Field Programmable Gate Array (FPGA) Curriculum update with Industry

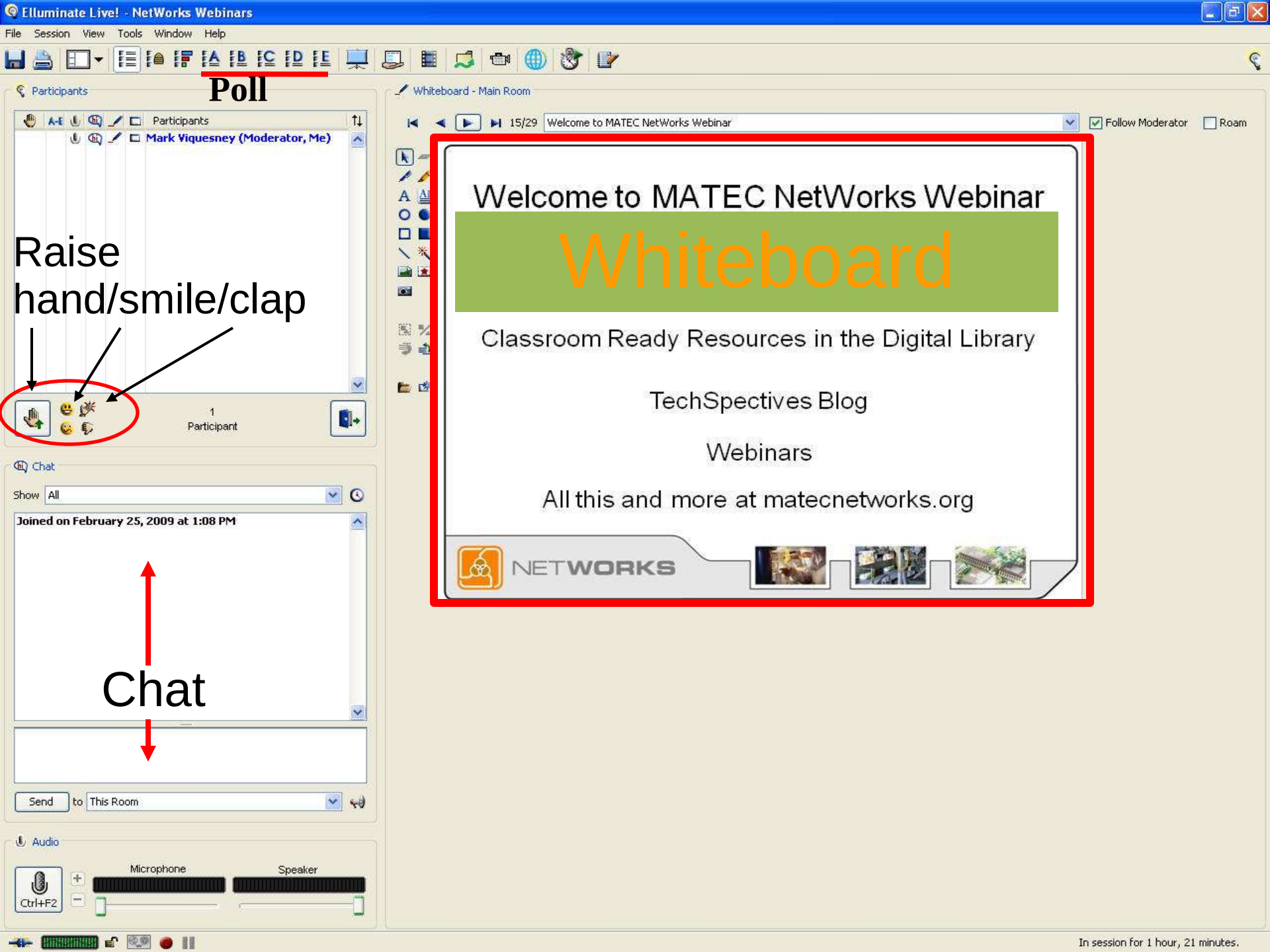
HTWI
High Tech Workforce Initiative

MARICOPA COMMUNITY COLLEGES®

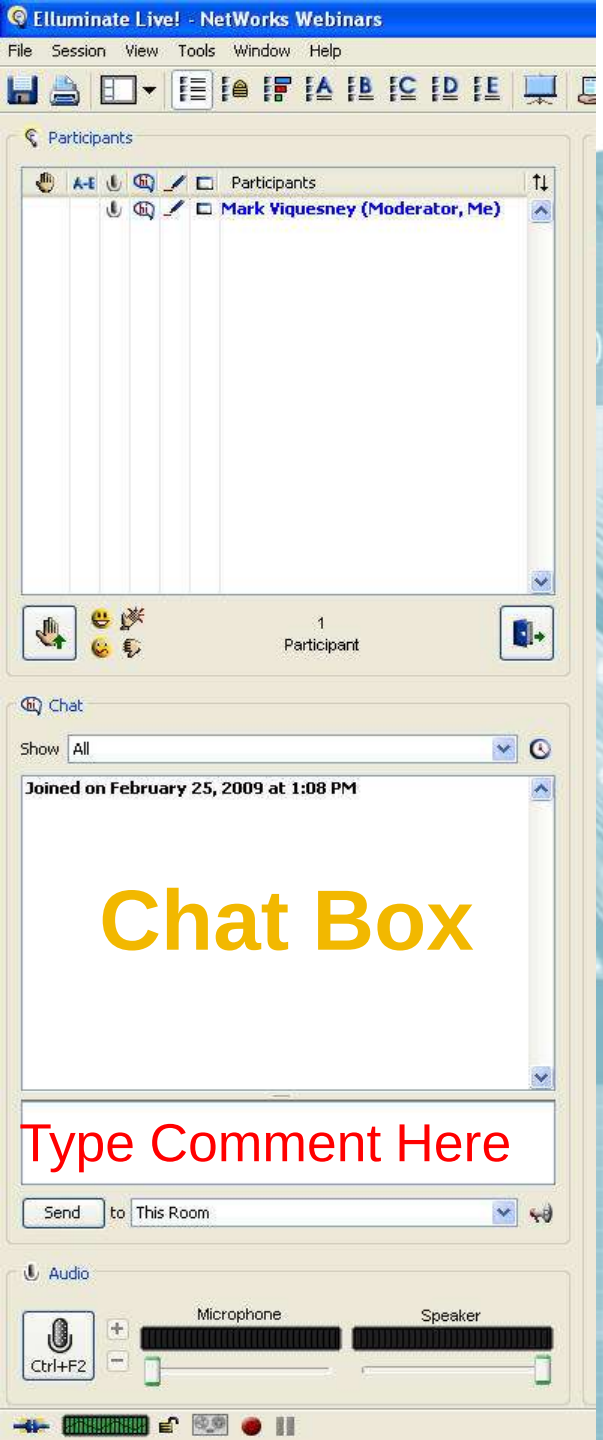
National Science Foundation
WHERE DISCOVERIES BEGIN

Hosted by MATEC NetWorks

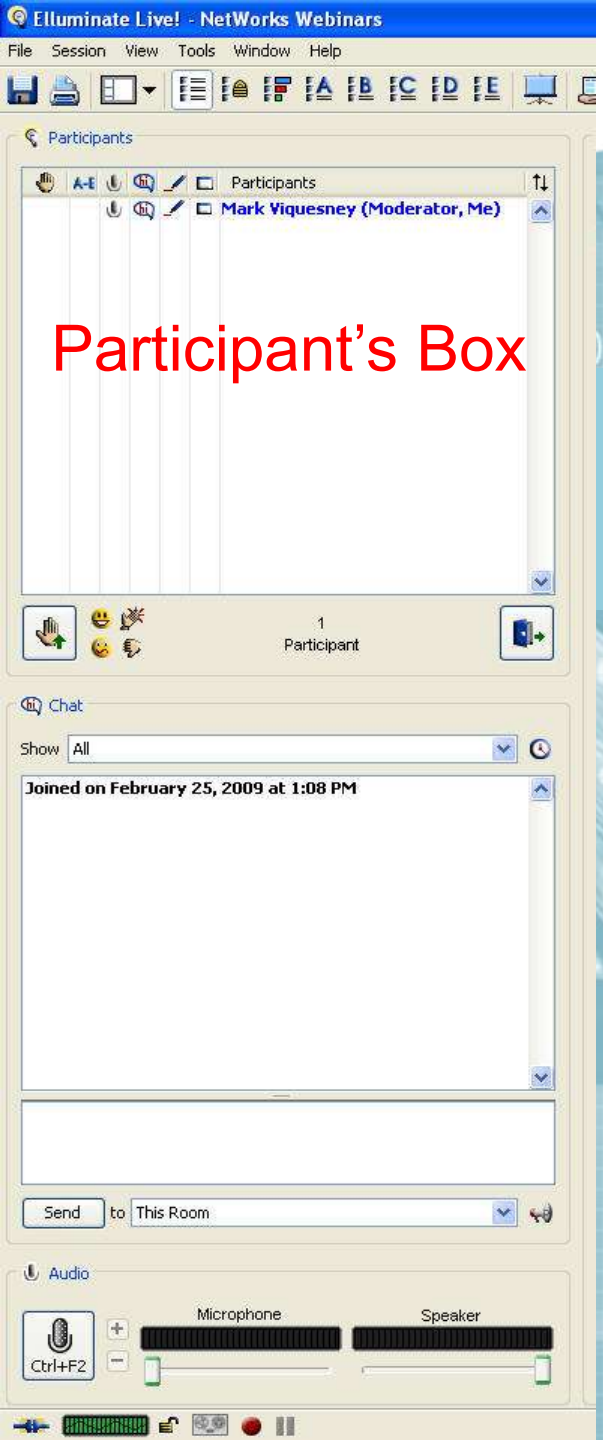
Funded, in part by a grant from the
National Science Foundation
DUE 1003542


Elluminate Live! - NetWorks Webinars
File Session View Tools Window Help
Poll
Participants
Participants
Mark Viquesney (Moderator, Me)
Raise hand/smile/clap
1 Participant
Chat
Show All
Joined on February 25, 2009 at 1:08 PM
Chat
Send to This Room
Audio
Microphone
Speaker
Ctrl+F2
Whiteboard - Main Room
15/29
Welcome to MATEC NetWorks Webinar
Follow Moderator
Roam
Welcome to MATEC NetWorks Webinar
Whiteboard
Classroom Ready Resources in the Digital Library
TechSpectives Blog
Webinars
All this and more at matecnetworks.org
NETWORKS
In session for 1 hour, 21 minutes.

