
IC220

Slide Set #8: Digital Logic Finale (Appendix B)

1

Big Picture

- Computer Overview (Chapter 1)
- A specific instruction set architecture (Chapter 2)
- Logic Design (Appendix B)
- Arithmetic and how to build an ALU (Chapter 3)
- Performance issues (Chapter 4)
- Constructing a processor to execute our instructions (Chapter 5)
- Pipelining to improve performance (Chapter 6)
- Memory: caches and virtual memory (Chapter 7)
- I/O (Chapter 8)
- A few advanced topics

3

ADMIN

- Project 1 due Mon Feb 5
 - Recall – No collaboration – start early & see instructor for help
- READING
 - Appendix: Read B.7,B.8,B.9, B.10, and B.12.
(skip the Verilog details).
- Course Paper *description* due by Feb 26 for approval
 - Current computer architectural topic/issue
 - 3-5 pages
 - Suggested topics on course calendar – but a topic alone is not a description! (see online instructions)
- 6 week exam, in class, Wed February 14

2

“Real World” Example

- Buzzer Feature for a Car
- Should Buzz when
 1. the engine is on, the door is closed, and the seat belt is unbuckled
 2. the engine is on, the door is open
- What are our input(s)?

- What are our output(s)?

4

(extra space)

Check Yourself

- Could you have filled in the truth table?
- Could you have filled in the K-Map?
- Can you use the K-Map to minimize the equation?
- Can you draw the circuit?

6

Bigger Units of Combinational Logic

- Gates useful but fairly low level
- Easier to construct circuits with higher-level building blocks instead:
 - **Combinational Logic**
 - Multiplexors (mux)
 - Decoders
 - **(later) Sequential Logic**
 - Registers
 - Arithmetic unit (ALU)
- What is this an example of?

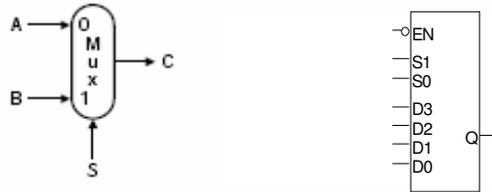
7

Multiplexor – Example Usage



8

Multiplexor – 1-bit version



- Think of a mux as a selector
- S selects one input to be the output
- N-way mux has
 - # inputs:
 - # selector lines (S):
 - # outputs:
- Implementation?

9

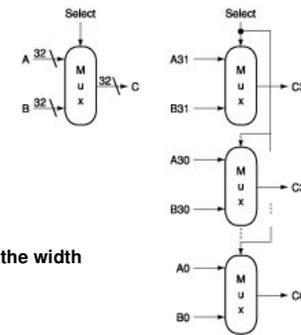
End of Combinational Logic

11

Multiplexor – Wider version

EX: B-31 to B-32

- 32 bit wide, 2-way Mux:



- Pictures don't always show the width (especially if 32 bits)

10

Combinational vs. Sequential Logic

- Combinational Logic – output depends only on
- Sequential Logic – output depends on:
 - Previous inputs are stored in “state elements”
 - _____ determines when an element is updated
 - State elements will involve use of feedback in circuit
 - Not permitted in combinational circuits

12

Truth Tables → Next State Tables

- New kind of input:

A	B	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

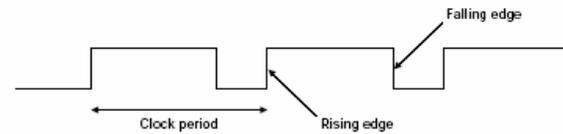
13

Clocks and State Elements

- Clock Frequency is the _____ of _____.
- When should updates occur to state elements?

