Block Diagrams

Engineering 304 Lab

Outline

- Overview of uses
- Example DE2 Block Diagram
  - Level 0
  - Level 1
  - Level 2

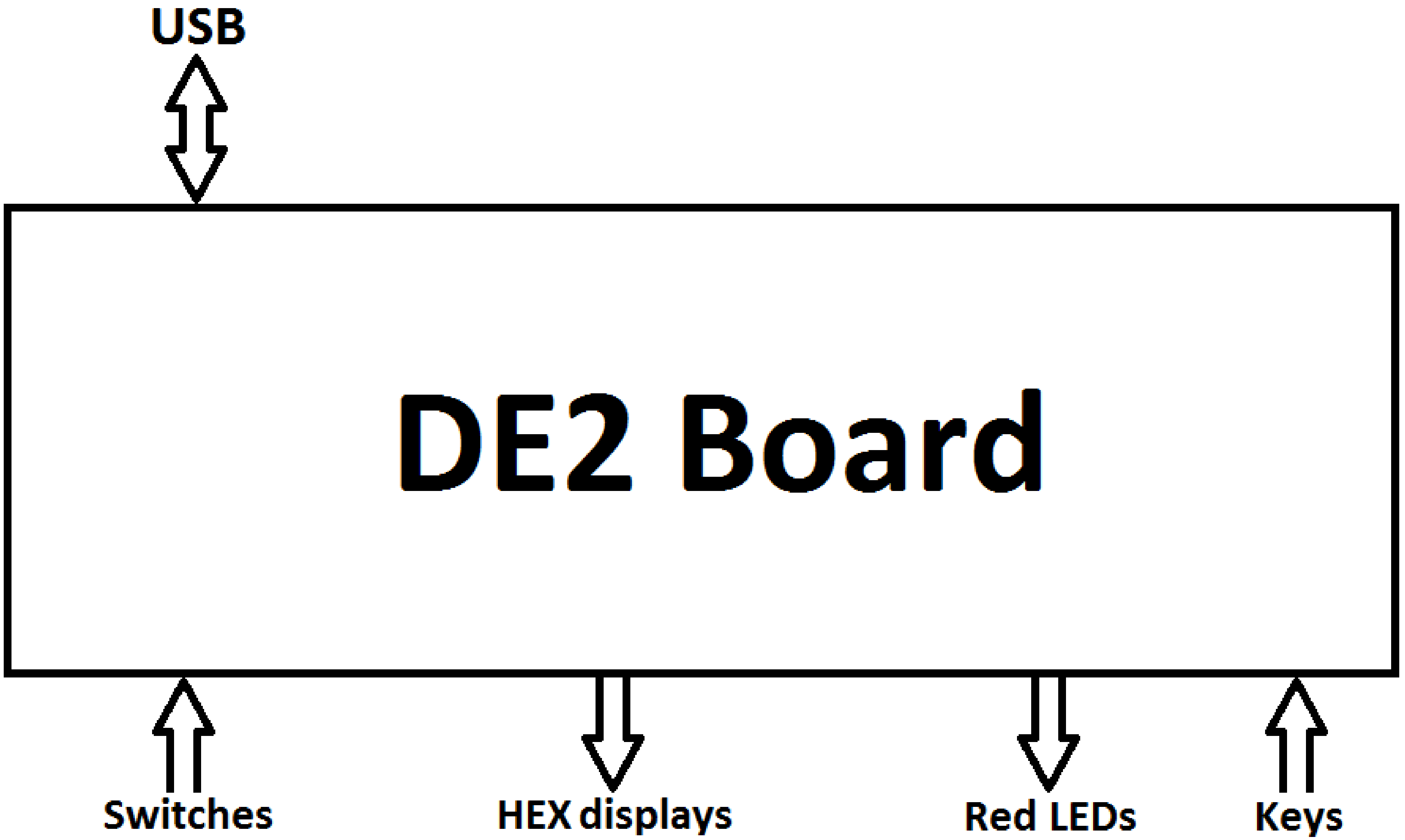
Why Use Block Diagrams?

