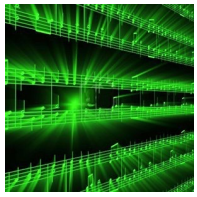
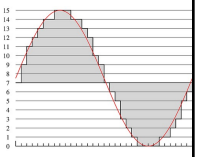
Penn Engineering ESE





Lecture #14 – Sequential Logic, FPGAs

ESE 1500 – Digital Audio Basics

ESE1500 Spring 2023

Based on slides © 2009--2023 DeHon

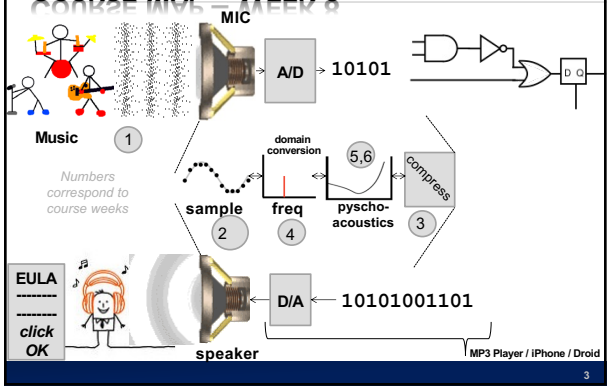
1

Lecture Topics

- Setup
- **Where are we?**
- **Part 1:**
  - Review: Combinational Logic
  - FPGAs
- **Part 2:** (as time permits)
  - Finite-State Machines (FSM)

2

Course Map – Week 8



3

Combinational Logic

4

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

Conclude

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

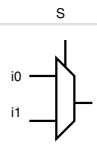
6

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….**

7

MULTIPLEXER GATE



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

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?

8

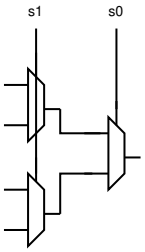
PRECLASS 2

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

9

PRECLASS 2

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

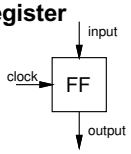


10

REGISTER

- **Clock**

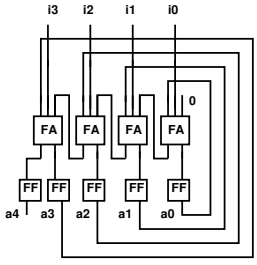
  - Defines the rate of the computation
  - Typically a square wave
  - Rising clock edge defines beginning of new cycle
- **State Element – Flip-Flop (FF) or Register**
  - Returns the value it was given on previous cycle = before the last rising clock edge

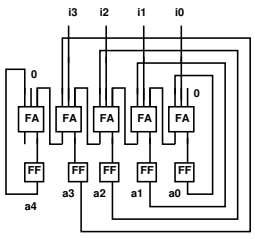


11

ACCUMULATOR REVISITED

- Maybe extend accumulator bits to hold larger sum





- Maybe more…

12

Penn Engineering

ESE

PROGRAMMABLE LOGIC

13

13

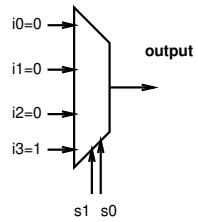
PROGRAMMABLE LOGIC

- **Demonstrated can build any combinational logic function from AND, OR, NOT gates.**
- **…but, it's a *different* circuit for each logic function.**
- **Can we build one circuit that can act as *any* logic function?**

14

PRECLASS 3

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

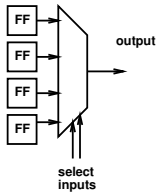


15

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

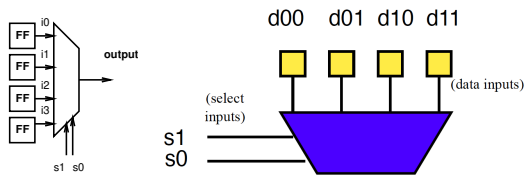
Input 0	Input 1	Output
0	0	
0	1	
1	0	
1	1	



16

EXAMPLE: OR (PRECLASS 4)

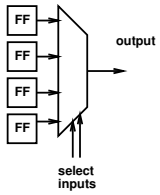
- **How do we program to behave as OR2?**



17

LOOK-UP TABLE (LUT)

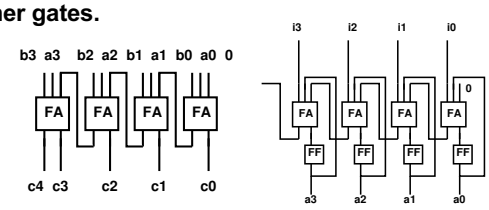
- **Can generalize to any number of inputs**



18

Connecting Gates

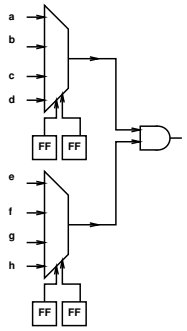
- **Once we can build gates**
- **…still need to connect the gates together.**
- **Select which gate outputs become inputs to other gates.**



19

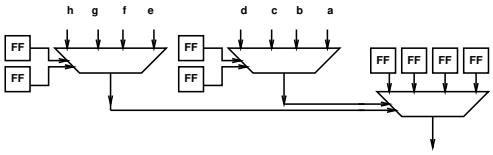
Mux can be Programmable Interconnect

Trick: Use multiplexer to programmably select gate input.



