ENGR-304-L

Hardware Lab 01

Agenda

1. Attendance
2. Recap
3. FPGA & VHDL
4. Block Diagrams
5. Quartus
6. Getting Started

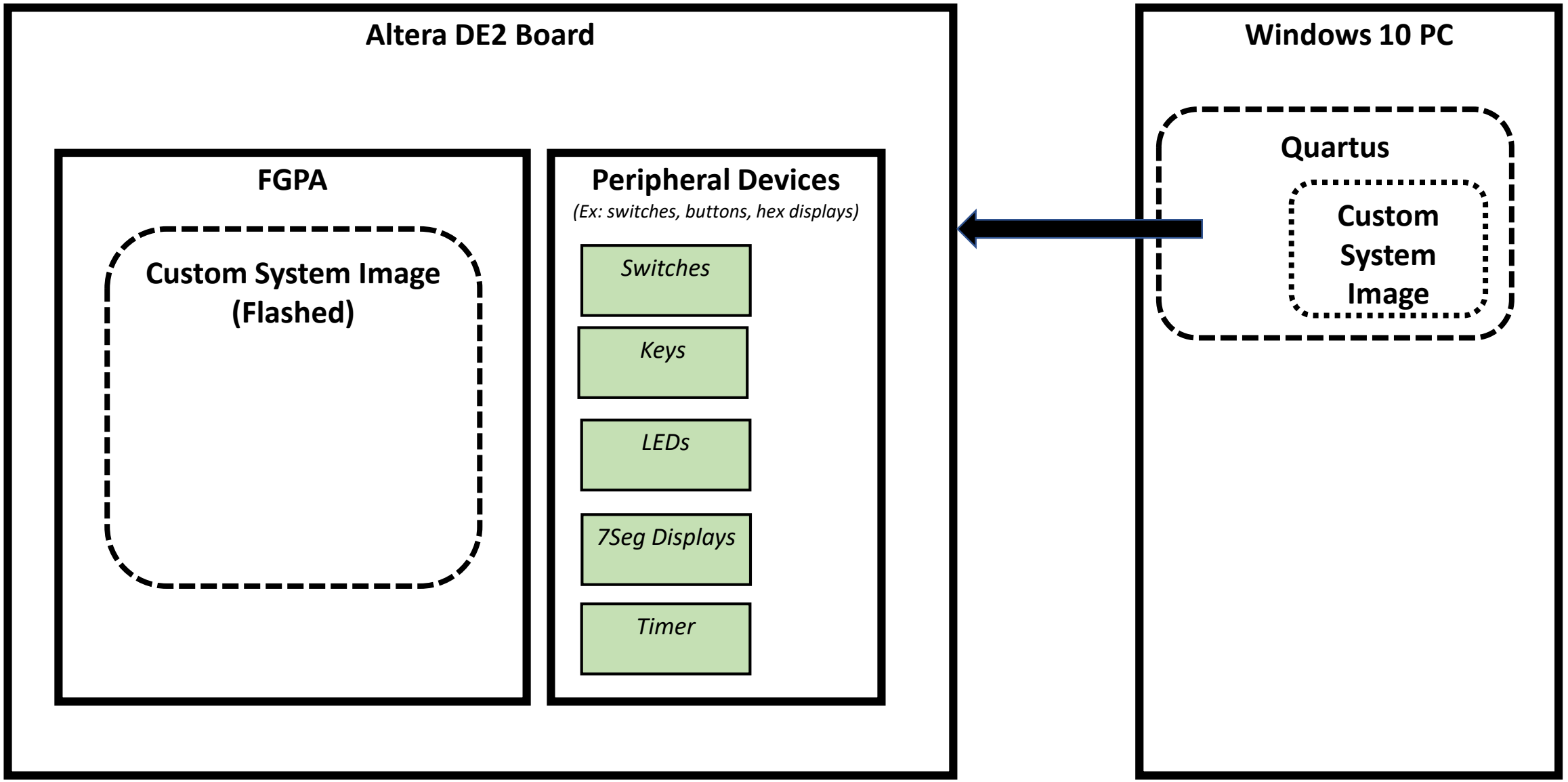
Recap

- IO Devices – Polling vs Interrupts
- The ISR & Main Routine
- Control Registers & New Instructions

FPGAs

- Field Programmable Gate Array Devices
- General purpose hardware configured as specific circuitry
- Hardware iterations are expensive
- Generic and re-flashable hardware opens many possibilities
  - Educational tools
  - Development tools
  - Nonphysical hardware upgrades
  - More!