Chat Box



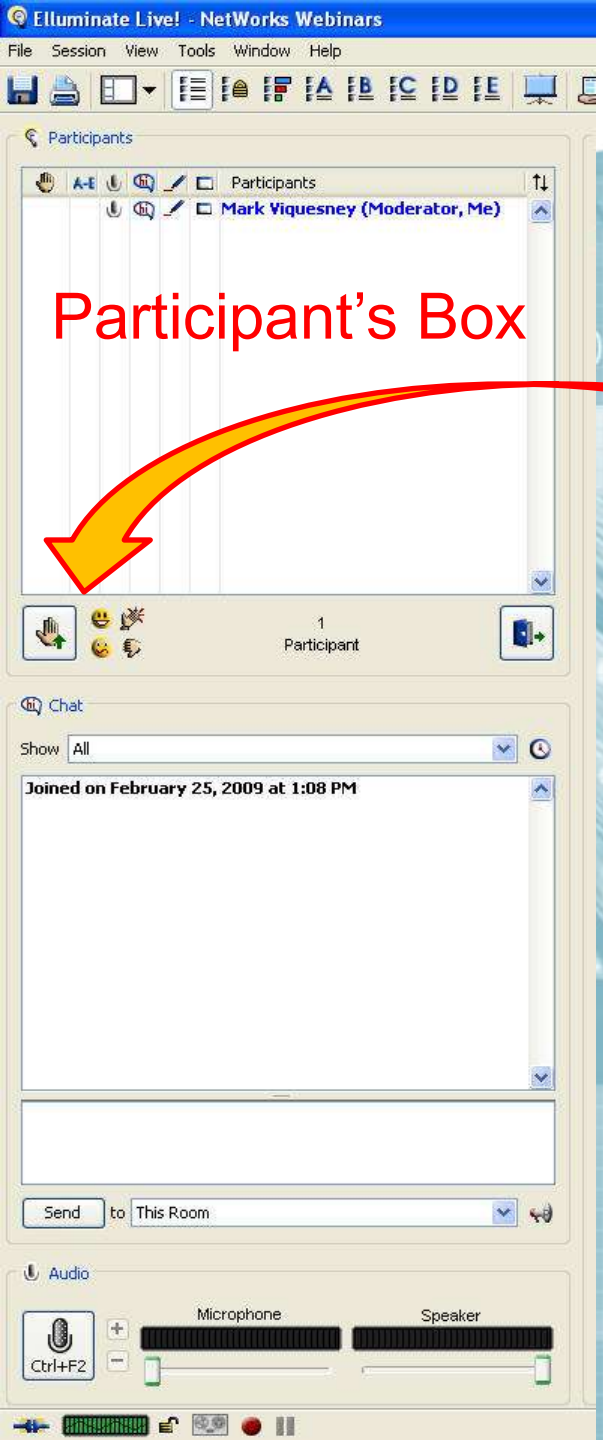
In the **Chat Box**,
please type the name of
your school or organization,
your location,
and how many people are
attending with you today.

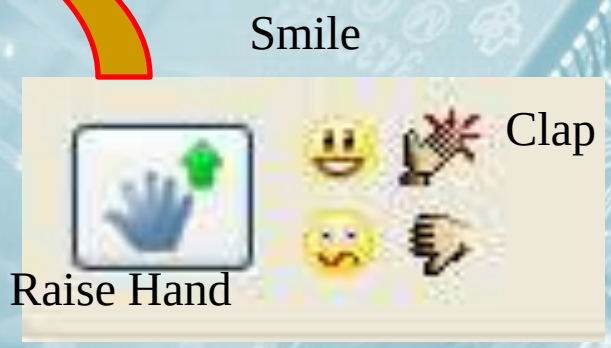


Participant's Box

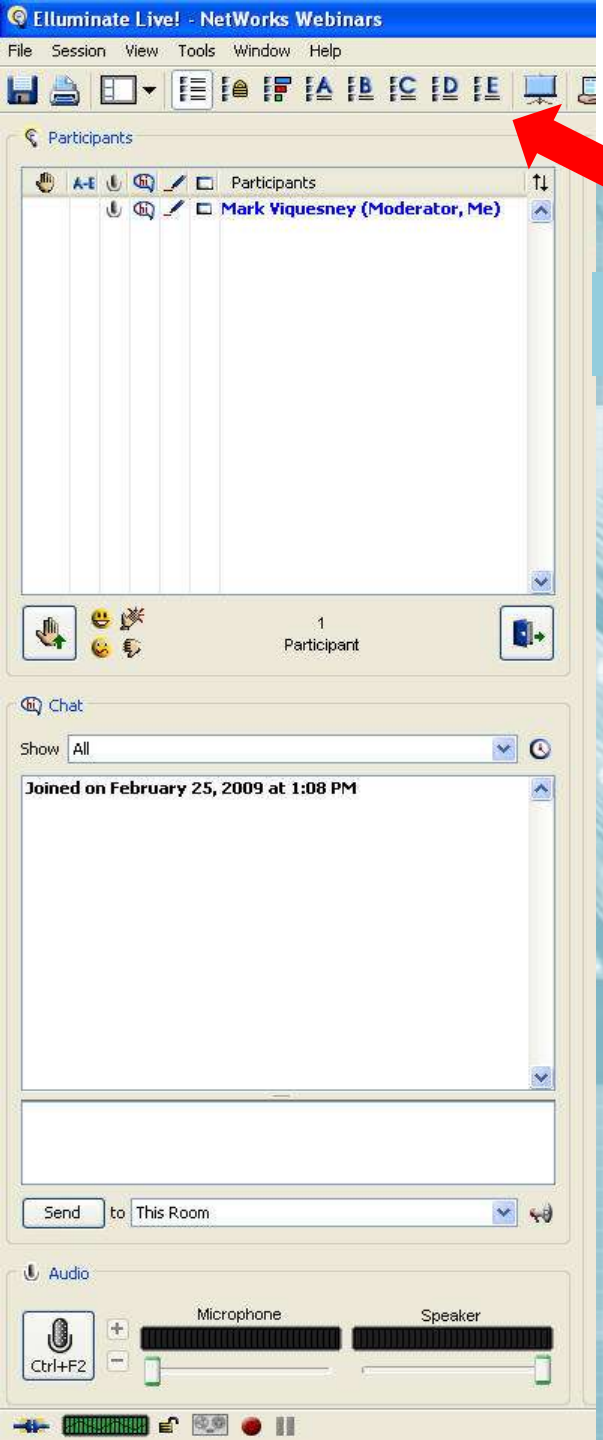
Allows you to non-verbally respond to the presenter's comments.

Participant's Box





Let the presenter know if you like what they say with a smile or clap. Raise a hand if you have a question – and then type it into the chat box.



Poll

Click A-E to take the Poll

This webinar will have a Poll. Please answer: I heard about this webinar through:

A. @matec
B. Email from ETD list serv
C. Email from HTWI
D. Friend or colleague
E. Other (please type where in chat box)

eSyst Webinar Presenters

HTWI
High Tech Workforce Initiative

Bassam Matar

Engineering Faculty and
Program Coordinator at
Chandler/Gilbert C.C.

Ui Luu

Electronics Faculty
Glendale C.C.

Field Programmable Gate Array (FPGAs) Curriculum Validation with Industry Agenda

- Overview of the existing course offerings at MCCCD.
- Overview of the externship with local industry
- Curriculum Update
- Xilinx Workshop
- Summer 2011 Workshop
- Survey and Final Questions from Participants

HTWI Project Team Members

HTWI
High Tech Workforce Initiative

Rick Hansen– Principal Investigator

Lizette Acosta - Coordinator Marketing & Academic Advisement

HTWI Project Overview

HTWI
High Tech Workforce Initiative

Faculty work side by side with industry to design, develop and deliver this curriculum.

HTWI Project Overview (cont.)

HTWI
High Tech Workforce Initiative

1) Research industry requirements and provide students with a relevant, rigorous and interdisciplinary course of study

HTWI Project Overview (cont.)

HTWI
High Tech Workforce Initiative

2) Reform curriculum and delivery to meet industry requirements and provide students with a relevant, rigorous and interdisciplinary course of study

HTWI Project Overview (cont.)

HTWI
High Tech Workforce Initiative

3) Redefine Maricopa Colleges' outreach and retention strategies to attract more students into the high-tech manufacturing fields to meet future employment demands

Electronics and Engineering Courses

HTWI
High Tech Workforce Initiative

- Digital Design Fundamentals (CSC/EEE 120)
- Introduction to Digital Logic (ELT 131)
- Microprocessor Applications inc. Microcontrollers (CSC/EEE 230 or ELT 241

Electronics and Engineering Courses

HTWI
High Tech Workforce Initiative

Our existing courses have been focused on Transistor-Transistor-Logic (TTL).

As a result of an externship in the Summer of 2009 we started implementing Field Programmable Gate Array (FPGA) using Schematics captures.

Externship with Local Industry

HTWI
High Tech Workforce Initiative

- In phase 1 of this project, we consulted with Honeywell & General Dynamics on FPGAs in industrial use
- Our initial findings: Very (High Speed Integrated Circuits) Hardware Description Language (VHDL)
- The growing trend in Digital Design is to adapt Verilog as a new design tool.

Externship with Local Industry (cont.)

HTWI
High Tech Workforce Initiative

After review we concluded the existing digital course curriculum requires

- more coverage in VHDL
- less with Schematic Capture

Xilinx software works better with VHDL programming than Schematic Capture.

Action Plan

HTWI
High Tech Workforce Initiative

This gap analysis prompts us to:

1. Attend FPGA/VHDL workshop sponsored by Xilinx.
   - If future externship opportunity exists, we will work with our industrial partner to develop further Verilog updates.

2. Update our Digital Design curriculum to increase coverage in VHDL.

General Lecture Recommendation

HTWI
High Tech Workforce Initiative

The following subjects should be added to a Digital course or more detailed coverage should be included:

- Increased coverage of PLDs.
- Basic programming techniques in VHDL, Verilog or an equivalent language should be introduced.