20

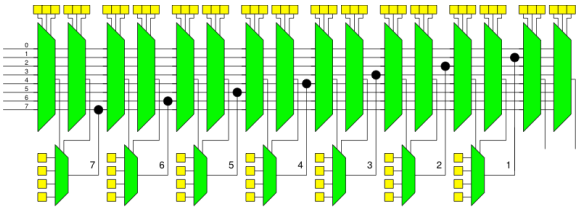
Programmable Blocks



21

Programmable Gates and Interconnect

Preclass 5:
How program (fill in yellow programmable cells) to implement a full adder?



Note: Work on Google Docs

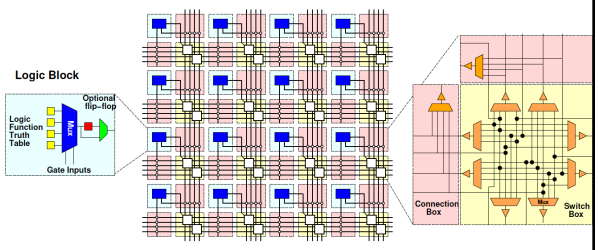
22

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

23

Field-Programmable Gate Array (FPGA)



24

Penn Engineering ESE

Part 3

FINITE STATE MACHINES (FSMS)

25

25

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

26

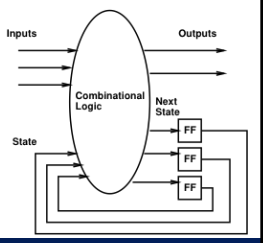
STATE FOR CONDITIONAL CONTROL

- **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

27

FINITE-STATE MACHINE (FSM)

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



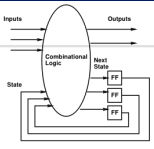
28

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)**

29

TRUTH TABLE MODEL



Complete together

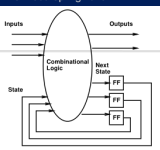
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1			
waiting	1	0	0	0	one
waiting	1	1			
one	0	0			
one	0	1			
one	1	0			
one	1	1			

30

ESE1500 Spring 2023

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			

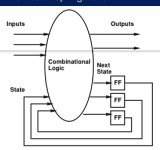
31

31

ESE1500 Spring 2023

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			

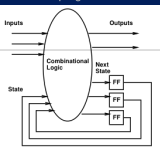
32

32

ESE1500 Spring 2023

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			

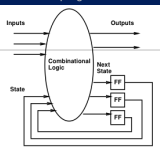
33

33

ESE1500 Spring 2023

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			

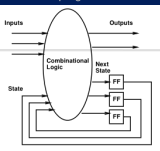
34

34

ESE1500 Spring 2023

TRUTH TABLE MODEL

Complete together



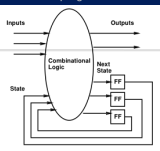
state	quarter	creturn	vend	qreturn	next
waiting	0	0	0	0	waiting
waiting	0	1	0	0	waiting
waiting	1	0	0	0	one
waiting	1	1	0	1	waiting
one	0	0	0	0	one
one	0	1	0	1	waiting
one	1	0	1	0	waiting
one	1	1			

35

35

ESE1500 Spring 2023

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

36

36

Switch-Statement Model

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

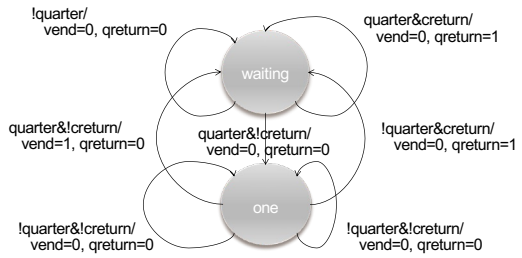

37

Switch-Statement Model (cont.)

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


38

FSM Graph Model



39

Big Ideas

- **Can implement any combinational digital logic function from:**
  - and, or, not gates
- **Can implement any FSM from:**
  - and, or, not 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

40

Learn More

- **CIS2400 – do a bit more logic**
- **ESE3700 – how to implement gates, latches, and memories from transistors**
- **CIS4710 – implement processor (next time) using Verilog mapped to FPGAs**
- **ESE5320 – how to build large-scale computations from logic**

41

Reminder

- **Feedback**
- **Lab 6 Report/formal writeup due today**
- **Lab 7 this afternoon**
  - Prelab (with canvas/quiz turnin)

42