Lab Components



VHDL

- VHSIC Hardware Definition Language
  - Very High Speed Integrated Circuit Hardware Definition Language
- “Circuitry as code”
- Syntactically similar to Ada
- Defines the intended logical behavior of the system hardware, used to flash generic FPGA with specific solution
- Tools convert VHDL into FPGA images, analogous to compilers
- VHDL is concurrent, logical behaviors are simultaneously applied

VHDL

- Entities:
  - A logical component containing a port of inputs & outputs and an architecture
- Ports:
  - Contain IO signals which may be of type std_logic, std_logic_vector, or others
  - std_logic refers to typical true/false, high/low, 1/0
- Architectures:
  - Describe the functionality of an entity, how its inputs become its outputs
- Signals:
  - Represent data to be communicated between entities or within an entity
  - std_logic signals have two possible logical values (true/false)
- Processes:
  - Assist in describing architectures by providing a means for some sequential definition (as opposed to VHDL's typically concurrent paradigm)

VHDL

- Example VHDL template
- Resources
  - http://vhdl.renerta.com/mobile/index.html
  - http://www.people.vcu.edu/~rhklenke/tutorials/vhdl/modules/m12_23/sld001.htm
  - http://gmvhdl.com/VHDL.html

```
--------------------------------------------------------------------------------
-- ENTITY: HWLab1
--------------------------------------------------------------------------------
ENTITY HWLab1 IS
  PORT (
    SW : IN std_logic_vector(15 downto 0);
    HEX0 : OUT std_logic_vector(6 downto 0);
    HEX1 : OUT std_logic_vector(6 downto 0);
    HEX2 : OUT std_logic_vector(6 downto 0);
    HEX3 : OUT std_logic_vector(6 downto 0);
    HEX4 : OUT std_logic_vector(6 downto 0);
    HEX5 : OUT std_logic_vector(6 downto 0);
    HEX6 : OUT std_logic_vector(6 downto 0);
    HEX7 : OUT std_logic_vector(6 downto 0)
  );
END HWLab1;

--------------------------------------------------------------------------------
-- ARCHITECTURE: behav
-- This architecture is implemented with behavioural VHDL
--------------------------------------------------------------------------------
ARCHITECTURE behav OF HWLab1 IS
  --- This area is used to define types and any internal signals
  --- Fill_In as necessary to define signals like "Result", etc
  
BEGIN
  --- concurrent signal assignments placed here
  --- Assign internal signals to external port names (e.g. SW15..8 = InputB)

  --- use the conversion function to display the least signif 4-bits of Result
  HEX0 <= convert_to_7seg(Result(3 downto 0));
  --- Fill_In as necessary to drive the remaining HEX displays
  
  --- use "others" to drive busses of wires (e.g. turning off hex displays)
  HEX7 <= (others => '1');

process (Fill_In) is
BEGIN
    -- Fill_In

end process;

END behav;
```

VHDL

- MUX as a logical component
  - Pass through one of two inputs based on a third input

A	B	S	F
0	d	0	0
d	0	1	0
1	d	0	1
d	1	1	1

- 4 ways to represent a MUX in VHDL…

MUX – Selected Signal Assignment

```
entity Multiplexer is
    port( A, B, S : IN  BIT ;
          F       : OUT BIT ) ;
end Multiplexer ;

architecture SelectedSignal of Multiplexer is
begin
    with S select
        F <= A when'0',
             B when OTHERS ;
end SelectedSignal ;
```

MUX – Conditional Signal Assignment

```
entity Multiplexer is
    port( A, B, S : IN  BIT ;
          F       : OUT BIT ) ;
end Multiplexer ;

architecture ConditionalSignal of Multiplexer is
begin
    F <= A when S = '0' else
         B ;
end ConditionalSignal ;
```