Rationale: A large percentage of new digital designs use a PLD and students must be familiar with the basic types, inputs, outputs, specifications, and architectures. Students must know how to access inputs and outputs and assess functionality.

General Lab Recommendation

HTWI
High Tech Workforce Initiative

- Less breadboarding with discrete TTL or CMOS logic.
- Add significant PLD coverage that can be programmed as well as including an Altera or Xilinx student demo board with
  - prebuilt I/O devices (switches, LEDs, LDC display, etc)
  - multiple common interfaces
    - ex: FPGA Spartan 3E, Nexys2 FPGA Board
- Add activities that allow students to practice test and measurement techniques and troubleshooting. Add a logic analyzer to the lab as budget permits.

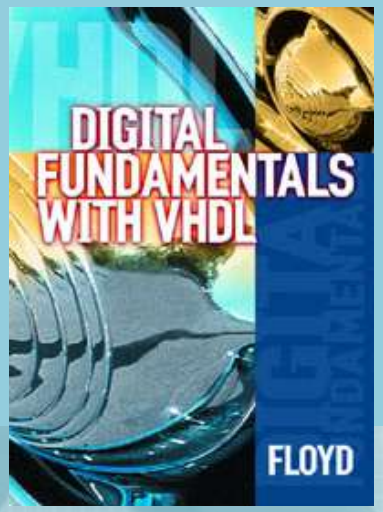
General Textbook Recommendation

HTWI
High Tech Workforce Initiative

- Use the existing textbook but edit the content to avoid or de-emphasize the topics discussed previously and to enhance the suggested topics.
- Search for supplementary material on the Internet and other sources to cover topics not adequately covered in the textbooks.
- Change to a textbook that incorporates the most recent techniques
  - ex: PLDs and microcontrollers.

Sample reference book:

Digital Fundamentals with VHDL by Floyd



HTWI
High Tech Workforce Initiative

Questions?

Sample

HTWI
High Tech Workforce Initiative

- Arithmetic Logic Unit (ALU) Lecture with VHDL

  By Bassam Matar

- Security System Design Lab with VHDL

  By Ui Luu

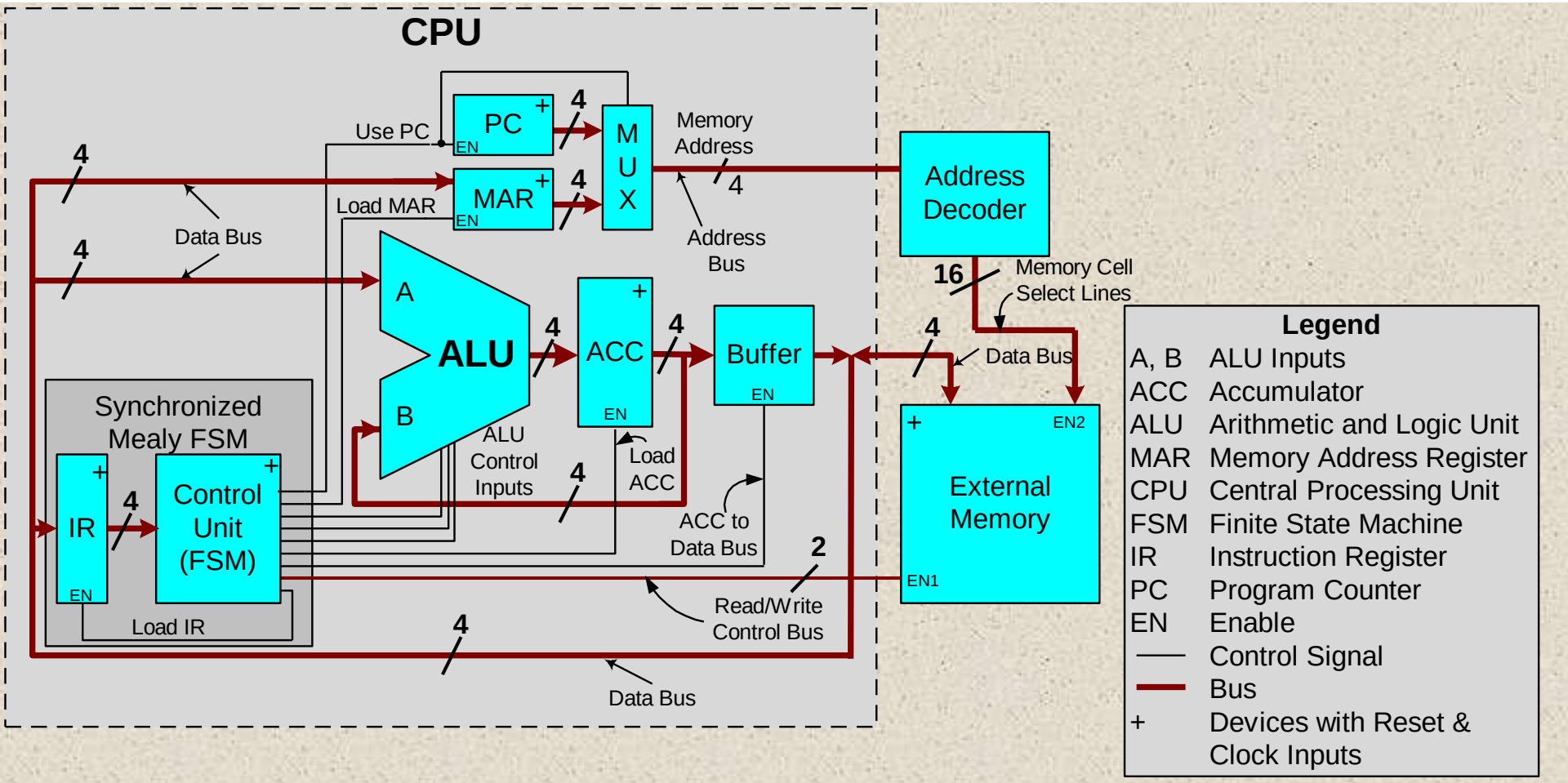
Arithmetic Logic Unit (ALU)

Design and Build an ALU with Schematic Capture and VHDL

ALU's

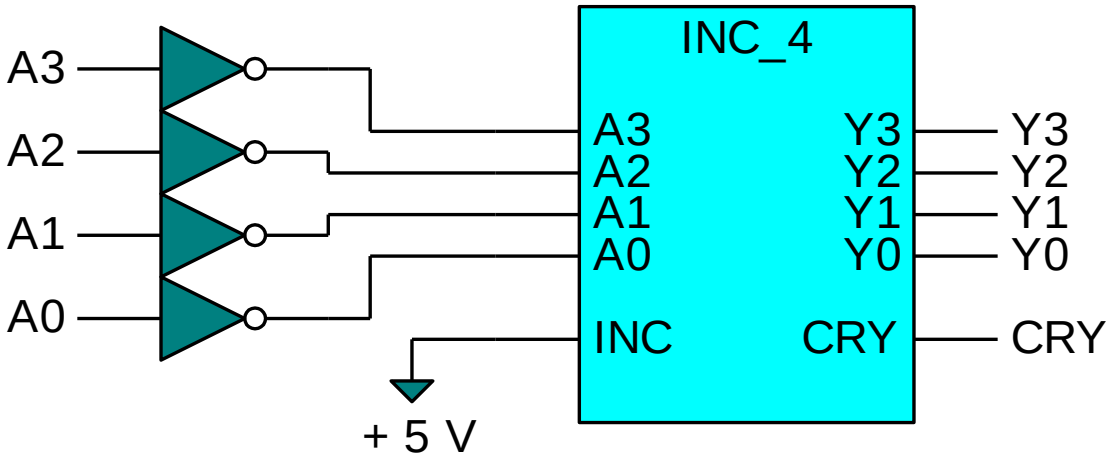
- Let's talk about Arithmetic and Logic Unit's (ALU's.)
- A microprocessor is made up of many components:
  - Central Processing Unit - Performs the data manipulation tasks of the computer and sequences each step in these tasks.
    - Arithmetic and Logic Unit (ALU) - is the part of the CPU which does the arithmetic (+,-,*,etc.) and the logical operations (AND, OR, NOT etc.).

ALU's

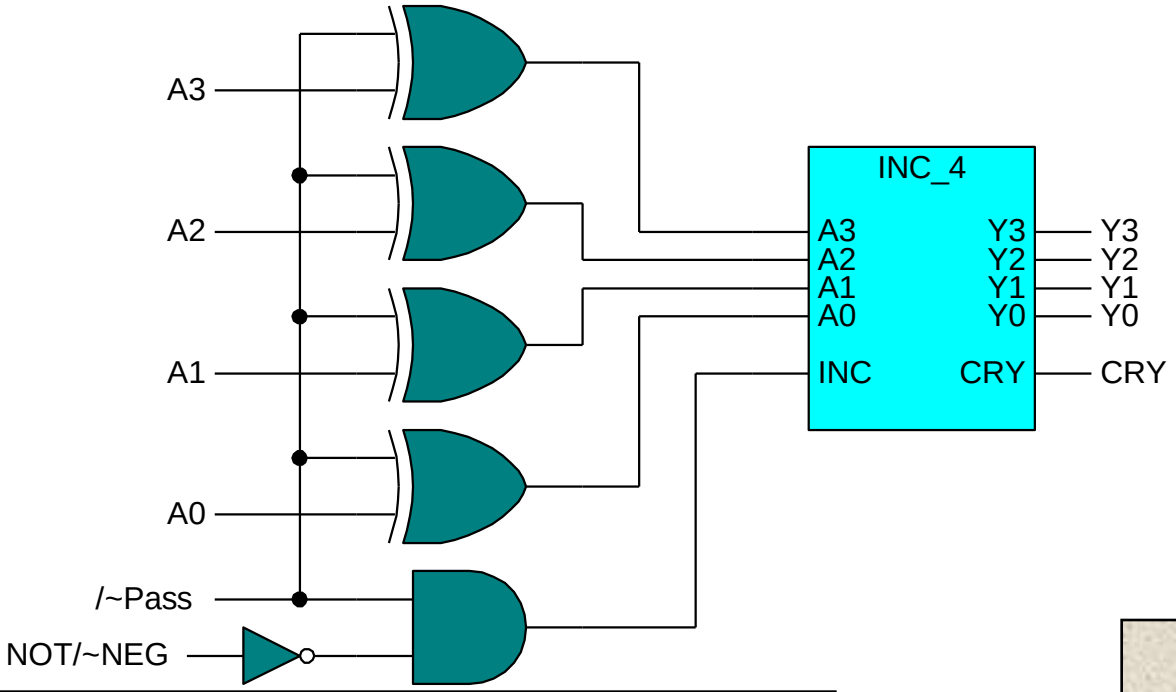

CPU
PC
EN
MAR
EN
MUX
Use PC
Load MAR
Memory Address
Address Bus
Data Bus
A
ALU
B
ACC
EN
Buffer
EN
ALU Control Inputs
Load ACC
ACC to Data Bus
Synchronized Mealy FSM
IR
EN
Control Unit (FSM)
Load IR
Read/Write Control Bus
Address Decoder
Memory Cell Select Lines
Data Bus
External Memory
EN1
EN2
Legend
A, B ALU Inputs
ACC Accumulator
ALU Arithmetic and Logic Unit
MAR Memory Address Register
CPU Central Processing Unit
FSM Finite State Machine
IR Instruction Register
PC Program Counter
EN Enable
Control Signal
Bus
+ Devices with Reset & Clock Inputs

