

Penn Engineering **ESE**

Lecture #14 – Sequential Logic, FPGAs

ESE 150 – DIGITAL AUDIO BASICS

ESE150 Spring 2021

Based on slides © 2009–2021 DeHon

ESE150 Spring 2021

LECTURE TOPICS

- × Setup
- × Where are we?
- × Review: Combinational Logic
- × Sequential Logic
- × FPGAs
- × Next Lab

2

ESE150 Spring 2021

COURSE MAP – WEEK 8

Music (1) → sample (2) → freq (4) → domain conversion (5,6) → psycho-acoustics → compress (3) → A/D → 10101 → logic gates → D/A ← 10101001101 → speaker

Numbers correspond to course weeks

EULA, click OK, MP3 Player / iPhone / Droid

3

ESE150 Spring 2021

COMBINATIONAL LOGIC

4

ESE150 Spring 2021

GATE

- × **Primitive binary function**
 - + Computes a binary output from a small number of binary inputs
- × **Can specify function with a Truth Table**
 - + Defines the output for each input combination

Input 0	Input 1	Output
0	0	
0	1	
1	0	
1	1	

5

ESE150 Spring 2021

CONCLUDE

- × **Can implement any combinational logic function out of a collection of**
 - + OR2, AND2, NOT gates

6

ESE150 Spring 2021

ARITHMETIC

- × **Addition is also a digital logic function**
 - + Maps set of inputs ($a_3 a_2 a_1 a_0 b_3 b_2 b_1 b_0$)
 - + To an output bit vector ($c_4 c_3 c_2 c_1 c_0$)
- × ...as is subtraction, multiplication, division, square root....

7

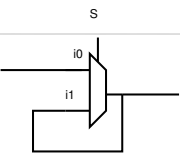
ESE150 Spring 2021

SEQUENTIAL LOGIC

8

ESE150 Spring 2021

MUX WITH FEEDBACK

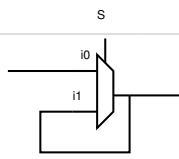


- × **What happens when S=0?**
- × **What happens when S=1?**

9

ESE150 Spring 2021

MUX WITH FEEDBACK

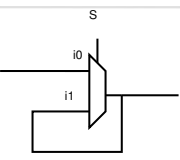


- × **Assuming i0 doesn't change what happens when S goes from 0 to 1?**

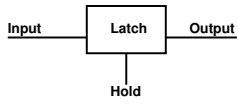
10

ESE150 Spring 2021

LATCH



- × **Element that can hold a previous value of an input**

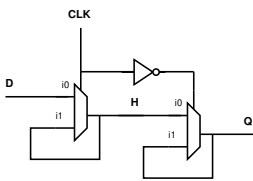
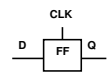


11

ESE150 Spring 2021

FLIP-FLOP (FF)

- × **Use a pair to create a flip-flop**
 - + Also call register

12

ESE150 Spring 2021

FLIP-FLOP (FF)

- × Use a pair to create a flip-flop
 - + Also call register
- × What happens when
 - + CLK is low (0) ?

13

ESE150 Spring 2021

FLIP-FLOP (FF)

- × Use a pair to create a flip-flop
 - + Also call register
- × What happens when
 - + CLK is high (1) ?

14

ESE150 Spring 2021

FLIP-FLOP (FF)

- × Use a pair to create a flip-flop
 - + Also call register
- × What happens when
 - + CLK transitions from 0 to 1?

15

ESE150 Spring 2021

FLIP-FLOP (FF)

- × Use a pair to create a flip-flop
 - + Also call register
- × Sample D input on 0→1 transition of clock (CLK)
- × Never an open path from D→Q
 - + One of the mux latches always in hold state

16

ESE150 Spring 2021

STATE ELEMENT

- × Latch or Register is a state element
- × Allows circuit to remember a value
- × Build computations that
 - + Depend on past inputs
 - + Reuse hardware in time

17

ESE150 Spring 2021

ACCUMULATOR REVISITED

- × Maybe extend accumulator bits to hold larger sum
- × Maybe more...

18

ESE150 Spring 2021

STATE FOR SEQUENCING AND CONTROL

- × Useful when trying to control things
 - + E.g. Perform a sequence of operations
- × Robot
 - + Open-gripper
 - + Move-forward
 - + Close-gripper
 - + Lift

19

ESE150 Spring 2021

STATE FOR CONDITIONAL CONTROL

- × Useful when need to behave differently based on something in the past
 - + Remember if elevator going up or down
 - + Remember/count coins from consumer
 - + Remember some mode set by user
 - × Displaying in Centigrade or Fahrenheit
- × Idea
 - + Store state
 - + Use as input to logic

20

ESE150 Spring 2021

FINITE-STATE MACHINE (FSM)

- × Sequential model of computation
- × State (in registers) + combinational logic
- × Compute outputs and next state from inputs and state

21

ESE150 Spring 2021

FSM EXAMPLE

- × Simplified Vending Machine
 - + Only input quarters
 - + Only vend one item (output signal to indicate vending)
 - + Item costs 2 quarters
 - + Coin Return request and control
- × Two states: waiting, one-quarter (one)
- × Two inputs: quarter, coin-return (creturn)
- × Two outputs: vend, return-quarter (qreturn)