MUX – Process If-Else

```
entity Multiplexer is
       port( A, B, S : IN   BIT ;
                F           : OUT BIT ) ;
end Multiplexer ;
architecture ProcessIfElse of Multiplexer is
begin
        process( A, B, S )
        begin
              if S = '0' THEN
                    F <= A ;
              else
                    F <= B ;
              end if ;
       end process;
end ProcessIfElse ;
```

MUX – Process Case Statement

```
entity Multiplexer is
       port( A, B, S : IN   BIT ;
             F              : OUT BIT ) ;
end Multiplexer ;
architecture ProcessCase of Multiplexer is
begin
       process( A, B, S )
        begin
              case S IS
                    when '0' =>     F <= A ;
                    when OTHERS =>  F <= B ;
              end case ;
       end process;
end ProcessCase ;
```

Block Diagrams

Use these in senior design

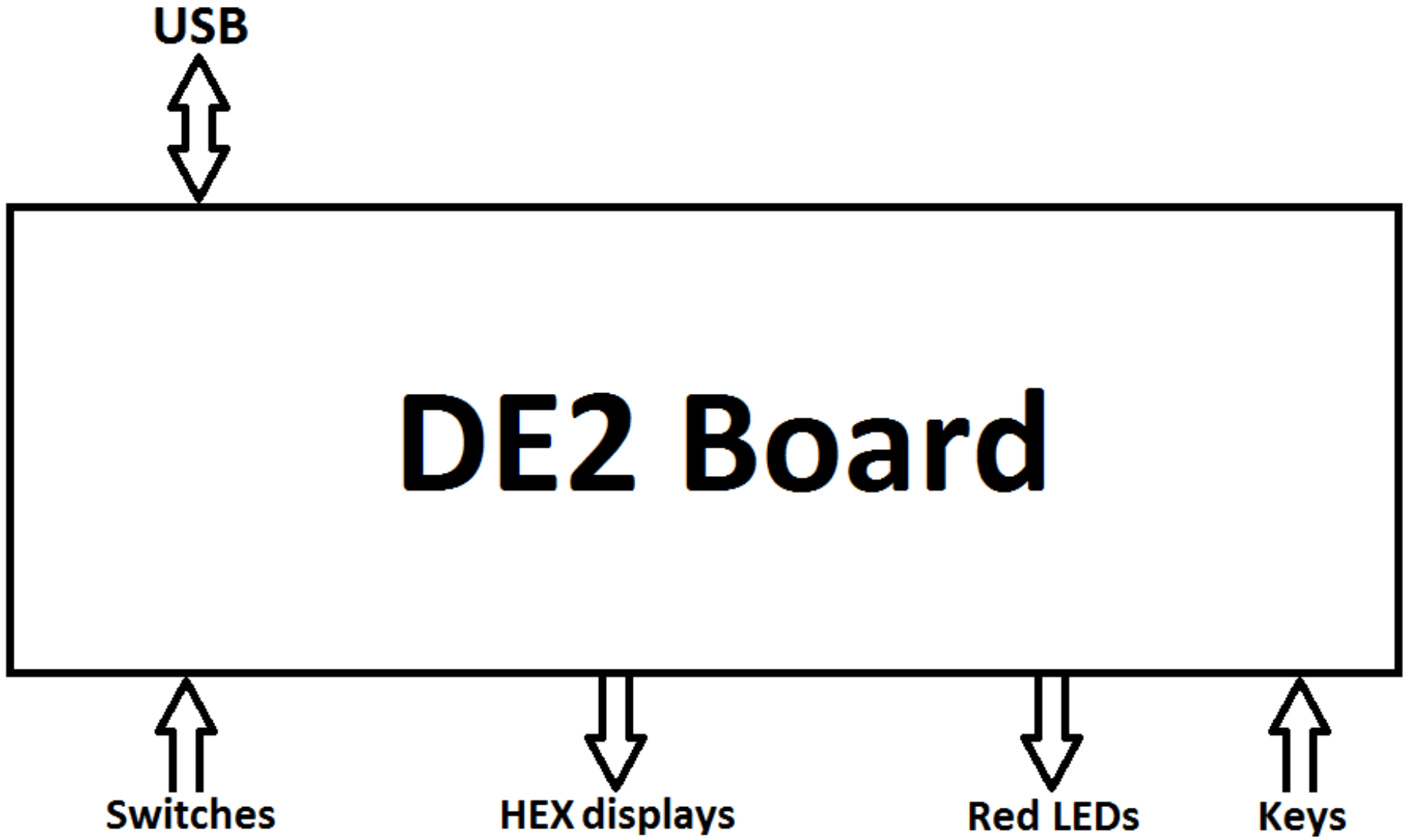
- Systems Engineering vs EE & CE
- Block diagrams as a tool:
  - Assist in communication
  - Facilitate design of complex solutions
  - May act as documentation of the design
  - Are often multi-leveled
  - Encourage modularization, simulation, and other best practices
  - Are a fundamental component of MBSE
  - May act as or support design requirements
  - Provide the context of a given component, its role in the big picture
  - … more

