

Penn Engineering **ESE**

Lecture #7 – Digital Logic

ESE 150 – DIGITAL AUDIO BASICS

ESE150 Spring 2018

Based on slides © 2009–2018 DeHon

ESE150 Spring 2018

SO FAR

MIC → A/D → 10101001101

Music (1) → sample (2) → domain conversion (5,6) → freq (4) → psycho-acoustics (5,6) → compress (3)

DFT Identify Masking Huffman encoding

Huffman Decode IDFT

EULA click OK → speaker ← D/A ← 10101001101

MP3 Player / iPhone / Droid

2

ESE150 Spring 2018

HOW PROCESS

- × How do we build a machine to perform these operations?
 - + From Digital Samples → compressed digital data → Digital Samples
- × Down to bottom
 - + If we can build **one** kind of primitive element,
 - × ...and connect together large collections of them
 - + can build a machine to perform *any* digital computation

3

ESE150 Spring 2018

LECTURE TOPICS

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

4

ESE150 Spring 2018

COURSE MAP

MIC → A/D → CPU → File-System (10) → NIC → Cloud (11)

Music (1) → sample (2) → domain conversion (5,6) → freq (4) → psycho-acoustics (5,6) → compress (3)

7,8,9

10101001101

10101001101

13

EULA click OK → speaker ← D/A ← 10101001101

MP3 Player / iPhone / Droid

12

5

ESE150 Spring 2018

COURSE MAP – WEEK 8

MIC → A/D → 10101

Music (1) → sample (2) → domain conversion (5,6) → freq (4) → psycho-acoustics (5,6) → compress (3)

10101

EULA click OK → speaker ← D/A ← 10101001101

MP3 Player / iPhone / Droid

6

ESE150 Spring 2018

COMBINATIONAL LOGIC

7

ESE150 Spring 2018

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	

8

ESE150 Spring 2018

AND GATE

- × **AND**
 - + Output is 1 (true) when all inputs are 1 (true)




Input 0	Input 1	AND
0	0	0
0	1	0
1	0	0
1	1	1

9

ESE150 Spring 2018

NOT GATE

- × **Not**
 - + Output is opposite of input




Input	NOT
0	1
1	0

10

ESE150 Spring 2018

OR GATE

- × **OR**
 - + Output is 1 (true) when any input is 1 (true)
 - + (fillin truth table for OR)



Input 0	Input 1	OR
0	0	
0	1	
1	0	
1	1	

11

ESE150 Spring 2018

CLAIM

- × **Can compute any Boolean Function from AND, OR, NOT**
 - + (actually from NAND)

12

ESE150 Spring 2018

MODEL: COMBINATIONAL LOGIC


- × **Compute some “function”**
 - + $f(i_0, i_1, \dots, i_n) \rightarrow o_0, o_1, \dots, o_m$
- × **Each unique input vector**
 - + implies a particular, deterministic, output vector

13

ESE150 Spring 2018

BIG AND

- × **AND**
 - + Output is 1 (true) when all inputs are 1 (true)
- × **How built n-input AND from AND2 gates?**




14

ESE150 Spring 2018

BIG OR

- × **OR**
 - + Output is 1 (true) when any input is 1 (true)
- × **How build n-input OR from OR2?**



15

ESE150 Spring 2018

INPUT CONDITION

- × **How can we create an expression that is true for a specific input case?**
 - + E.g. have a function of 4 inputs: a, b, c, d
- × **How many potential values for a, b, c, d?**
 - + Rows in our truth table
- × **Give one example of values for a, b, c, d?**
- × **How create an expression that is true for that case?**

16

ESE150 Spring 2018

SINGLE OUTPUT DIGITAL FUNCTION

- × **Given have logic to implement each input case**
- × **How implement entire function?**

a	b	c	F(a,b,c)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

17

ESE150 Spring 2018

MULTIPLE OUTPUT FUNCTION

- × **What do you do if your Digital Function needs multiple output bits?**

18

ESE150 Spring 2018

COMBINATIONAL LOGIC AS GATES

- ✦ **Start with truth table**
- ✦ **Single output {0, 1}**
 - + Use inverters to produce complements of inputs
 - + For each input case
 - ✦ If output is a 1
 - Develop an AND to detect that case
 - Decompose AND into gates
 - + OR together the output of all such AND functions
 - Decompose OR into gates
- ✦ **Multiple outputs**
 - + Repeat for each output

This solution won't typically be the smallest or fastest...

19

ESE150 Spring 2018


CONCLUDE

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

20

ESE150 Spring 2018

NAND2 GATE



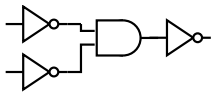
- ✦ **NAND = NOT AND**
 - + Output is 0 (true) when all inputs are 1 (true); 0 otherwise

Input 0	Input 1	NAND
0	0	1
0	1	1
1	0	1
1	1	0

21

ESE150 Spring 2018

NAND UNIVERSALITY

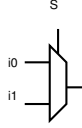


- ✦ **Can implement**
 - + AND2 from NAND2
 - + NOT from NAND2
 - + OR2 from NAND2
- ✦ **Can implement any combinational logic function out of a collection of**
 - + OR2, AND2, NOT gates
- ✦ **Therefore: Can implement any combinational logic function out of a collection of NAND2 gates**

22

ESE150 Spring 2018

MULTIPLEXER GATE



- ✦ **MUX**
 - + When S=0, output=i1
 - + When S=1, output=i0

S	i0	i1	Mux2(S,i0,i1)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Truth Table?
AND, OR, NOT Implementation?