- System Engineering is how you design a system with multiple components that interact
  - Much of your work in the future will be a form of system engineering
    - Less focus on the low level components
    - Move focus on how components connect together
- Good tool for communicating your ideas
- Helpful for dividing and conquering a complex design
  - E.g. senior design project

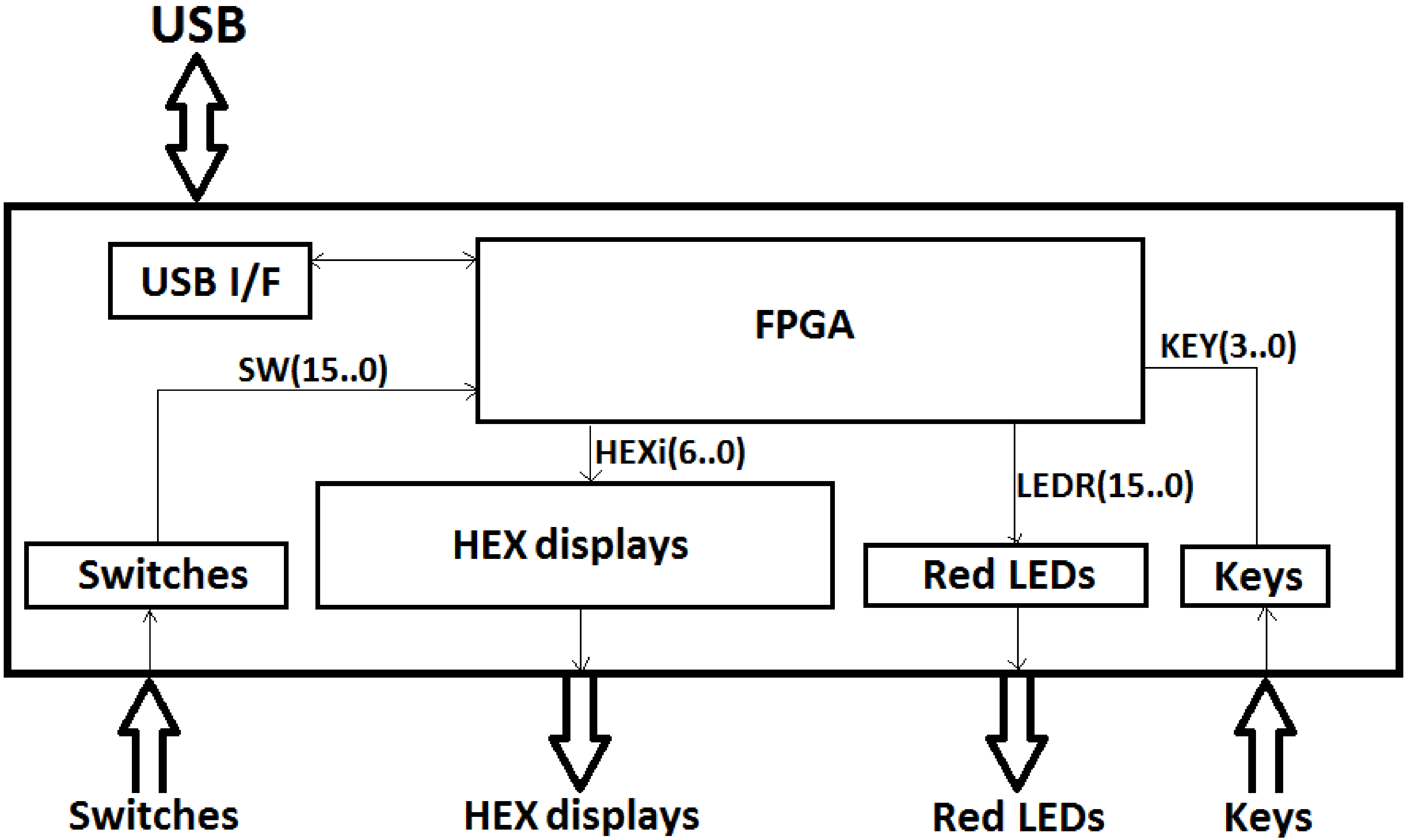
What are the different levels?