Block Diagrams

- Depictive syntax ranges from ‘back of the napkin’ to formalized industry standards
- Levels of decomposition
- Level 0:
  - The system being diagrammed, its inputs, its outputs, its functionality
- Level 1:
  - Primary sub-systems, their connections, their respective functionalities
- Level 2:
  - Secondary sub-systems, their connections, their respective functionalities
- Level N…

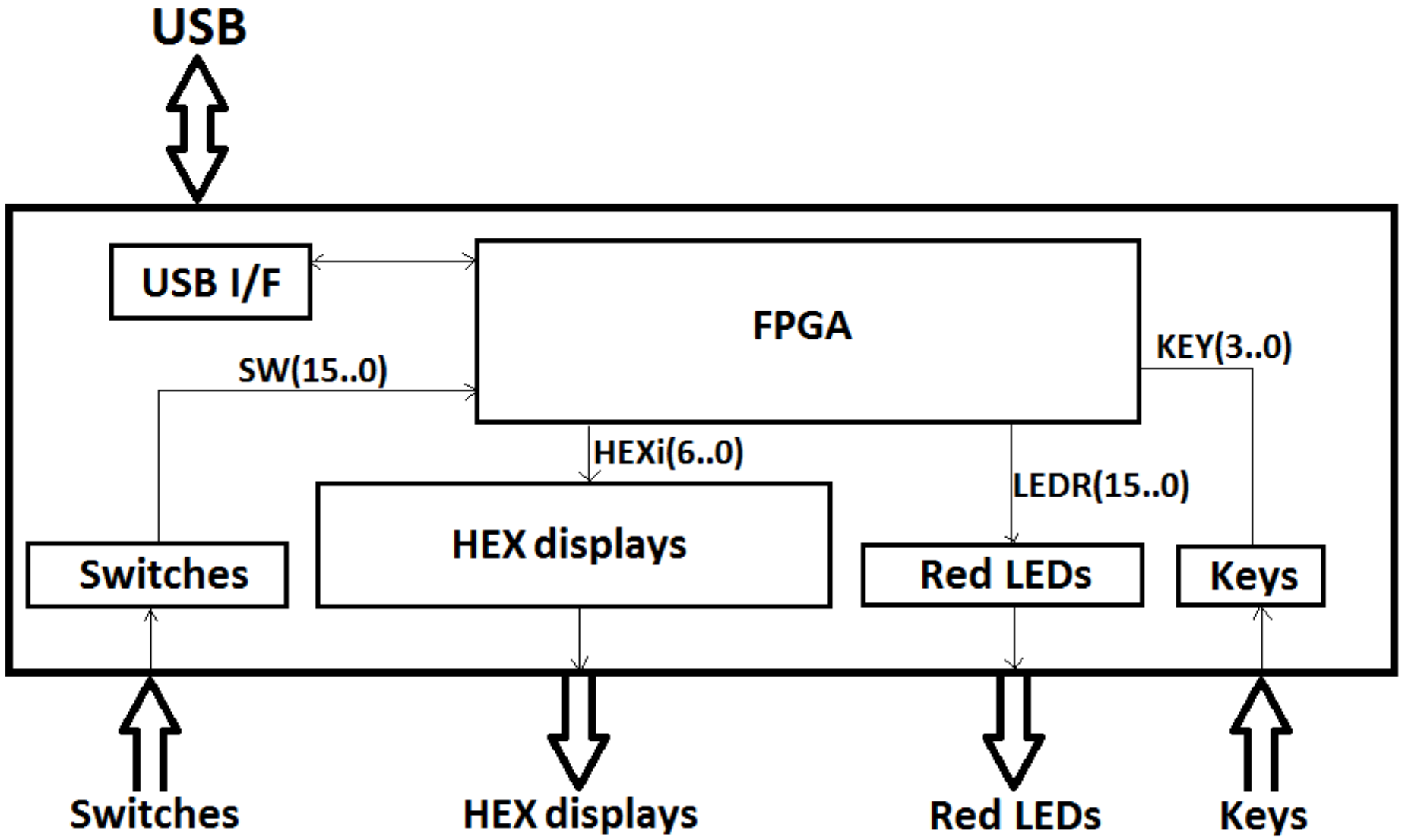