ALU's

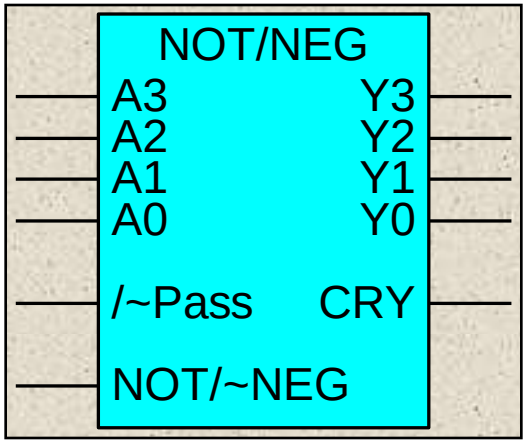
- Each ALU has a different design. Lets look at the one you're building in simulation lab 3.
- You already took the first step in the ALU design. In Sim Lab 1 you built the circuit that does the two's complement operation.



ALU's

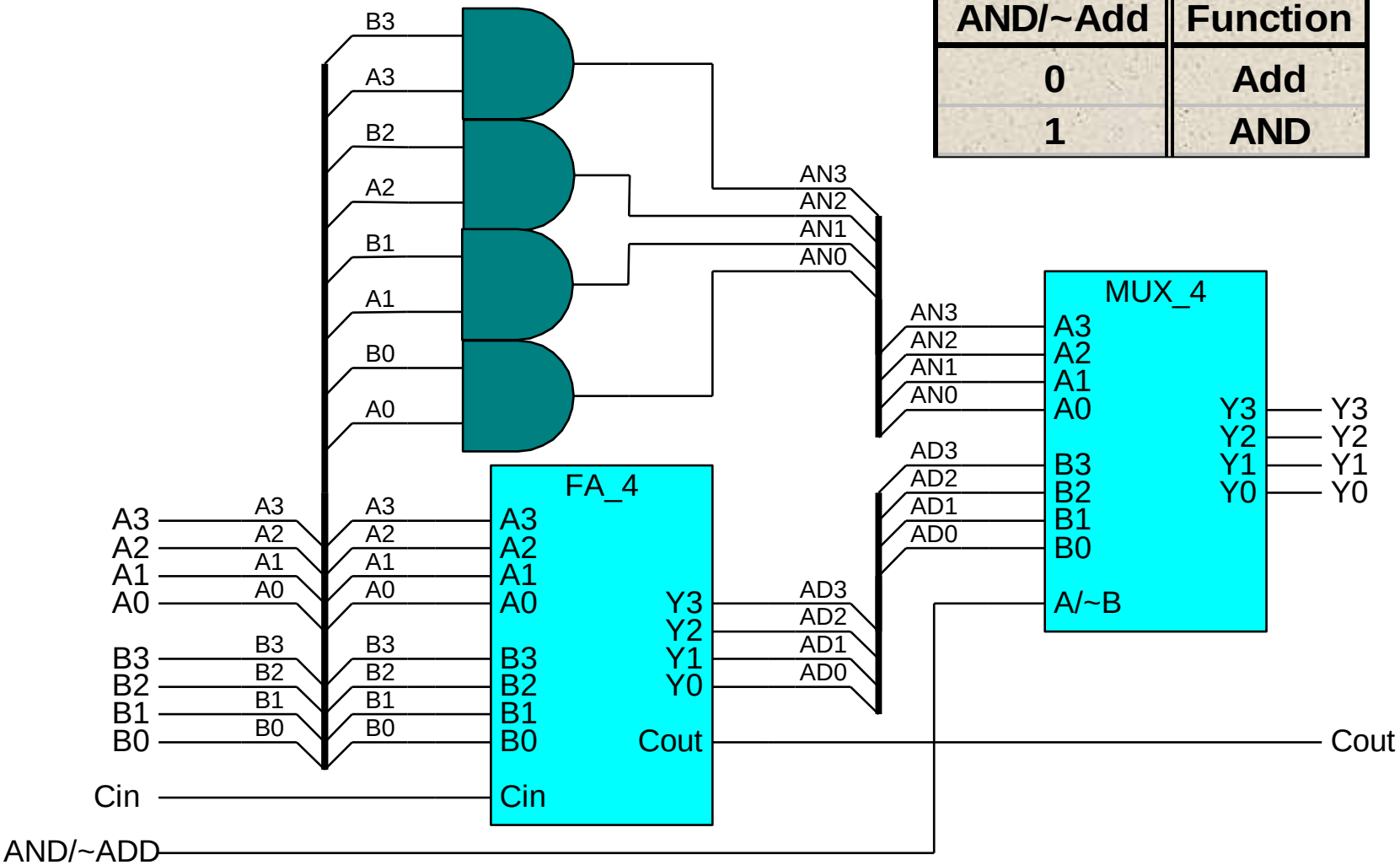


/~Pass	NOT/~NEG	FUNCTION
0	0	Pass-Through A
0	1	Pass-Through A
1	0	Two's Complement
1	1	One's Complement



