

Lecture #13 – Combinational Logic

ESE 1500 – DIGITAL AUDIO BASICS

ESE1500 Spring 2023

Based on slides © 2009–2023 DeHon

1

SO FAR

MUSIC → MIC → A/D → 10101001101

10101001101 → domain conversion (sample) → freq → pyscho-acoustics → compress → Huffman encoding

compress → Huffman Decode → IDFT → D/A → 10101001101 → speaker

Numbers correspond to course weeks

EULA, click OK, MP3 Player / iPhone / Droid

2

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 (maybe 2),
 - × ...and connect together large collections of them
 - + can build a machine to perform *any* digital computation

3

LECTURE TOPICS

- × Setup
- × Where are we?
- × Combinational Logic
- × Arithmetic (Part 2)
- × Accumulator (Part 3)
- × Next Lab

4

COURSE MAP

MUSIC → MIC → A/D → CPU → File-System → NIC → Cloud

Cloud → NIC → CPU → D/A → speaker

Numbers correspond to course weeks

EULA, click OK, MP3 Player / iPhone / Droid

5

COURSE MAP - WEEK 8

MUSIC → MIC → A/D → 10101 → Logic Gate → D/A → speaker

Numbers correspond to course weeks

EULA, click OK, MP3 Player / iPhone / Droid

6

ESE1500 Spring 2023

COMBINATIONAL LOGIC

7

7

ESE1500 Spring 2023

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

8

ESE1500 Spring 2023

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

9

ESE1500 Spring 2023

NOT GATE

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



input	NOT
0	1
1	0


10

10

ESE1500 Spring 2023

OR GATE

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



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

11

11

ESE1500 Spring 2023

CLAIM

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

12

12

ESE1500 Spring 2023

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

13

ESE1500 Spring 2023

BIG AND

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




14

14

ESE1500 Spring 2023

BIG OR

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



15

15

ESE1500 Spring 2023

INPUT CASE

- × **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
 - × Want identify case a=0, b=1, c=1, d=0
- × **How many potential values for a, b, c, d?**
 - + Rows in our truth table
- × **How create an expression (in and and not) that is true for the a=0, b=1, c=1, d=0 case?**

16

16

ESE1500 Spring 2023

SINGLE OUTPUT DIGITAL FUNCTION

- × **Given have logic to identify each input case where output is a 1:**
- × **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

17

ESE1500 Spring 2023

MULTIPLE OUTPUT FUNCTION

- × **What if your Digital Function needs multiple output bits?**
 - + That is: how extend to handle multiple outputs?

18

18

ESE1500 Spring 2023

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

19

ESE1500 Spring 2023

CONCLUDE

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

20

20

ESE

Penn Engineering

Part 2

ARITHMETIC

21

21

ESE1500 Spring 2023

ARITHMETIC

- × **Addition is also a digital (combinational) 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....**

22

22

ESE1500 Spring 2023

POSITIONAL BINARY NUMBERS

- × Binary number: $a_{n-1} a_{n-2} \dots a_0$
 - + Eg., 01101
 - + $n=5$ $a_4=0$, $a_3=1$, $a_2=1$, $a_1=0$, $a_0=1$
- × **value** = $\sum_{i=0}^{i<n} a_i \times 2^i$
- × **What value 01101?**

23

23

ESE1500 Spring 2023

ADDITION

- × **Add two binary digits, what do we get?**
 - + $0 + 0 =$
 - + $0 + 1 =$
 - + $1 + 0 =$
 - + $1 + 1 =$
- × **Conclude:** May get two-bit result

24

24

ESE1500 Spring 2023

ADDITION

- × Add three binary digits, what's largest binary result?
 - + 1+1+1=
- × **Conclude:** Also get two-bit result adding 3 bits

25

25

ESE1500 Spring 2023

FULL ADDER

- × Adds 3 inputs to produce 2b output
 - + Binary inputs: a, b, c
 - + Binary outputs: carry, sum
 - + Two bit result:
 - + carry*2 +sum = a+b+c
 - + It's just another gate we can define
 - + Can produce truth table and logic (Lab)

26

26

ESE1500 Spring 2023

EXAMPLE: BIT-LEVEL ADDITION

- × Addition $S = A + B$
- × $s_{n-1} s_{n-2} \dots s_0 = a_{n-1} a_{n-2} \dots a_0 + b_{n-1} b_{n-2} \dots b_0$
 - + Base 2 example
 - + Work together

c	1	1	1	1	1	1	0	0	0
A	0	1	1	1	0	1	1	1	0
B	0	0	1	1	1	0	1	0	0
S	1	0	1	1	0	0	0	1	0

27

27

ESE1500 Spring 2023

FULL ADDER

- × Adds 3 inputs to produce 2b output
 - + Binary inputs: a, b, c
 - + Binary outputs: carry, sum
 - + Two bit result:
 - + carry*2 +sum = a+b+c
 - + Can produce truth table and logic (Lab)
- × **Natural primitive for bit-level addition with carry**

28

28

ESE1500 Spring 2023

N-BIT ADDER

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

29

29

Part 3


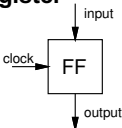
ACCUMULATOR

30

30

ESE1500 Spring 2023

REGISTER

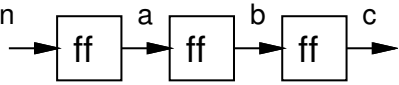
- × **Clock**

 - + Defines the rate of the computation
 - + Typically a square wave
 - + Rising clock edge defines beginning of new cycle
- × **State Element – Flip-Flop (FF) or Register**
 - + Returns the value it was given on previous cycle = before the last rising clock edge