Block Diagram Example – DE2 & NOIS CPU

- Level 0



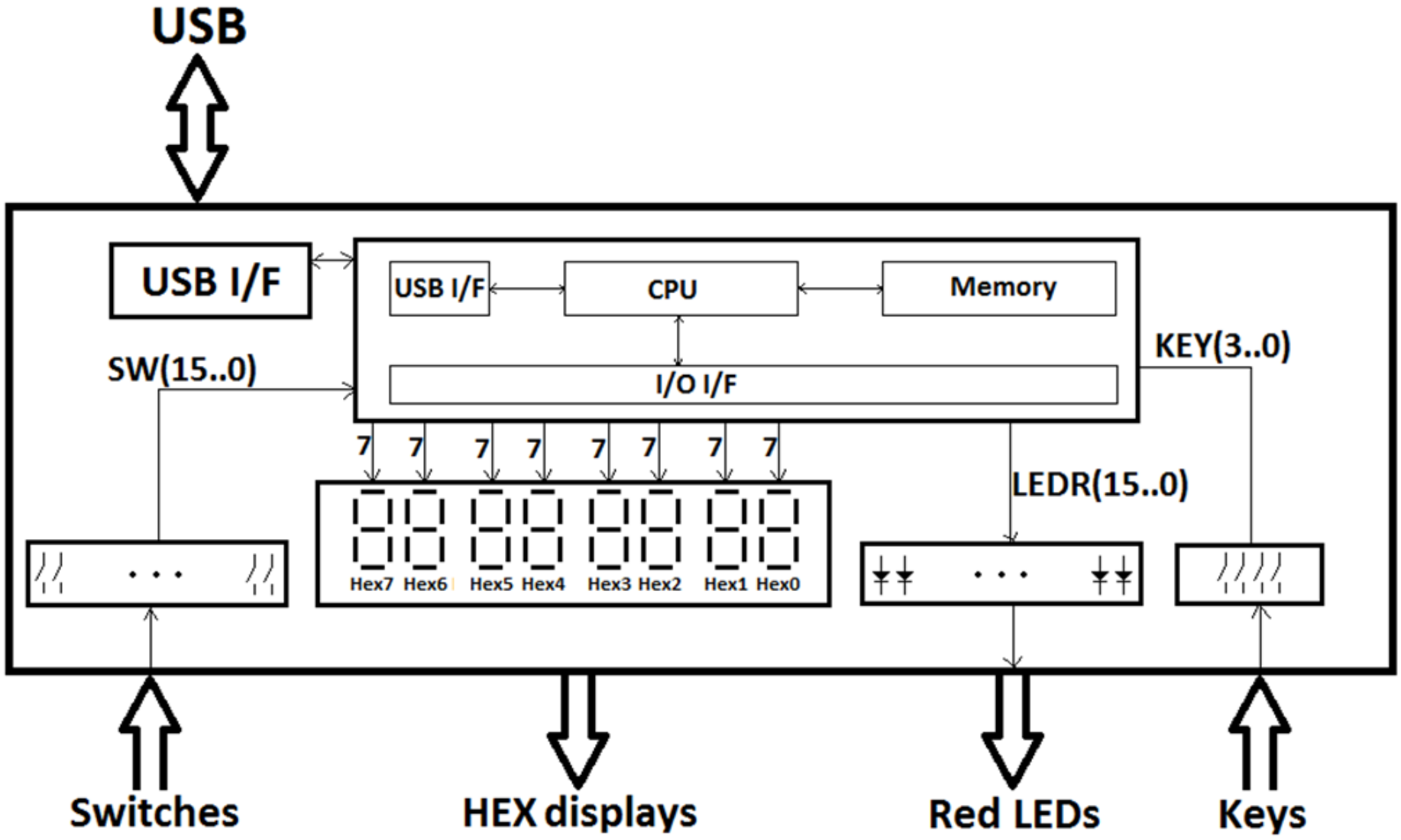
Block Diagram Example – DE2 & NOIS CPU

- Level 1



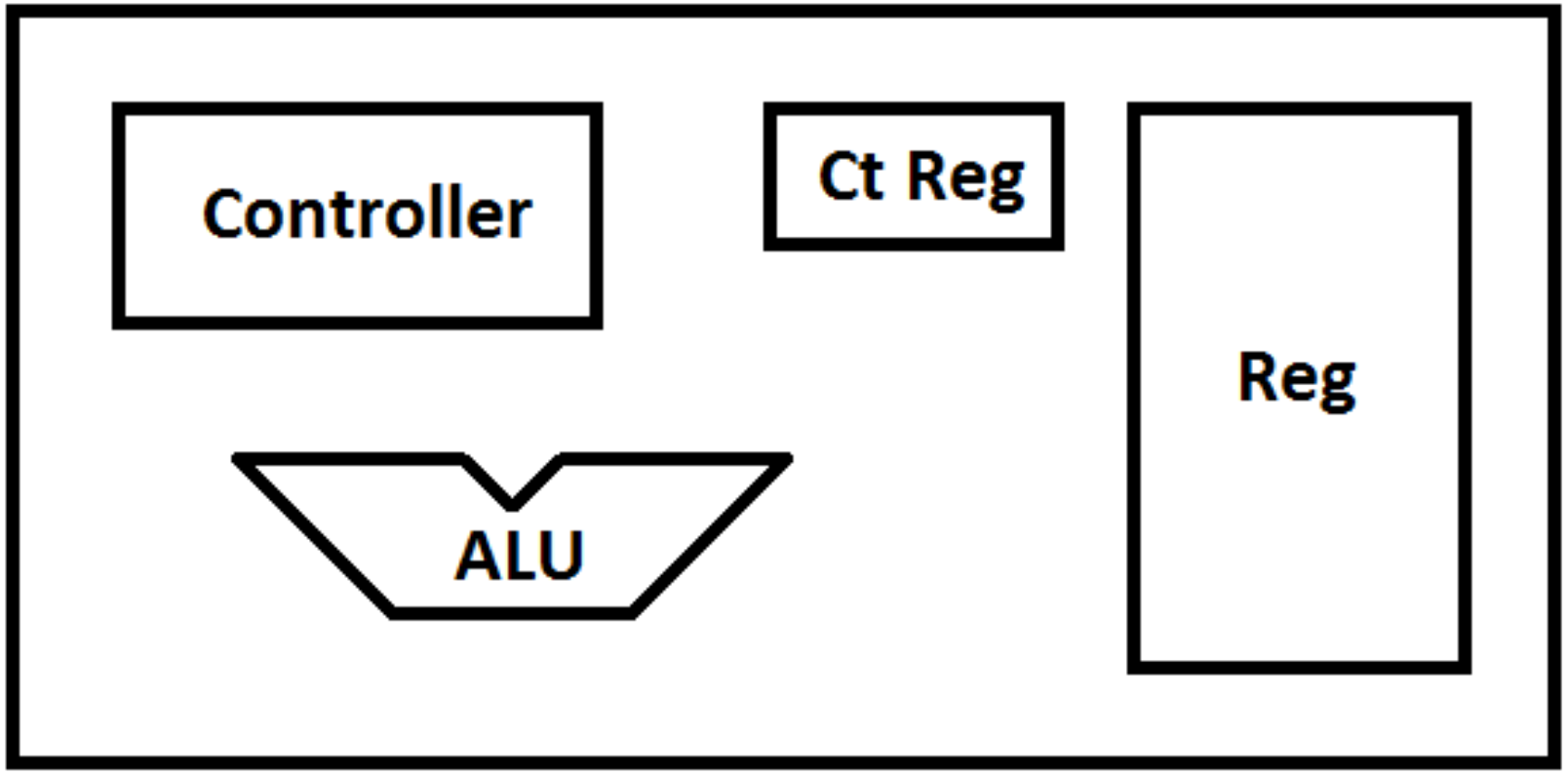
