂</th>
<th>Sh₃ !Sh₃</th>
<th>Sh₄ !Sh₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>shifts of 0 or 1 bits</td>
<td>shifts of 0 or 2 bits</td>
<td>shifts of 0 or 4 bits</td>
<td>shifts of 0 or 8 bits</td>
<td>shifts of 0 or 16 bits</td>
</tr>
</tbody>
</table>

Data In: 0, 1 shifts, 0, 1, 2, 3 shifts, 0, 1, 2, 3, 4, 5, 6, 7 shifts
Data Out: 0, 1, 2...15 shifts, 0, 1, 2...31 shifts
8-bit Logarithmic Shifter
8-bit Logarithmic Shifter

log N stages
8-bit Logarithmic Shifter Layout Slice

\[ \text{Width}_{\log} \sim p_m(2^K + (1+2+\ldots+2^{K-1})) = p_m(2^K + 2K - 1) \]

\[ K = \log_2 N \]

\[ \text{Delay} \sim K \text{ fets} + 2 \text{ diff caps} \]
## Shifter Implementation Comparisons

<table>
<thead>
<tr>
<th>N</th>
<th>K</th>
<th>Barrel Width</th>
<th>Barrel Speed</th>
<th>Logarithmic Width</th>
<th>Logarithmic Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>$16 p_m$</td>
<td>$1 + 8$</td>
<td>$13 p_m$</td>
<td>$3 + 2$</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>$32 p_m$</td>
<td>$1 + 16$</td>
<td>$23 p_m$</td>
<td>$4 + 2$</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>$64 p_m$</td>
<td>$1 + 32$</td>
<td>$41 p_m$</td>
<td>$5 + 2$</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>$128 p_m$</td>
<td>$1 + 64$</td>
<td>$75 p_m$</td>
<td>$6 + 2$</td>
</tr>
</tbody>
</table>

Barrel shifter needs an $K \times 2^K$ shift amount decoder.
Decoders

- Decodes inputs to activate one of many outputs

Enable

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>In₀</td>
<td>Out₀ = !In₁ &amp; !In₀</td>
</tr>
<tr>
<td>In₁</td>
<td>Out₁ = !In₁ &amp; In₀</td>
</tr>
<tr>
<td></td>
<td>Out₂ = In₁ &amp; !In₀</td>
</tr>
<tr>
<td></td>
<td>Out₃ = In₁ &amp; In₀</td>
</tr>
</tbody>
</table>

- In random gate logic need two inverters, four 2-input nand gates, four inverters plus enable logic
- how about for a 3-to-8, 4-to-16, etc. decoder?
Dynamic NOR Row Decoder
Dynamic NOR Row Decoder

V_{DD} \quad GND \quad GND

!A_0 \quad A_0 \quad !A_1 \quad A_1

Out_0 \quad 1 \rightarrow 0
Out_1 \quad 1 \rightarrow 0
Out_2 \quad 1 \rightarrow 0
Out_3 \quad 1 \rightarrow 1

precharge

0 \rightarrow 1
Dynamic NAND Row Decoder

!A_0  A_0  !A_1  A_1

Out_0  Out_1  Out_2  Out_3

precharge
Dynamic NAND Row Decoder

\begin{align*}
\text{Out}_0 & \ 1 \rightarrow 1 \\
\text{Out}_1 & \ 1 \rightarrow 1 \\
\text{Out}_2 & \ 1 \rightarrow 1 \\
\text{Out}_3 & \ 1 \rightarrow 0
\end{align*}

precharge

0 \rightarrow 1
Building Big Decoders from Small

Active low enable
Active low output

1 → 0 → 1
enable

1x2

2x4

2x4

2x4

2x4

A4
0

A3 A2
0 0

A1 A0
0 1
Multiplexers

- Selects one of several inputs to gate to the single output

\[ \text{Out} = \text{In}_0 \& \neg S_1 \& \neg S_0 | \text{In}_1 \& \neg S_1 \& S_0 | \text{In}_2 \& S_1 \& \neg S_0 | \text{In}_3 \& S_1 \& S_0 \]

- In random gate logic need two inverters, four 3-input nands, one 4-input nand
- how about for an 8x1, 16x1, etc. mux?
Review: TG 2x1 Multiplexer

F = !((In₁ & S) | (In₂ & S))
Building Big Muxes from Small

```
A0 \rightarrow 2x1 \rightarrow \rightarrow A1
A1 \rightarrow 2x1 \rightarrow \rightarrow A2
A2 \rightarrow 2x1 \rightarrow \rightarrow A3
\hspace{1cm} S_0 \hspace{1cm} S_1
\hspace{1cm} 2x1 \hspace{1cm} 2x1
\hspace{1cm} \rightarrow \rightarrow Out
```
Building Big Muxes from Small
Review: Datapath Bit-Sliced Organization

Tile identical bit-slice elements
Layout of Bit-Sliced Datapaths
Layout of Bit-sliced Datapaths

Without feedthroughs or pitch matching (4.2\(\mu\)m\(^2\))

With feedthroughs (3.2\(\mu\)m\(^2\))

With feedthroughs and pitch matching (2.2\(\mu\)m\(^2\))
Alpha 21264 Integer Unit Datapath
Next Lecture and Reminders

Next lecture

- Semiconductor memories
  - Reading assignment – Rabaey, et al, 12.1-12.2.1

Reminders

- HW#5 will (optional) due November 20^{th}
- Project final reports due December 4^{th}
- Final grading negotiations/correction (except for the final exam) must be concluded by December 10^{th}
- Final exam scheduled
  - Tuesday, December 16^{th} from 10:10 to noon in 118 and 113 Thomas