ALU's

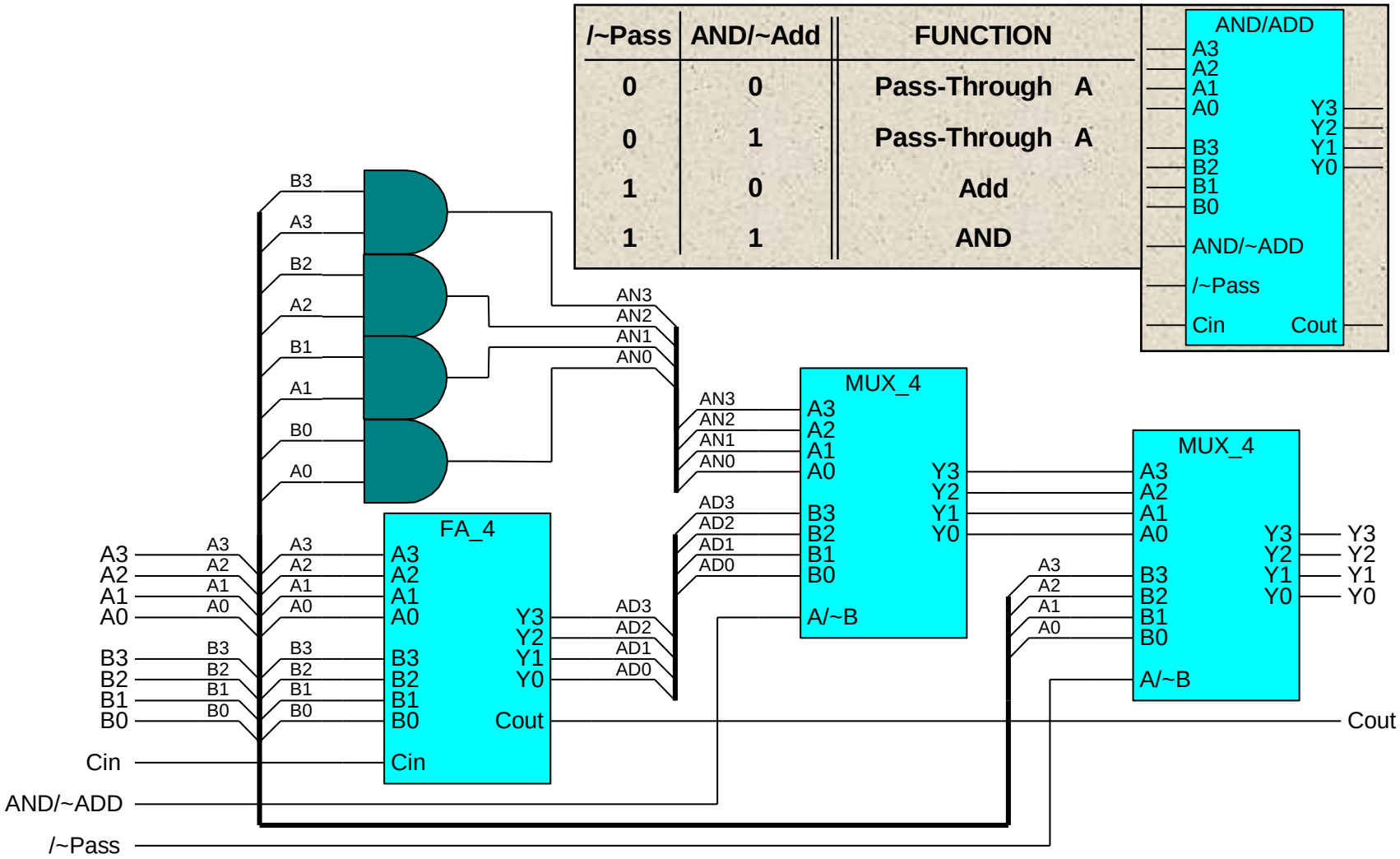
AND/~Add	Function
0	Add
1	AND



AND/ADD Circuit
w/ Pass-Through

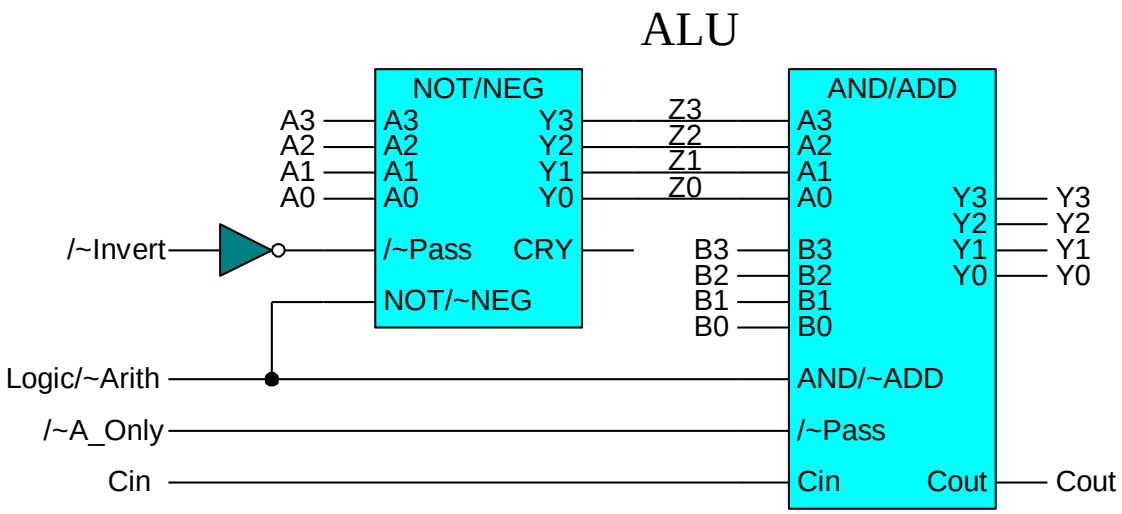
ALU's

/~Pass	AND/~Add	FUNCTION
0	0	Pass-Through A
0	1	Pass-Through A
1	0	Add
1	1	AND



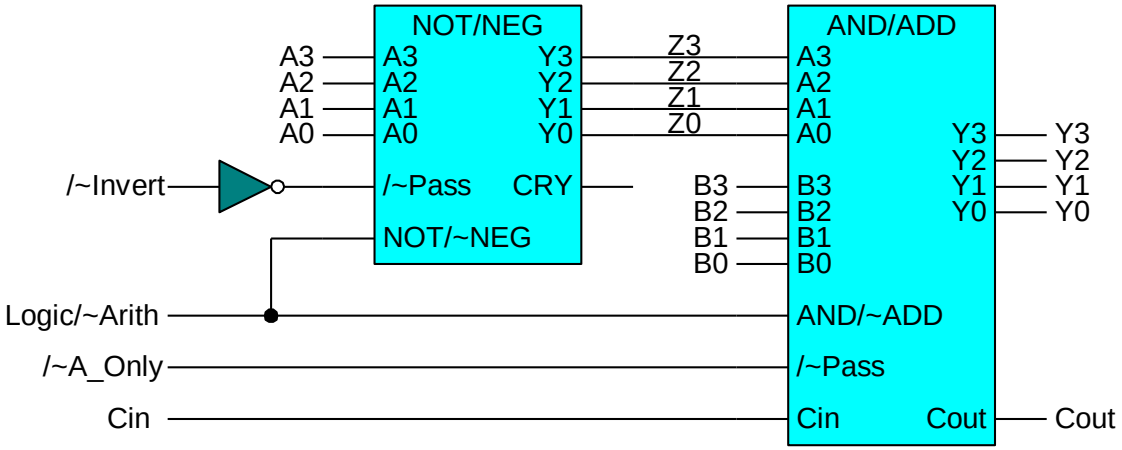
ALU's

TEAMS: Fill in the function definition table below for the ALU.



A ONLY'	ARITH'/LOGIC	INVERT	FUNCTION	HEX INPUT A	HEX INPUT B	HEX OUTPUT
0	0	0	PASS-THROUGH	E	D	E
0	0	1	2's COMPLIMENT of A	C	E	4
0	1	0	PASS-THROUGH	F	9	F
0	1	1	1's COMP of A	F	C	0
1	0	0	A plus B	2	5	7
1	0	1	(2's COMP A) plus B	F	A	B
1	1	0	A AND B	F	8	F
1	1	1	(1's COMP of A) AND B	2	5	D

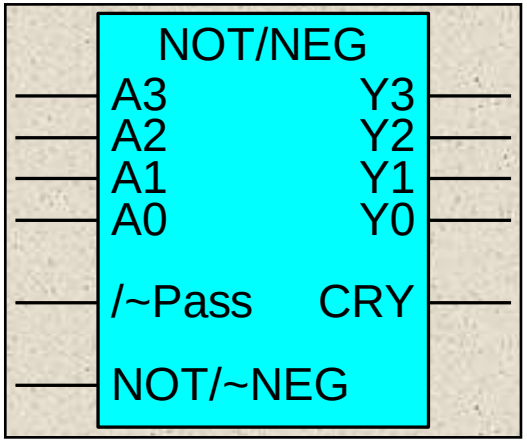
Top Level VHDL Programming of ALU's



A ONLY'	ARITH'/LOGIC	INVERT	FUNCTION	HEX INPUT A	HEX INPUT B	HEX OUTPUT
0	0	0	PASS-THROUGH	E	D	E
0	0	1	2's COMPLIMENT of A	C	E	4
0	1	0	PASS-THROUGH	F	9	F
0	1	1	1's COMP of A	F	C	0
1	0	0	A plus B	2	5	7
1	0	1	(2's COMP A) plus B	F	A	B
1	1	0	A AND B	F	8	F
1	1	1	(1's COMP of A) AND B	2	5	D

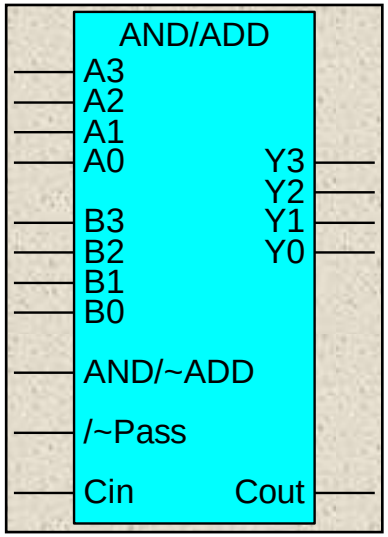
NEG_NOT VHDL Component Programming of ALU's

/~Pass	NOT/~NEG	FUNCTION
0	0	Pass-Through A
0	1	Pass-Through A
1	0	Two's Complement
1	1	One's Complement



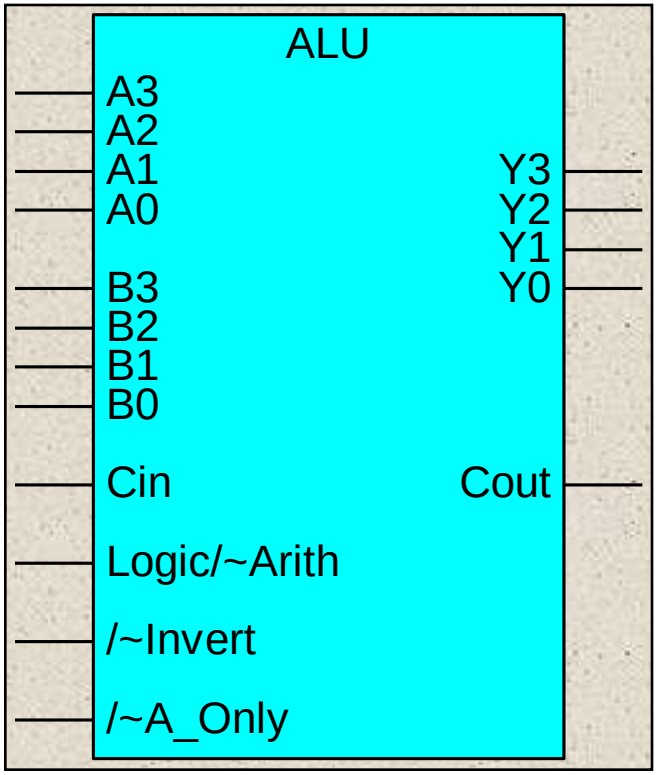
AND/ADD VHDL Component Programming of ALU's

/~Pass	AND/~Add	FUNCTION
0	0	Pass-Through A
0	1	Pass-Through A
1	0	Add
1	1	AND



ALU's

- We need the function definition tables because after the ALU is put in a "box", we need to know how to use it and connect it.