Block Diagram Example – DE2 & NOIS CPU

- Level 2



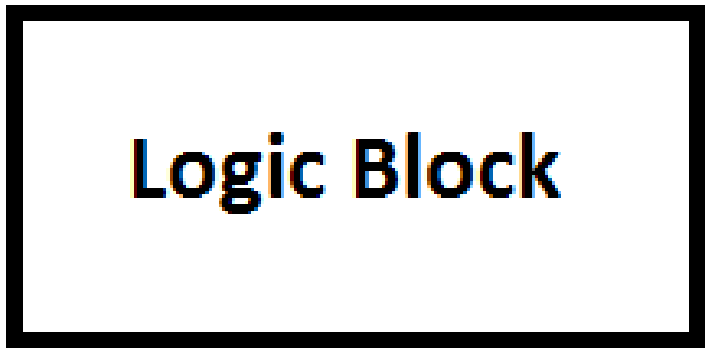
Block Diagram Example – DE2 & NOIS CPU

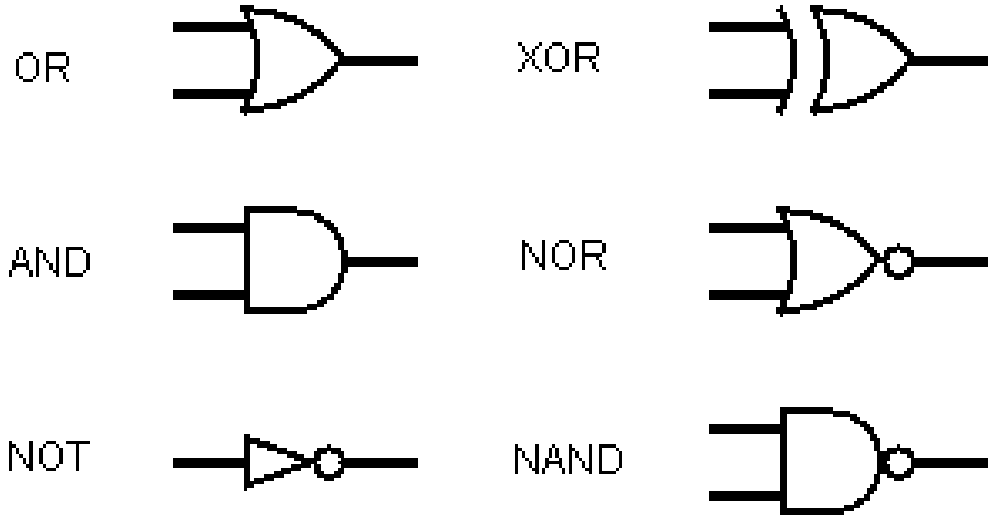
- Level 3
  - Simplified & Only CPU shown