23

ESE150 Spring 2018

ARITHMETIC

- ✦ **Addition is also a digital logic function**
 - + Maps set of inputs (a3 a2 a1 a0 b3 b2 b1 b0)
 - + To an output bit vector (c4 c3 c2 c1 c0)
- ✦ **...as is subtraction, multiplication, division, square root....**

24

ESE150 Spring 2018

FULL ADDER

- × Adds 3 inputs to produce 2b output
 - + Can produce truth table and logic (Lab)

25

ESE150 Spring 2018

N-BIT ADDER

- × Given Full Adders
 - + Can build N-bit adder by connecting N full adders

26

ESE150 Spring 2018

SEQUENTIAL LOGIC

27

ESE150 Spring 2018

MUX WITH FEEDBACK

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

28

ESE150 Spring 2018

MUX WITH FEEDBACK

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

29

ESE150 Spring 2018

LATCH

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

30

ESE150 Spring 2018

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

31

ESE150 Spring 2018

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

32

ESE150 Spring 2018

ACCUMULATOR

- × What does this do?

33

ESE150 Spring 2018

ACCUMULATOR

- × while (true)
 - + a=a+getInput();
- × Accumulates input values i
 - + Integration or summation

34

ESE150 Spring 2018

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

35

ESE150 Spring 2018

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

36

ESE150 Spring 2018

FINITE-STATE MACHINE (FSM)

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

37

ESE150 Spring 2018

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)**

38

ESE150 Spring 2018

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

39

ESE150 Spring 2018

SWITCH-STATEMENT MODEL

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

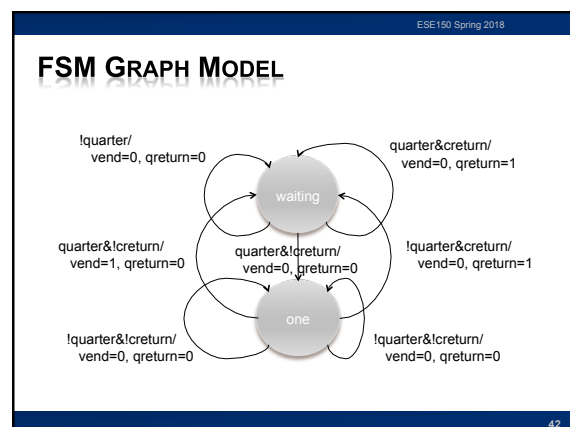
40

ESE150 Spring 2018

SWITCH-STATEMENT MODEL (CONT.)

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

41



ESE150 Spring 2018

PROGRAMMABLE LOGIC

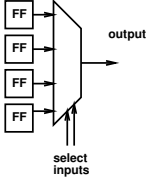
43

ESE150 Spring 2018

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	

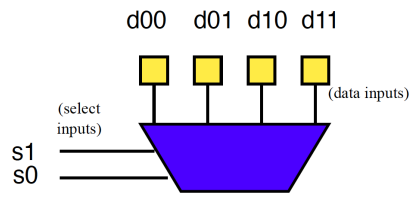


44

Penn ESE532 Fall 2017 - DeHon

EXAMPLE: AND

- × **How do we program to behave as AND2?**

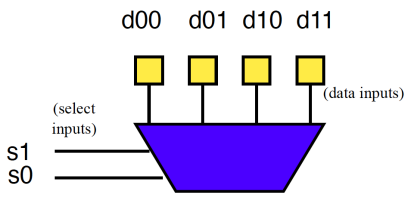


45

Penn ESE532 Fall 2017 - DeHon

EXAMPLE: OR

- × **How do we program to behave as OR2?**

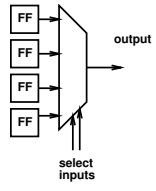


46

ESE150 Spring 2018

LOOK-UP TABLE (LUT)

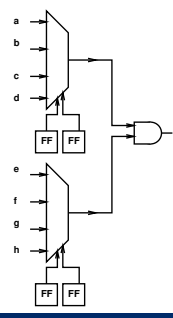
- × **Can generalize to any number of inputs**



47

ESE150 Spring 2018

MUX CAN BE PROGRAMMABLE INTERCONNECT



48

ESE150 Spring 2018

PROGRAMMABLE BLOCKS

49

ESE150 Spring 2018

PROGRAMMABLE GATES AND INTERCONNECT

50

ESE150 Spring 2018

FIELD-PROGRAMMABLE GATE ARRAY

- × **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

51

Penn ESE532 Fall 2017 - DeHon

FIELD-PROGRAMMABLE GATE ARRAY (FPGA)

52

ESE150 Spring 2018

NEXT LAB

- × **Program an FPGA in Verilog**
 - + Build an adder
 - + Build an FSM

53

ESE150 Spring 2018

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

54

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**

REMINDER

- × **Formal Lab Report Due Sunday**