Sample

HTWI
High Tech Workforce Initiative

- Arithmetic Logic Unit (ALU) Lecture with VHDL

  By Bassam Matar

- Security System Design Lab with VHDL

  By Ui Luu

Security State Machine

Ui Luu

Glendale Community College

Security System I/O

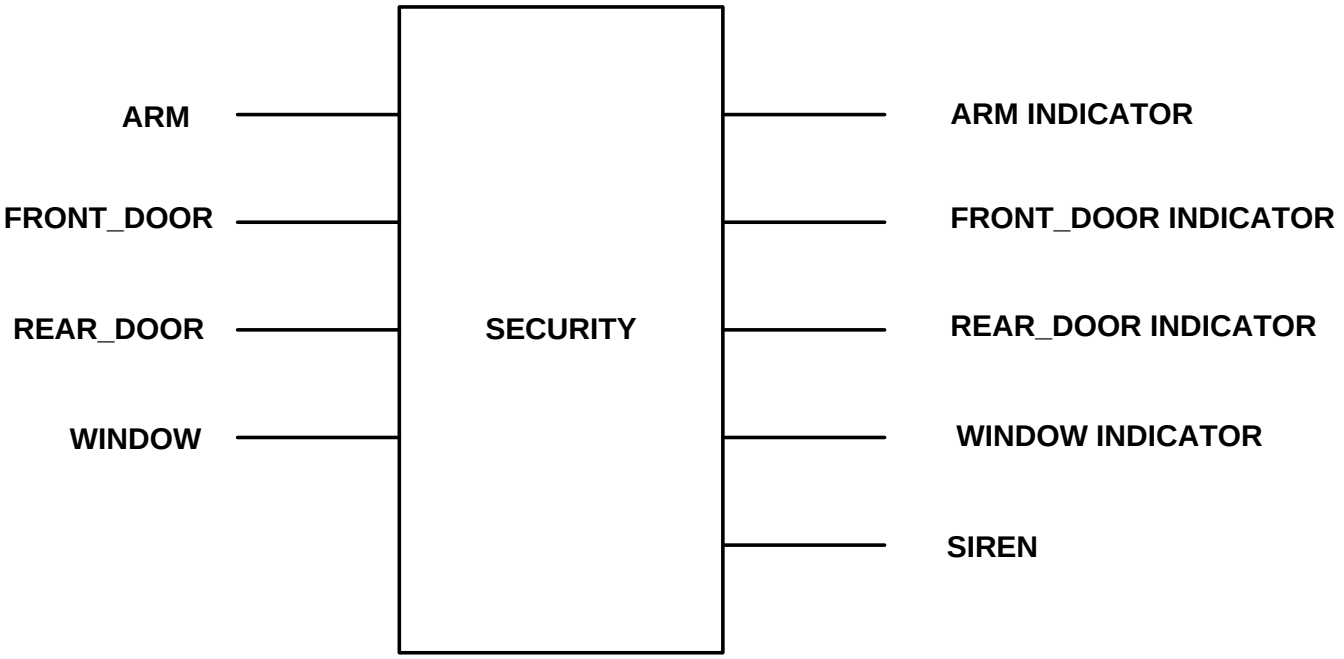


Figure 1. Security System I/O

Security System State Diagram

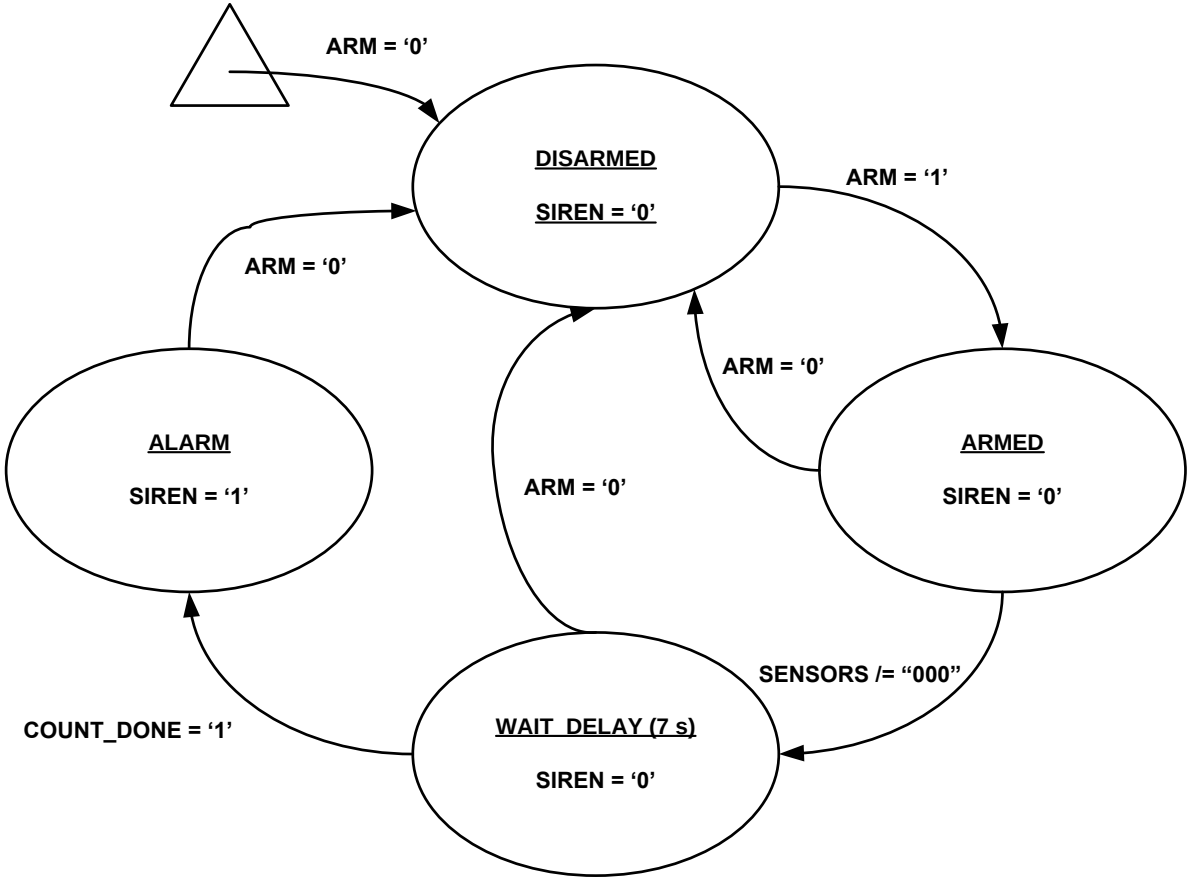


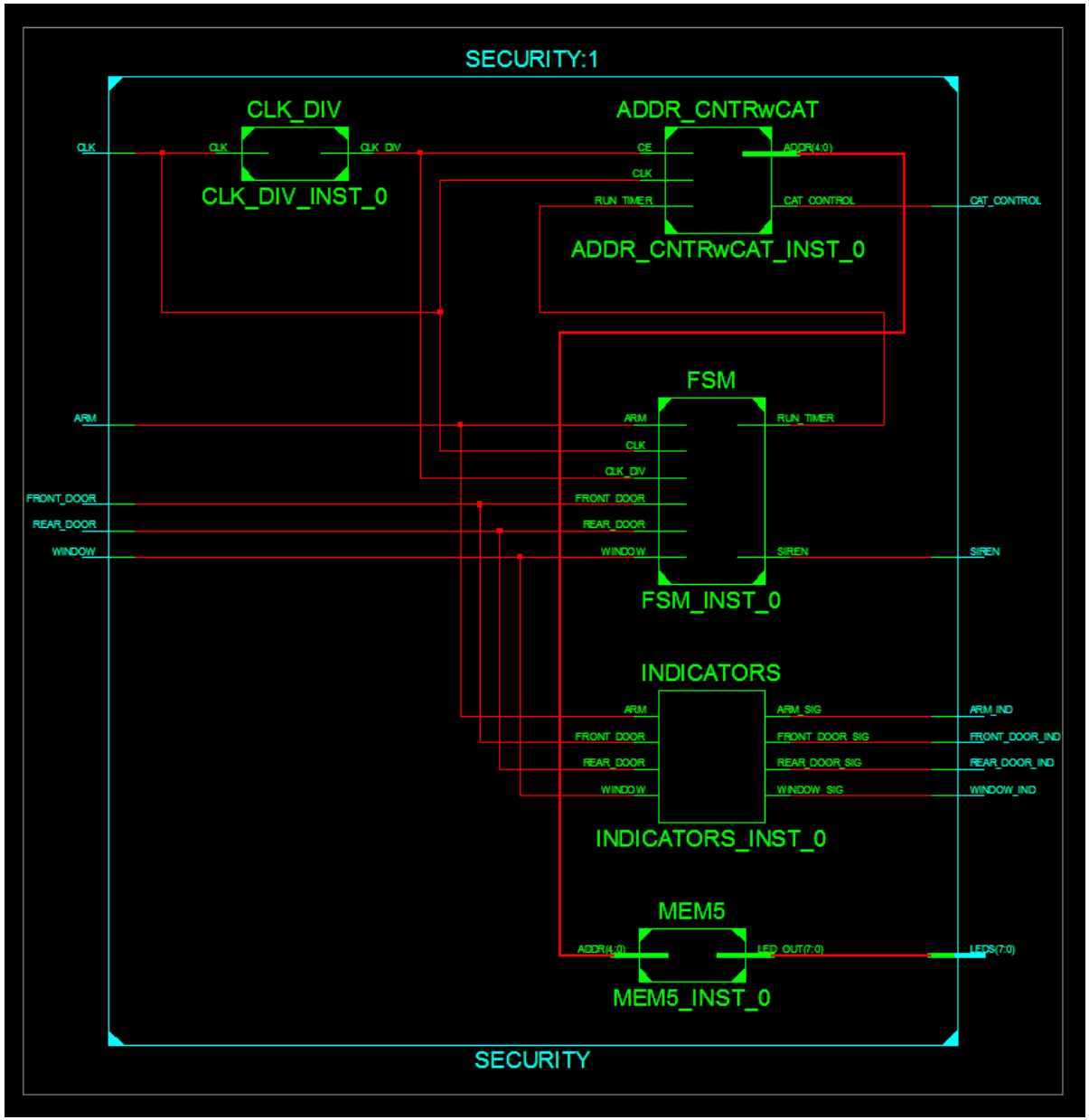
Figure 2. SECURITY SYSTEM STATE DIAGRAM

Objectives & Learning Outcomes

After completing this module, you will be able to:

- Create a top-level HDL structure
- Write VHDL code to describe a synchronous state machine diagram
- Utilize LED_DRIVER module developed in previous lab for timer display
- Create / use a constraint file for pin-out assignments
- Download to the SPARTAN 3E demo board for in-circuit verification.