22

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together

state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1			one
waiting	1	0	0	0	one
waiting	1	1			
one	0	0			
one	0	1			
one	1	0			
one	1	1			

23

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together

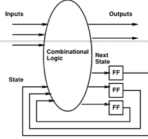
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1			
one	0	0			
one	0	1			
one	1	0			
one	1	1			

24

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together



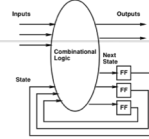
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0			
one	0	1			
one	1	0			
one	1	1			

25

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together



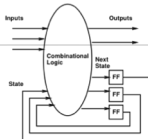
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0	0	0	one
one	0	1			
one	1	0			
one	1	1			

26

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together



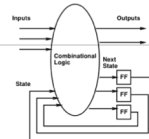
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0	0	0	one
one	0	1	0	1	waiting
one	1	0			
one	1	1			

27

ESE150 Spring 2021

TRUTH TABLE MODEL

Complete together

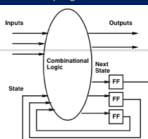


state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0	0	0	one
one	0	1	0	1	waiting
one	1	0	1	0	waiting
one	1	1			

28

ESE150 Spring 2021

TRUTH TABLE MODEL



state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0	0	0	one
one	0	1	0	1	waiting
one	1	0	1	0	waiting
one	1	1	0	1	one

29

ESE150 Spring 2021

SWITCH-STATEMENT MODEL

```

x While (true)
x   switch (state) {
x     case waiting:
x       if (quarter && !creturn)
x         state=one;
x       else
x         state=waiting;
x       qreturn=quarter && creturn;
x       vend=0;
x       break;
  
```

30

ESE150 Spring 2021

SWITCH-STATEMENT MODEL (CONT.)

```

x   case one:
x     if ((quarter && !creturn)||
x       (!quarter&&creturn))
x       state=waiting;
x     else
x       state=one;
x     qreturn=creturn;
x     vend=quarter&& !creturn;
x     break;
x   } // switch
x } // while
    
```

31

ESE150 Spring 2021

FSM GRAPH MODEL

```

graph TD
    waiting((waiting))
    one((one))
    waiting -- "quarter/creturn=0, vend=0, qreturn=0" --> waiting
    waiting -- "quarter&creturn/creturn=1, vend=0, qreturn=1" --> one
    one -- "quarter&!creturn/creturn=0, vend=0, qreturn=0" --> waiting
    one -- "quarter&creturn/creturn=1, vend=0, qreturn=1" --> one
    
```

32

ESE150 Spring 2021

PROGRAMMABLE LOGIC

33

ESE150 Spring 2021

PRECLASS 2

- How build 4-input mux from 2-input muxes?

34

ESE150 Spring 2021

PRECLASS 3

- What function of s0, s1 is this circuit configuration computing?

35

ESE150 Spring 2021

MUX CAN BE A PROGRAMMABLE GATE

- Programmable Gate**
 - Can be programmed to act as any gate
 - Use state (e.g. FF) to "program" truth table of a gate

Input 0	Input 1	Output
0	0	
0	1	
1	0	
1	1	

36

ESE150 Spring 2021

EXAMPLE: OR (PRECLASS 4)

× How do we program to behave as OR2?

37

ESE150 Spring 2021

LOOK-UP TABLE (LUT)

× Can generalize to any number of inputs

38

ESE150 Spring 2021

CONNECTING GATES

× Once we can build gates
 × ...still need to connect the gates together.
 × Select which gate outputs become inputs to other gates.

39

ESE150 Spring 2021

MUX CAN BE PROGRAMMABLE INTERCONNECT

Trick: Use multiplexer to programmably select gate input.

40

ESE150 Spring 2021

PROGRAMMABLE BLOCKS

41

ESE150 Spring 2021

PROGRAMMABLE GATES AND INTERCONNECT

If time permits:
 How program (fill in yellow programmable cells) to implement a full adder?

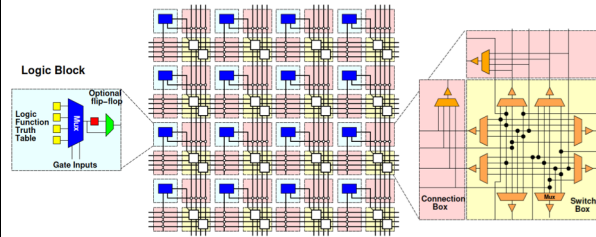
42

FIELD-PROGRAMMABLE GATE ARRAY (FPGA)

- × **Collection of Programmable Gates**
 - + Can “program” by setting state bits
 - + LUTs that can be programmed to be any gate
 - × With optional Flip-Flops to use for state
 - + Programmable interconnect to “wire” the gates together

43

FIELD-PROGRAMMABLE GATE ARRAY (FPGA)



44

BIG IDEAS

- × **Can implement any combinational digital logic function from nand2 gates**
- × **Can implement any FSM from nand2 gates and registers**
- × **Can build a single chip that can be programmed to behave as any collection of gates**
 - + As long as don't need more gates than it provides

45

LEARN MORE

- × **CIS240 – do a bit more logic**
- × **ESE370 – how to implement gates, latches, and memories from transistors**
- × **ESE532 – how to build large-scale computations from logic**

46

REMINDER

- × **Feedback**
- × **Lab 7 due on Friday**

47