Block Diagrams

- Common block diagram symbols for Engr304

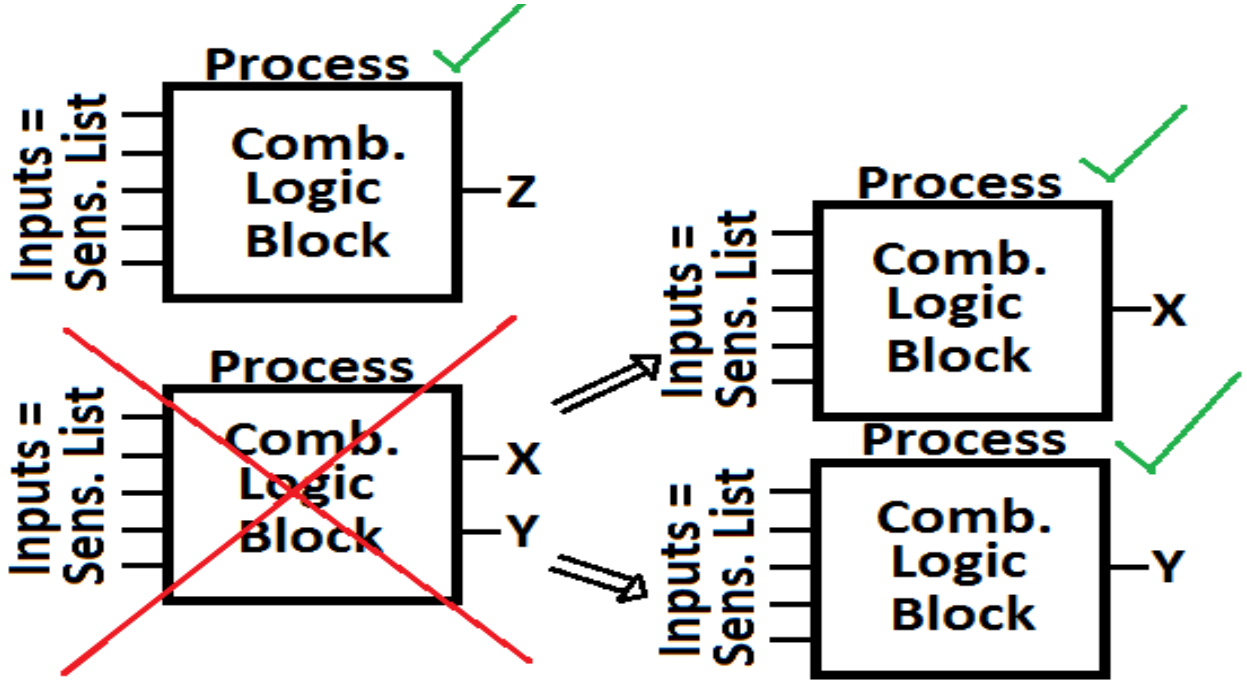




http://www.multiwingspan.co.uk/as2.php?page=logic

Block Diagrams

- VHDL Processes & Block Diagrams
  - A VHDL process can implement the functionality described by a component in a block diagram
  - A VHDL process block should have just 1 output, maybe 2 in very simple cases



Quartus

- Quartus can build FPGA images from VHDL definitions
- Images are specific to a particular device, be sure to select carefully
- DE2 FPGAs have fixed pins that are connected to peripheral devices, these must be known ‘ahead of time’ by Quartus to generate an FPGA image that correctly associates the fixed pins to the custom VHDL signals

ModelSim

- We will use ModelSim to exercise the system solution on a provided test bench
- The test bench drives combinations of inputs into the system at particular times and expects the correct outputs to emerge in response
- Often, a test bench and system will be independently developed based on the same requirements. Then, if the system passes the test bench's tests, there can be a high confidence that it correctly implements the requirements.

Lab HW01 Tips

- Think about the different components available for this lab, note that they are all onboard the DE2 board but not all within the FPGA
- Consider the difference in the input and output signals, a simple way to state the goal of the system is ‘make the inputs into the outputs’
- Think about what conversions or intermediate steps are needed to ‘make the inputs into the outputs’, based on the described requirements of the lab
- Draw & label signals conceptually first, and add detail about data representation, bit counts, etc. later
- Add your own logical blocks or other commonly used components
- Note: VHDL convert_to_7seg() function is provided, a logical block(s)

Getting Started

1. Setup project directory for HW01 on the **H: drive**
2. Download HW01 files from Moodle or S: drive
3. Start following directions in the lab assignment document