Security System Block Diagram



Creating the Top-Level Module (SECURITY.vhd)

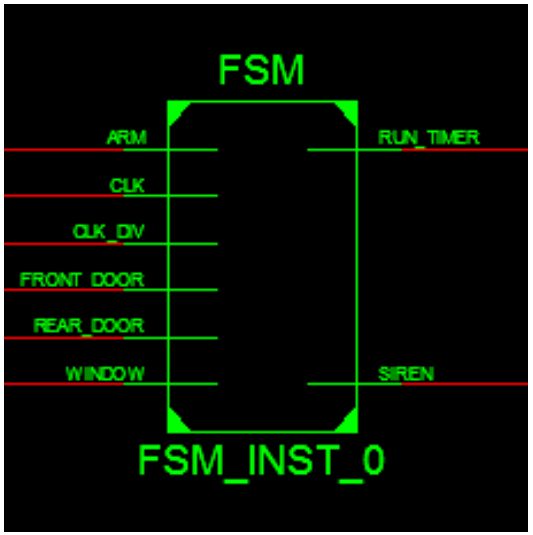
```
-- Module Name:    SECURITY.vhd
-- Project Name:   Security State Machine
-- Target Devices: Xilinx Spartan 3E / xc3s500e
-- Tool versions:  Win7 / Xilinx ISE 12.1

-- Description: Sample solution for Security State Machine

--       * This is the top level module that pulls together all sub modules
--       * use LED Drivers developed in previous lab for Timer display

--                SECURITY.vhd  (this module)
--                * CLK_DIV.vhd (Clock Divider)
--                * ADDR_CNTRwCAT.vhd (Address Counter with CAT control)
--                * MEM5.vhd (Memory for 2 digits 7-segment LEDs Display
--                * FSM.vhd (State Machine)
--                * INDICATORS.vhd (Onboard LED Indicators)
--                * SECURITY.ucf (I/O Assignments)
```

Create Entity SECURITY

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SECURITY is
port (
                CLK : in std_logic;
                ARM : in std_logic;
                FRONT_DOOR: in std_logic;
                REAR_DOOR: in std_logic;
                WINDOW: in std_logic;
                SIREN : out std_logic;

                ARM_IND : out std_logic;
                FRONT_DOOR_IND: out std_logic;
                REAR_DOOR_IND: out std_logic;
                WINDOW_IND: out std_logic;

                LEDS : out  STD_LOGIC_VECTOR (7 downto 0);
                CAT_CONTROL :out  STD_LOGIC
                );
end entity SECURITY;
```

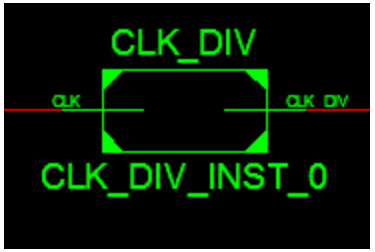
Architecture & Components

```
architecture RTL of SECURITY is

component FSM
    Port (        CLK : in  STD_LOGIC;
                  CLK_DIV:in  STD_LOGIC;
                  ARM : in  STD_LOGIC;
                  FRONT_DOOR : in  STD_LOGIC;
                  REAR_DOOR : in  STD_LOGIC;
                  WINDOW : in  STD_LOGIC;
                  RUN_TIMER:out  STD_LOGIC;
                  SIREN : out  STD_LOGIC);
end component;
```

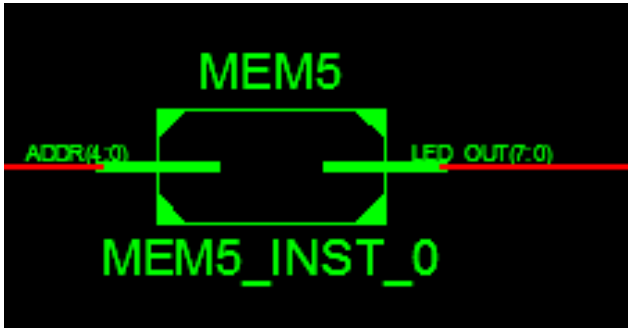


Components CLK_DIV & MEM5

```
component CLK_DIV
        Port (   CLK: in STD_LOGIC;
                 CLK_DIV: out STD_LOGIC);
end component;
```

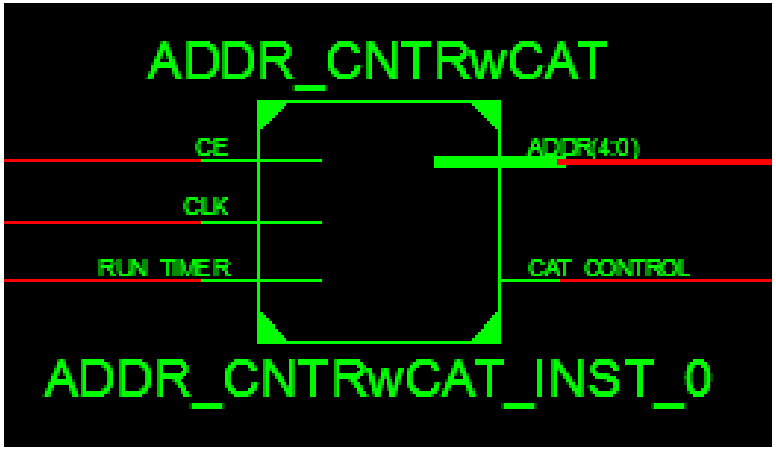



```
component MEM5
        Port (   ADDR: in std_logic_vector (4 downto 0);
                 LED_OUT: out STD_LOGIC_VECTOR(7 downto 0));

end component;
```



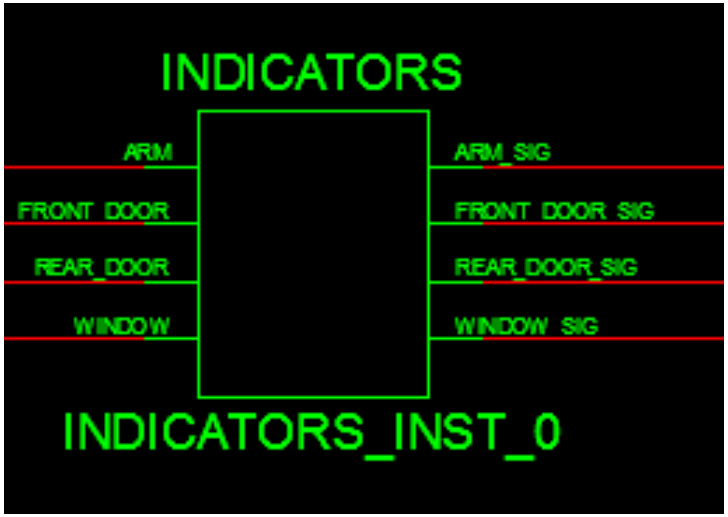
Component: ADDR_CNTRwCAT

```
component ADDR_CNTRwCAT
        port (   CLK: in STD_LOGIC;
                 RUN_TIMER: in STD_LOGIC;
                 CE: in std_logic;
                 ADDR: out STD_LOGIC_VECTOR(4 downto 0);
                 CAT_CONTROL: out std_logic
                 );
end component;
```



Component: INDICATORS

```
component INDICATORS
    Port ( ARM : in  STD_LOGIC;
           FRONT_DOOR : in STD_LOGIC;
           REAR_DOOR : in STD_LOGIC;
           WINDOW : in STD_LOGIC;

           ARM_SIG : out STD_LOGIC;
           FRONT_DOOR_SIG : out STD_LOGIC;
           REAR_DOOR_SIG : out STD_LOGIC;
           WINDOW_SIG : out STD_LOGIC);
end component;
```

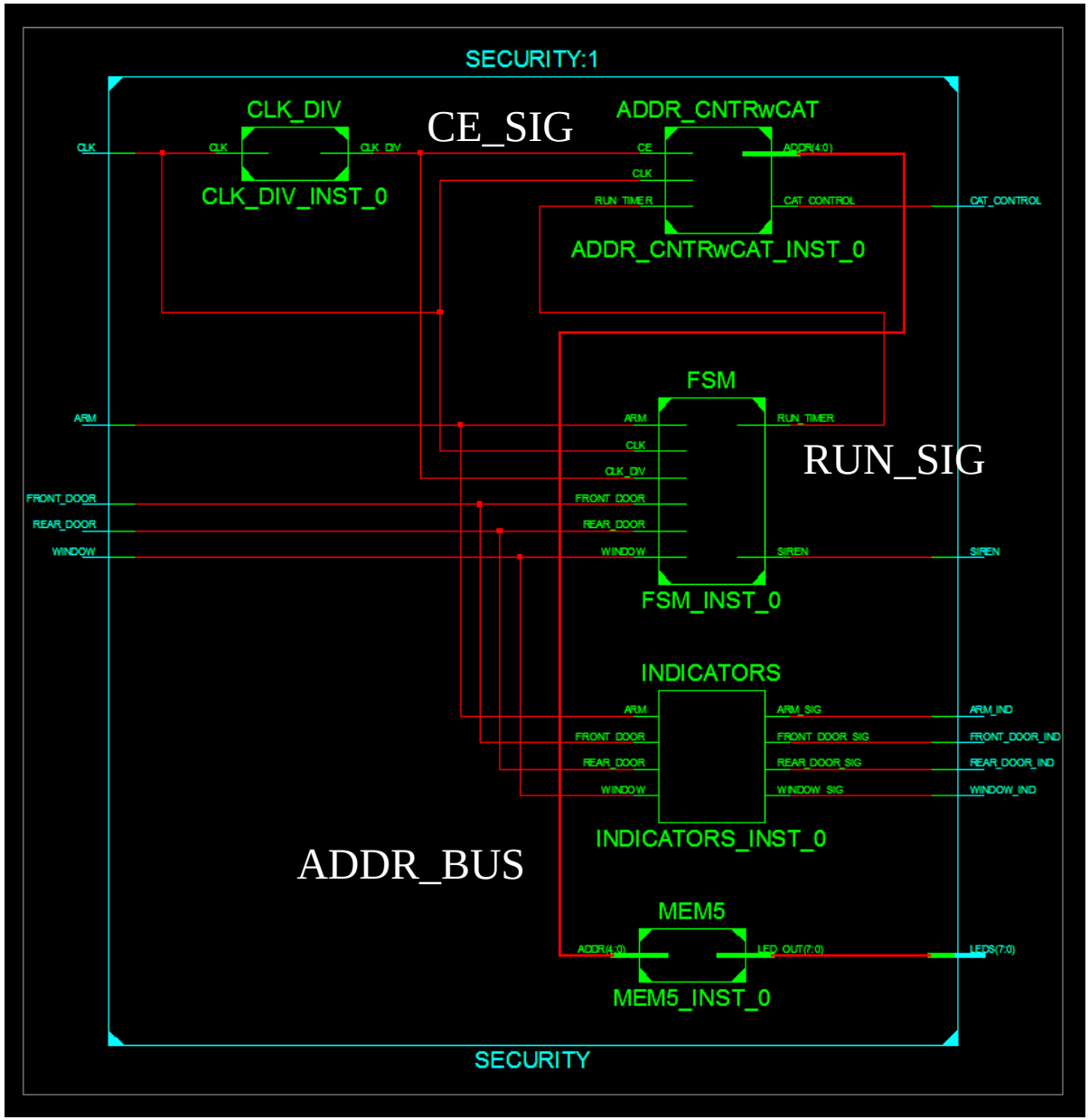


Signal provides interconnect between Components

