
IC220

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

1

“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)?

3

ADMIN

- Project 1 due Thurs Feb 19
 - Recall – No collaboration – start early & see instructor for help
- READING
 - Appendix: Read C.7-C.10, and C.12.
(skip the Verilog details).
- Course Paper *description* due by Thurs Feb 26 for approval (email)
 - 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, Thurs February 12

- NO homework on this part of Appendix B – just in-class exercises
 - Will be related lab and project

2

(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?

5

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?

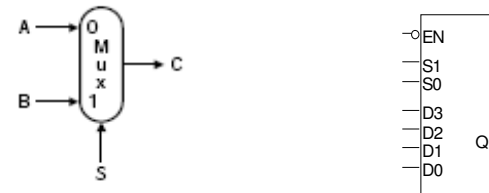
6

Multiplexor – Example Usage



7

Multiplexor – 1-bit version



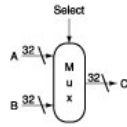
- Think of a mux as a selector
- S selects one input to be the output
- N-way mux has
 - # inputs:
 - # selector lines (S):
 - # outputs:
- Implementation?

8

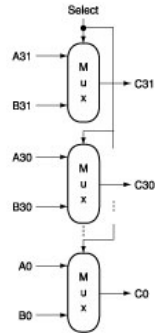
Multiplexor – Wider version

EX: B-31 to B-32

- 32 bit wide, 2-way Mux:



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



9

(5 pts) Exercise B-32

- Draw an 8-input mux with inputs: A, B, C, D, E, F, G, H and output: OUT (Remember to draw the selector bits)
(you don't need to draw the internals, just the external view)

11

(5 pts) Exercise B-31

- A. A 8-way mux has _____ "inputs", _____ selector bit(s), and _____ output(s)

10

End of Combinational Logic

12

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

13

Truth Tables → Next State Tables

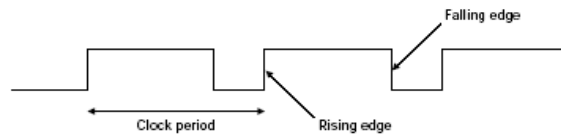
- New kind of input:

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

14

Clocks and State Elements

- Clock Frequency is the _____ of _____.
- When should updates occur to state elements?
 - Edge – change state when
 - Level – change state when

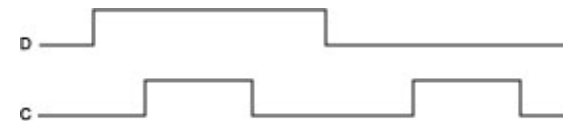
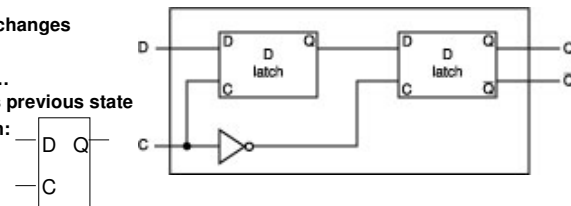


15

D-Type Flip Flop

EX: B-41

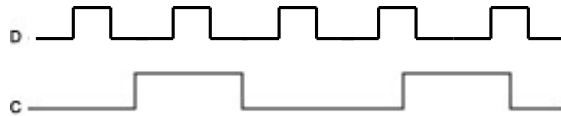
- State only changes
- Otherwise... remembers previous state
- Abstraction:



Q-flipflop

16

(5 pts) Exercise B-41 – Complete the timing diagram below



Q-FlipFlop
(falling edge triggered)

Q-FlipFlop
(rising edge triggered)

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?

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



Example: Candy Machine

EX: B-51 to B-53

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

(5 pts) Exercise B-51

- Draw a state diagram for the following next state function:
- How would you describe what input 'A' is accomplishing?

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

21

(10 pts) Exercise B-52

- John and Mary agree to play rock-paper-scissors to decide who has to pay for dinner. The overall winner will be whoever wins two rounds in a row.
- Assume you have 6 inputs:
 - JR, JP, JS (only one true depending on if John plays rock, paper, or scissors)
 - MR, MP, MS
- At each round,
 1. If John and Mary play the same (both scissors, etc.), then the game returns to the initial state.
 2. If either John or Mary has just won twice in a row, the next state should be a "Game over" state.
 3. Otherwise, the next state should reflect who won the most recent round

Your task:

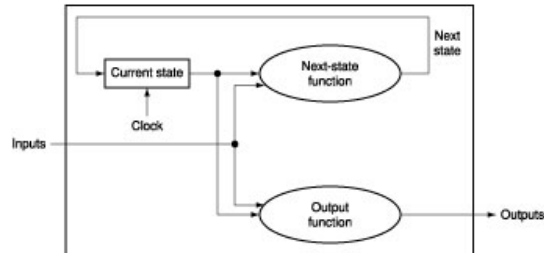
1. How many different states do you need?
2. Draw the next state diagram for this game

Of course:

Rock beats scissors
Paper beats rock
Scissors beats paper

23

Implementing Finite State Machines



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

25

FSM Example

26

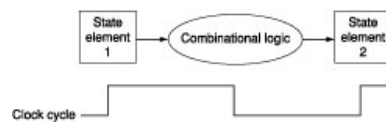
Combining Combinational and Sequential Logic

- 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

27

Registers and Register Files

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

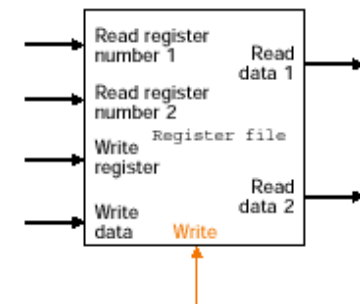
• Register files contain:

- 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?



28

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

29

Appendix C 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

30