– Edge – change state when

– Level – change state when

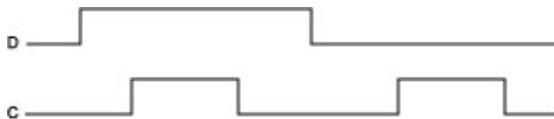
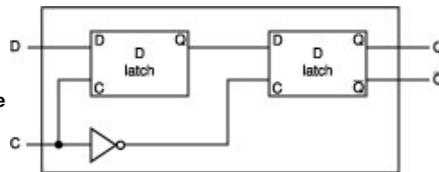


14

D-Type Flip Flop

EX: B-41

- State only changes
- Otherwise... remembers previous state
- Abstraction:



Q-flipflop

15

State Diagrams

- State = Contents of memory
- Diagrams are a tool to represent ALL transitions from one state to another
 - What causes state changes?
- Example for D Flip-Flop:

NEXT STATE TABLE

D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

RESET

SET



16

Finite State Machines

- Can use state diagrams to express more complex sequential logic.
- Example: Candy Machine
 - Inputs: N (nickel received), D (dime received)
 - Outputs: C (dispense candy), R (give refund)
 - Should dispense candy after 15 cents deposited, + refund if overpaid. Then await next customer.
- We'll use *Moore machine* – output depends only on
- What states do we need?

17

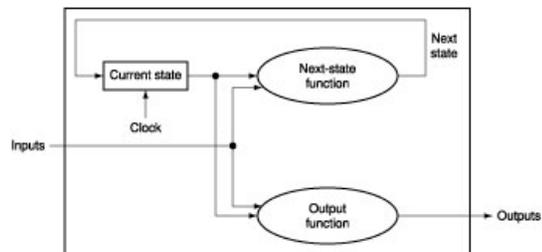
Example: Candy Machine

EX: B-51 to B-53

Inputs: (N)ickel, (D)ime
Outputs: (C)andy, (R)efund

18

Implementing Finite State Machines



- Squares =
- Circles =
- We don't always show the clock for registers/memory diagrams, but will be implicit

19

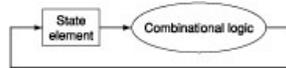
FSM Example

20

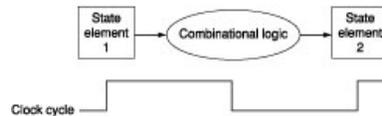
Combining Combinational and Sequential Logic

- Finite State Machine was our first example of this
- Two general patterns:

1. State Machine



2. Pipeline



- In either case, have important timing concerns
 - Output of combinational logic block may oscillate before settling
 - Clock cycle time must be long enough so combo-logic settles before the sequential logic (state) reads the new value
 - State elements ensure that combo-logic inputs remain stable

21

Registers and Register Files

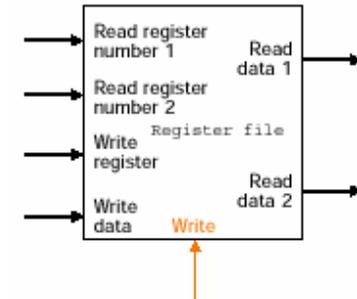
- Registers store data (bits) (i.e. have memory)
 - Each register =

- Register files contain:
 - Set of registers
 - Logic for read/write

- MIPS register file has how many registers?

- How does it store data?

- How does it know which register to access?



22

Memory

- Why so many types?
- Basic types:
 - RAM “random access memory” (read/write)
 - Main memory
 - Volatile
 - Types:
 - SRAM – async, sync, pipeline burst, cache;
 - DRAM – M, FPM, EDO, burst EDO, sync, DR, DDR
 - ROM (read only)
 - Small
 - Stores critical operating instruction (BOOT strap)
 - Non-volatile
 - Common in embedded system (toys, cameras, printers, etc)
 - Types: PROM, EPROM, EEPROM, flash memory

23

Appendix B Summary

- Truth tables and Gates
 - AND, OR, NOT, NOR, NAND, XOR
- Boolean Algebra
 - Distributive, DeMorgan’s, Inverse, Identity, etc
- Combinational Logic
 - Circuits – Design, reduction / minimization, K-maps
 - Multiplexor
- Sequential Logic
 - Flip/flops
 - Clock & state diagrams
- Register files
- Memory
 - RAM vs ROM, SRAM vs. DRAM

24