Combinational Datapath

James C. Hoe
Dept of ECE, CMU
April 14, 2015

Today’s Goal: Structured combinational digital circuits
Announcements: Read Rizzoni 12.4 and 12.5
Read Rizzoni 12.6 for next time
Exam 3 on April 30
Final Exam, Fri., May 8, 8:30~11:30, GHC 4401
HW9 due Tuesday 4/21
No class on Thursday; no lab this week

Combinational Logic Blocks

- With K-map, you can build any combinational functions you want, but not everything should be build from K-map
  - minimum SOP/POS ≠ globally “optimum”
  - some functions have special structures to be taken advantage of
  - some functions are so frequently used they have become conventions
- Major examples
  - multiplexer
  - demultiplexer
  - encoder/decoder of various types
  - arithmetic: adder, multiplier, etc
  - read-only memory
Multiplexer (aka mux)

- Simplest form is a 1-bit 2-to-1 multiplexer
  
  $\begin{array}{c|cc|c}
  S & D_0 & D_1 & O \\
  \hline
  0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 1 & 0 & 1 \\
  0 & 1 & 1 & 1 \\
  1 & 0 & 0 & 0 \\
  1 & 0 & 1 & 1 \\
  1 & 1 & 0 & 0 \\
  1 & 1 & 1 & 1 \\
  \end{array}$

- This is the hardware “if-then-else”; O gets either $D_0$ or $D_1$ depending on $S$

4-to-1 Mux Example

You don’t get this by K-map
Multiplexer Generalizations

- N-to-1 selection using
  - $S$ of $\log_2 N$ bits as a binary number
  - $S$ of $N$ bits as a “one-hot” bit mask (this is called a decoded mux)

- $D_x$ and $O$ could be a bundle of wires of width $w$ (aka a bus), e.g., to carry a binary number

Demultiplexer (aka demux)

- Example: 1-bit 1-to-2 demultiplexer

- Generalizable to 1-to-$N$ demux’ing

- Does it also make sense to demux a bus?

if $D=1$, $O$ is 1-hot of $S$
1-to-4 Demux Example

Encoders/Decoders

- Mapping between well-known representations, e.g.,
  - between binary and 1-hot
  - between BCD and binary
  - from BCD to 7-segment display
- Mapping input to well-known output functions
  - e.g., priority encoder
    - input: 1-bit inputs from N requesters
    - output: \( n=\log_2 N \) bit binary value that chooses the lowest number'ed requester asserting \( R \)

What is \( S \) if no one is requesting?
(Unsigned) Binary Addition

- How to add A and B, each 4 bits, by long hand

```
\begin{align*}
\text{b}_3 & \text{ a}_3 \\
\text{b}_2 & \text{ a}_2 \\
\text{b}_1 & \text{ a}_1 \\
\text{b}_0 & \text{ a}_0 \\
\text{c}_4 & \text{c}_3 \\
\text{c}_2 & \text{c}_1 \\
\text{c}_0 & \text{c}_3 \\
\text{s}_3 & \text{s}_2 \\
\text{s}_1 & \text{s}_0 \\
carry? & \\
\end{align*}
```

Full Adder

```
\begin{array}{c|ccc|cc}
\text{c}_{in} & \text{a} & \text{b} & \text{c}_{out} & \text{s} \\
\hline
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
```

\[ s = a \oplus b \oplus c_{in} \]
\[ c_{out} = bc_{in} + ac_{in} + ab \]

3-way majority parity

a, b, c_{in} are functionally indistinguishable as inputs
Unsigned Binary Addition

Overflow? = if (a_3 \oplus b_3) then false else (a_3 \oplus s_3)
- can’t overflow when adding a pos. and a neg. number
- if 2 pos. numbers yield a neg. number \Rightarrow V; vice versa
2’s-Complement Subtraction

- Subtracting is like adding the negative
- Negation is easy in a 2’s-complement representation

How do you build a comparator (i.e., >, <)?

Cost and Speed of an n-bit “Ripple-Carry” Adder

- Cost is \( n \times \text{SizeOf(Full Adder)} \)
- \( S \) and \( C_{\text{out}} \) do not change instantaneously when \( a \) and \( b \) are changed
  - Longest delay (aka Critical Path Delay) is from \( a_0, b_0, \) or \( c_{\text{in}} \) to \( s_{n-1} \) or \( c_{\text{out}} \)
  - \( n \times \text{DelayOf(Full Adder)} \)
  - \( n \times 2 \) gate delays (assuming 2-level SOP is used)

BTW, ripple-carry is adder design for babies . . . .
Prefix Carry-Look-Ahead

Example: 8-bit, 2-ary CLA

\[
\begin{align*}
c_{in0} &= Cin \\
c_{in1} &= g_0 + p_0 \cdot Cin \\
c_{in2} &= G10 + P10 \cdot Cin \\
c_{in4} &= G30 + P30 \cdot Cin \\
c_{out} &= G70 + P70 \cdot Cin \\
c_{in3} &= g_2 + p_2 \cdot c_{in2} \\
c_{in5} &= g_4 + p_4 \cdot c_{in4} \\
c_{in6} &= G54 + P54 \cdot c_{in4} \\
c_{in7} &= g_6 + p_6 (G54 + P54 \cdot c_{in4})
\end{align*}
\]

Read-Only Memory (ROM)

- Input: an \( n \)-bit address
- Output: data of width \( w \)
  stored at location \( \text{ROM}[\text{addr}] \) (\( 2^n \) such locations)
- Since it is read-only, the ROM’s contents have to be initialized by an offline process
- Since it is read-only, a ROM is combinational!!
  - the same \( \text{addr} \) always returns the same answer
  - an ROM is an \( n \)-input combinational function; each address is a minterm; reading \( \text{ROM}[\text{addr}] \) is like reading the truth table
  - it is actually a reasonable way to implement some very irregular combinational functions
Let’s refresh our memory....

- All “sequential” digital systems comprise
  - input and output
  - state stuff that remembers
  - “combinational” stuff that computes a function (has no memory)
- In an execution, state is updated to a “next state” based on a function of the current state

Memory 101

- Pure combinational logic always go from a current input to a current output without any looping back
  - There is no notion of time, past or future
- To remember, must somehow incorporate previous values
- You mean like this?

Does it remember? What does it remember?
(It is easier to see if you associate a small propagation delay with wires and gates)