```
signal CE_SIG: std_logic;

signal ADDR_BUS: std_logic_vector(4 downto 0);

signal RUN_SIG: std_logic;
```

See System Block Diagram (next slide)

Signal provides interconnect between Components



Create Instance of Component INDICATORS

```
component INDICATORS
    Port ( ARM : in  STD_LOGIC;
           FRONT_DOOR : in STD_LOGIC;
           REAR_DOOR : in STD_LOGIC;
           WINDOW : in STD_LOGIC;

           ARM_SIG : out STD_LOGIC;
           FRONT_DOOR_SIG : out STD_LOGIC;
           REAR_DOOR_SIG : out STD_LOGIC;
           WINDOW_SIG : out STD_LOGIC);
end component;

INDICATORS_INST_0: INDICATORS port map(
                   ARM => ARM,
                   FRONT_DOOR => FRONT_DOOR,
                   REAR_DOOR =>  REAR_DOOR ,
                   WINDOW =>  WINDOW,

                   ARM_SIG => ARM_IND,
                   FRONT_DOOR_SIG =>  FRONT_DOOR_IND,
                   REAR_DOOR_SIG => REAR_DOOR_IND,
                   WINDOW_SIG => WINDOW_IND
                   );

-- Left hand side includes I/Os at the component ports
-- Right hand side includes signals that the ports are attached to connect
to other components
```

Create Instances of Components: CLK_DIV, ADDR_CNTRwCAT, MEM5, FSM

```
CLK_DIV_INST_0: CLK_DIV  port map (CLK=>CLK, CLK_DIV=>CE_SIG);

ADDR_CNTRwCAT_INST_0: ADDR_CNTRwCAT port map (
                CLK=>CLK,
                RUN_TIMER => RUN_SIG,
                CE=>CE_SIG,
                ADDR => ADDR_BUS,
                CAT_CONTROL=>CAT_CONTROL);

MEM5_INST_0: MEM5 port map (ADDR=> ADDR_BUS, LED_OUT=>LEDS);

FSM_INST_0: FSM port map (
        CLK => CLK,
        CLK_DIV => CE_SIG,
        ARM => ARM,
        FRONT_DOOR => FRONT_DOOR,
        REAR_DOOR => REAR_DOOR,
        WINDOW => WINDOW,
        RUN_TIMER => RUN_SIG,
        SIREN => SIREN);
```

SECURITY Architecture Syntax Summary

```
architecture RTL of SECURITY is

component FSM    …
component CLK_DIV …
component MEM5   …
component ADDR_CNTRwCAT …
component INDICATORS …

signal CE_SIG: std_logic;
signal ADDR_BUS: std_logic_vector(4 downto 0);
signal RUN_SIG: std_logic;

Begin

INDICATORS_INST_0: …
CLK_DIV_INST_0: …
ADDR_CNTRwCAT_INST_0: …
MEM5_INST_0: …
FSM_INST_0: …

end architecture RTL;
```

Hands-on (30 minutes)

Step 1 / Lab procedure

Creating the Top-Level Module SECURITY.vhd

using Xilinx ISE

(see sample screen shot)

Finite State Machine Module: FSM.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FSM is
    Port (       CLK : in  STD_LOGIC;
                 CLK_DIV:in  STD_LOGIC;
                 ARM : in  STD_LOGIC;
                 FRONT_DOOR : in  STD_LOGIC;
                 REAR_DOOR : in  STD_LOGIC;
                 WINDOW : in  STD_LOGIC;
                 RUN_TIMER:out  STD_LOGIC;
                 SIREN : out  STD_LOGIC);
end FSM;

architecture Behavioral of FSM is

type SECURITY_STATE is (ARMED,DISARMED,WAIT_DELAY, ALARM);
signal CURR_STATE,NEXT_STATE: SECURITY_STATE;

signal START_COUNT,COUNT_DONE: std_logic;
signal SENSORS:std_logic_vector (2 downto 0); --combine inputs

signal TIMER_CNTR: std_logic_vector (2 downto 0) := (others => '0');
```

FSM / SYNC process

```
Begin

SENSORS <= FRONT_DOOR & REAR_DOOR & WINDOW; -- &: concatenate

SYNC: process (CLK,ARM)
begin
        if ARM = '0' then
                CURR_STATE <= DISARMED;
        elsif rising_edge (CLK) then
                CURR_STATE <= NEXT_STATE;
        end if;

end process SYNC;
```

```
STATE_MACHINE: process (CURR_STATE,SENSORS,ARM,COUNT_DONE)
begin
            START_COUNT <= '0'; -- establish default
            case (CURR_STATE) is
                        when DISARMED =>
                                    if ARM = '1' then
                                    NEXT_STATE <= ARMED;
                                    else NEXT_STATE <= DISARMED;
                                    end if;
                                    -- Output:
                                    SIREN <= '0';
                                    RUN_TIMER <= '0';
                        when ARMED =>
                                    if (SENSORS /= "000") then
                                    NEXT_STATE <= WAIT_DELAY;
                                    else NEXT_STATE <= ARMED;
                                    end if;
                                    -- Output:
                                    SIREN <= '0';
                                    RUN_TIMER <= '0';
                        when WAIT_DELAY =>
                                    START_COUNT <= '1';
                                    if (COUNT_DONE = '1') then
                                    NEXT_STATE <= ALARM;
                                    elsif (ARM ='0') then
                                                NEXT_STATE <= DISARMED;
                                    else NEXT_STATE <= WAIT_DELAY;
                                    end if;
                                    -- Output:
                                    SIREN <= '0';
                                    RUN_TIMER <= '1';
                        when ALARM =>
                                    if (ARM = '0') then
                                    NEXT_STATE <= DISARMED;
                                    else NEXT_STATE <= ALARM;
                                    end if;
                                    -- Output:
                                    SIREN <= '1';
                                    RUN_TIMER <= '0';
            end case;
end process STATE_MACHINE;
```

DELAY_TIMER process

```
DELAY_TIMER: process(CLK_DIV,CURR_STATE,START_COUNT,TIMER_CNTR)

begin
        COUNT_DONE <= '0';  -- default value
        if (rising_edge (CLK_DIV) and (START_COUNT = '1')) then
                TIMER_CNTR <= TIMER_CNTR + 1;

        end if; --rising_edge (CLK_DIV)

        if (CURR_STATE/=WAIT_DELAY) then  -- /= : NOT equal
                TIMER_CNTR <= "000";
        end if;

        if (TIMER_CNTR = "111") then
                COUNT_DONE <= '1';
        end if;

end process DELAY_TIMER;
```

FSM.vhd Syntax Summary

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FSM is
    Port (        …         );
end FSM;

architecture Behavioral of FSM is
type …
signal …

Begin

SYNC: process …
STATE_MACHINE: process …
DELAY_TIMER: process …

end Behavioral;
```

Hands-on (30 minutes)

Step 2 / Lab Procedure
Create FSM.vhd using Xilinx ISE
(see sample screen shot)

Hands-on (30 minutes)

Step 3 / Lab Procedure
Implementing and Downloading the Design to Spartan 3E

Hands-on (15 minutes)

Step 4 / Lab Procedure
System Verification

Hands-on (Bonus Round)

Step 5: (Bonus Round)

If time is available, modify the source files to reflect the following new system requirements:

- The REAR_DOOR sensor is replaced by FIRE sensor.
- A new output SPRINKLER is created to control a sprinkler system.

In addition to the previous requirements, when FIRE is detected, Siren and Sprinkler system are activated immediately without time delay.

Resource for Instructors

(Resources will be available for Summer 2010 FPGA Workshop)

- SECURITY.vhd (Top-Level module that ties the following sub modules together)

- FSM.vhd (Security State Machine)
- INDICATORS.vhd (Spartan 3E Onboard LED Indicators)

- CLK_DIV.vhd (Divide system clock from 50 MHz to 1.49 Hz for timer display)
- ADDR_CNTRwCAT.vhd (Address Counter with scan signal for 7-segment LEDs)
- MEM5.vhd (Memory for 2-digit 7-segment LED display)

- SECURITY.ucf (I/O Assignments for Spartan 3E demo board)

Contrast TTL vs VHDL Experiment

TTL	VHDL/ISE Project Navigator
Method • Create State Excitation & Transition Table • Karnaugh Map for simplification • Create schematics from design equations	Method • Create Process using Hardware Description Language • Using Synthesize tool • Using Implement design tool
Implementation • TTL breadboard	Implementation • Download to target board using ISE Project Navigator
Time • 8 hours (including trouble shooting for wiring connections problem)	Time • 2 hours
Complexity • Limited (Karnaugh map gets complicated after 4 variables)	Complexity • open

HTWI
High Tech Workforce Initiative

Questions?

VHDL/FPGA Workshop

HTWI
High Tech Workforce Initiative

- Plan a workshop for Summer 2011.
- The workshop will be provided by NSF. Each participant will receive stipend for travel and lodging.
- If you are interested please email Bassam Matar at:

b.matar@cgcmail.maricopa.edu

HTWI
High Tech Workforce Initiative

Questions?

Help us become better

HTWI
High Tech Workforce Initiative

Please complete this quick 1 minute survey to help us become better and to let us know what webinars you would like to see in the future.

http://www.questionpro.com/t/ABkVkZIOXa

Webinar Recordings

HTWI
High Tech Workforce Initiative

To access this recording, visit
www.matecnetworks.org,
Keyword Search:
"**webinar HTWI FPGA**"

HTWI Upcoming Webinars

HTWI
High Tech Workforce Initiative

December 3: Supply Chain and Inventory Control: Limiting Factors and the Technologies That Help Them Work

Visit ***www.matecnetworks.org*** for more details about these and other upcoming webinars.

NetWorks Upcoming Webinars

HTWI
High Tech Workforce Initiative

October 8: Innovative STEM Resources

November 12: Electronics Education Today

Visit ***www.matecnetworks.org*** for more details about these and other upcoming webinars.

Thank you for attending Field Programmable Gate Array (FPGA) Curriculum update with Industry

HTWI
High Tech Workforce Initiative

MARICOPA COMMUNITY COLLEGES®

National Science Foundation
WHERE DISCOVERIES BEGIN

Hosted by MATEC NetWorks

Funded, in part by a grant from the
National Science Foundation
DUE 1003542