31

31

ESE1500 Spring 2023

REGISTER EXAMPLE



Cycle	in	a	b	c
0	1	?	?	?
1	0	1	?	?
2	0	0	1	?
3	1	0	0	1

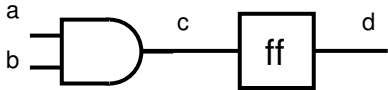
Fill in values

32

32

ESE1500 Spring 2023

LOGIC EXAMPLE



Cycle	a	b	c	d
0	1	1	1	?
1	0	1	0	1
2	0	0	0	0
3	1	1	1	0

Fill in values

33

33

ESE1500 Spring 2023

ACCUMULATOR

- × **Sum a sequence of values**
 - × **a=0**
 - × **while (true)**
 - + a=a+getInput();

Input[i]	a
	0
1	1
3	4
1	5

$$a = \sum_{i=0} input[i]$$

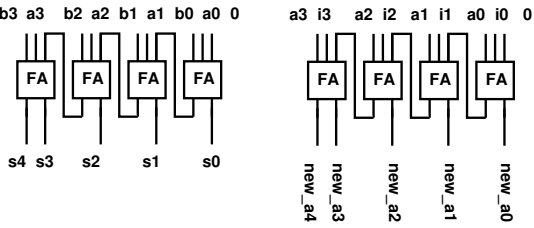
34

34

ESE1500 Spring 2023

ACCUMULATOR

- × **Start with an Adder**



35

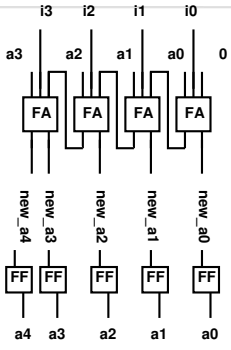
35

ESE1500 Spring 2023

ACCUMULATOR

- × **Store running sum as state in registers**
- × **Sum up new values provided on successive cycles**

Input[i]	a
	0
1	1
3	4
1	5



36

36

ESE1500 Spring 2023

ACCUMULATOR

Inputs: i_3, i_2, i_1, i_0
Outputs: $new_a_3, new_a_2, new_a_1, new_a_0$

× Wrap register outputs back to inputs

37

ESE1500 Spring 2023

ACCUMULATOR

Inputs: i_3, i_2, i_1, i_0
Outputs: a_4, a_3, a_2, a_1, a_0

× Maybe extend accumulator bits to hold larger sum

× Maybe more...

38

ESE1500 Spring 2023

ACCUMULATOR

Input[i]	a
	0
15	15
15	30
15	45
15	60
15	75

× Maybe extend accumulator bits to hold larger sum

× Why want more?

39

ESE1500 Spring 2023

ACCUMULATOR

× **What happens:**

- + Start with $a_3:a_0$ at 0
- + CLK low
- + $I_3:i_0=2$ (0010)
- + FF inputs?

40

ESE1500 Spring 2023

ACCUMULATOR

× **What happens:**

- + Start with $a_3:a_0$ at 0
- + CLK low
- + $I_3:i_0=2$ (0010)
- + FF inputs?
- + CLK goes high: $a_3:a_0$?

41

ESE1500 Spring 2023

ACCUMULATOR

× **What happens:**

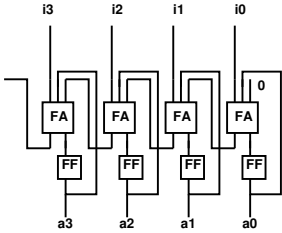
- + Start with $a_3:a_0$ at 0
- + CLK low
- + $I_3:i_0=2$ (0010)
- + CLK goes high: $a_3:a_0$?
- + $I_3:i_0=3$ (0011)
- + FF inputs?

42

ESE1500 Spring 2023

ACCUMULATOR

- × **What happens:**
 - + Start with a3:a0 at 0
 - + CLK low
 - + I3:i0=2 (0010)
 - + CLK goes high: a3:a0?
 - + I3:i0=3 (0011)
 - + FF inputs?
 - + CLK goes high: a3:a0?



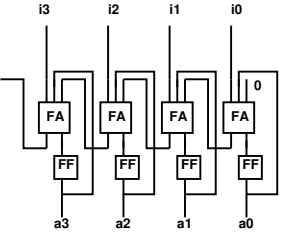
43

43

ESE1500 Spring 2023

ACCUMULATOR

- × **a=0**
- × **while (true)**
 - + a=a+getInput();

$$a = \sum_{i=0} input[i]$$


44

44

ESE1500 Spring 2023

NEXT LAB (MONDAY)

- × Lab 7 posted on web
- × Program an FPGA in Verilog
 - + Build an adder
 - + Build an accumulator

45

45

ESE1500 Spring 2023

BIG IDEAS

- × Can implement any combinational digital logic function from (and, or, not) gates
- × Can store previous values
 - + Flip-flops or registers
- × Enough to perform math we need for audio processing (...and much...)

46

46

ESE1500 Spring 2023

LEARN MORE

- × CIS2400 – do a bit more logic
- × ESE3700 – how to implement gates, latches, and memories from transistors
- × ESE5320 – how to build large-scale computations from logic

47

47

ESE1500 Spring 2023

REMINDER

- × Feedback
- × Monday Lecture: Sequential Logic
- × Formal Lab Report Due Monday

48

48