- Level 0
  - Identifies system inputs, outputs, and function
- Level 1
  - Identifies primary sub-systems, connections between them, and functionality of each
- Level 2
  - For each primary sub-system, secondary sub-systems are identified.
- Level n
  - Continues until you reach the lowest component level desired
- Example follows for the DE2 board with a NIOS CPU

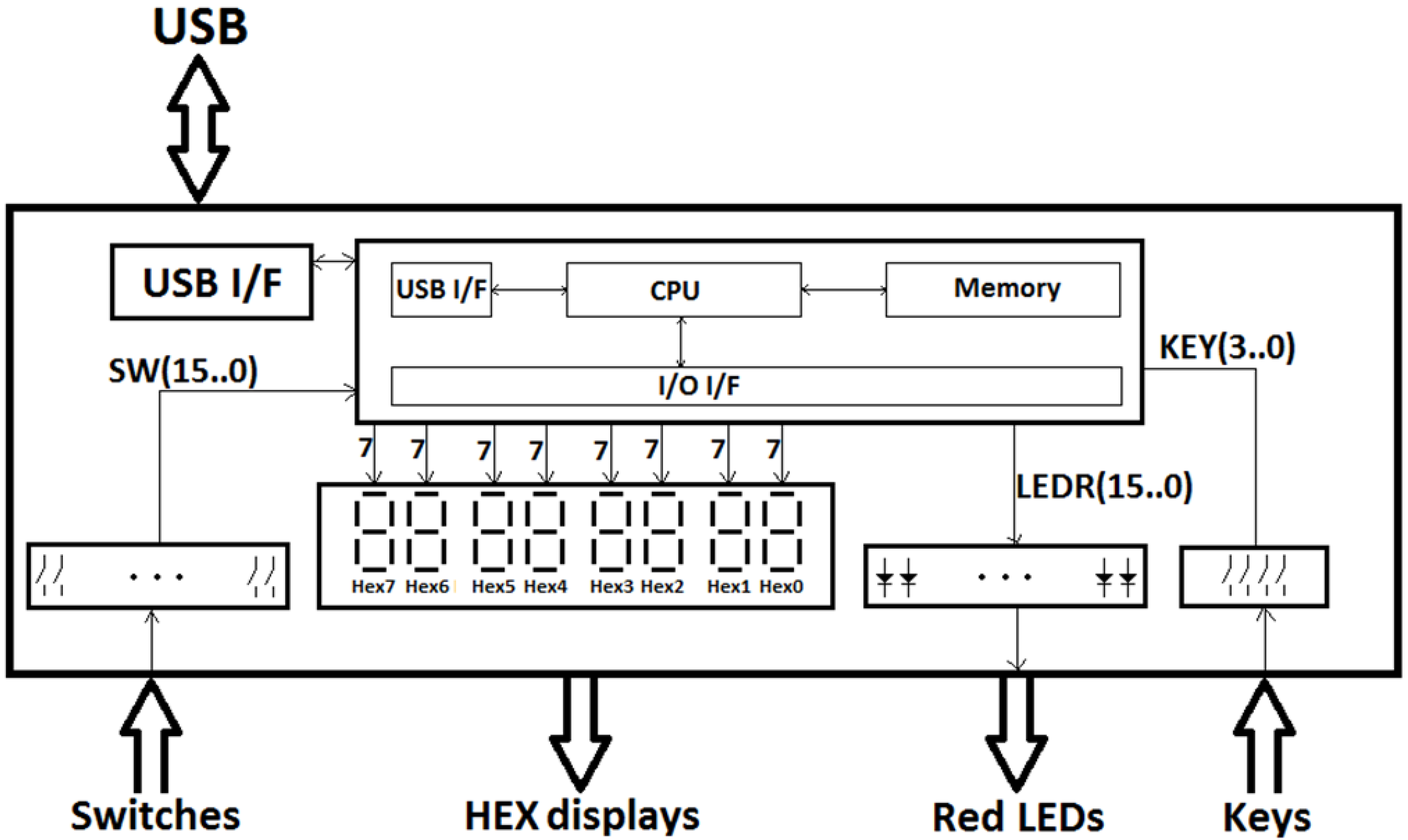
DE2 Board – Level 0



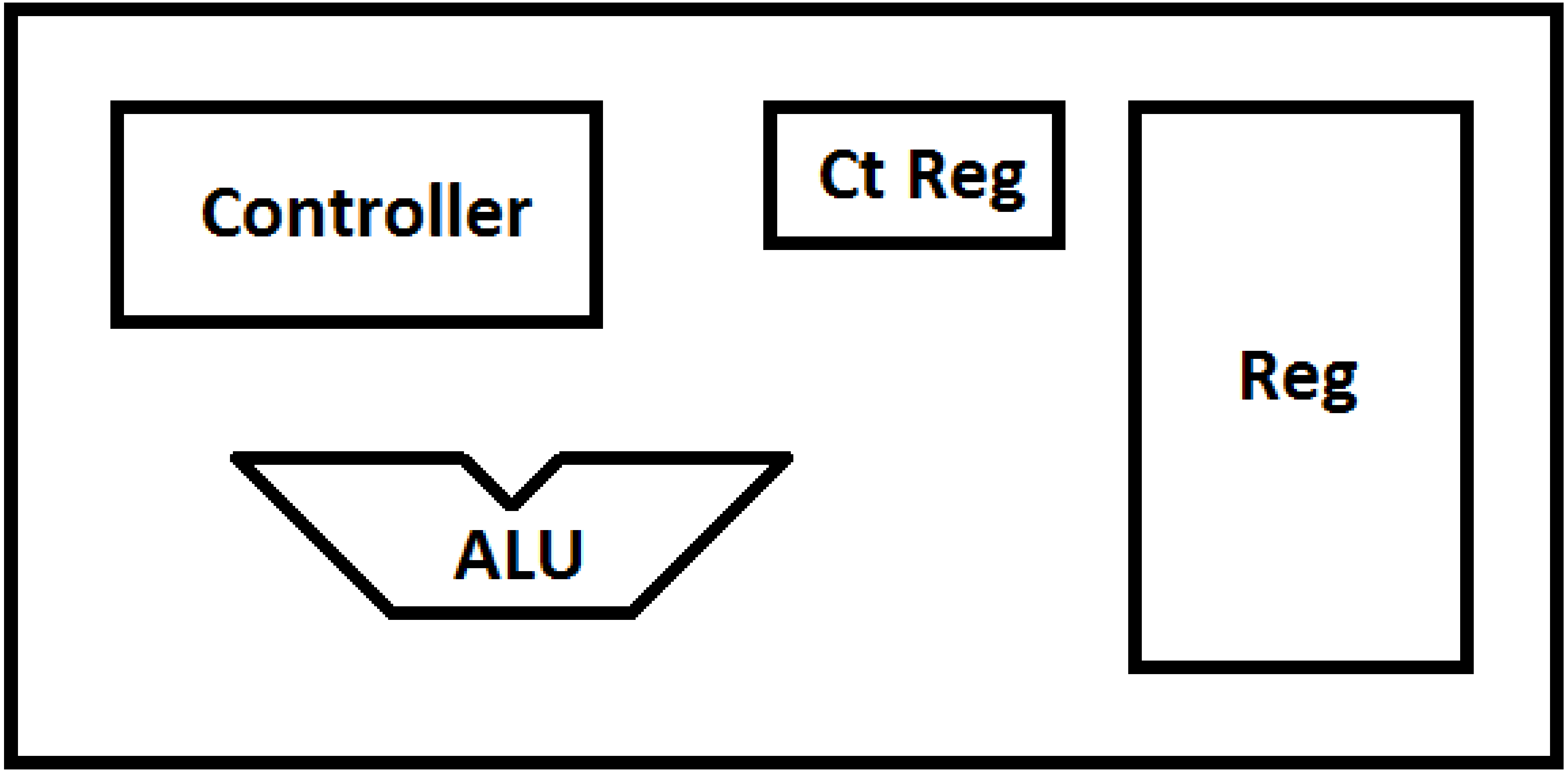
DE2 Board – Level 1



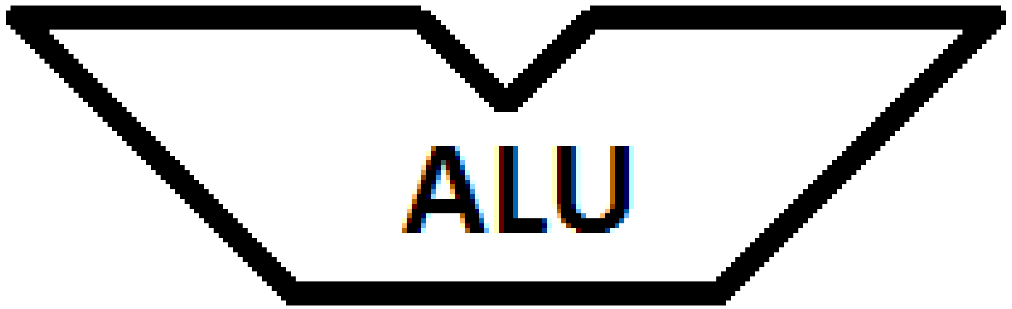
DE2 Board – Level 2

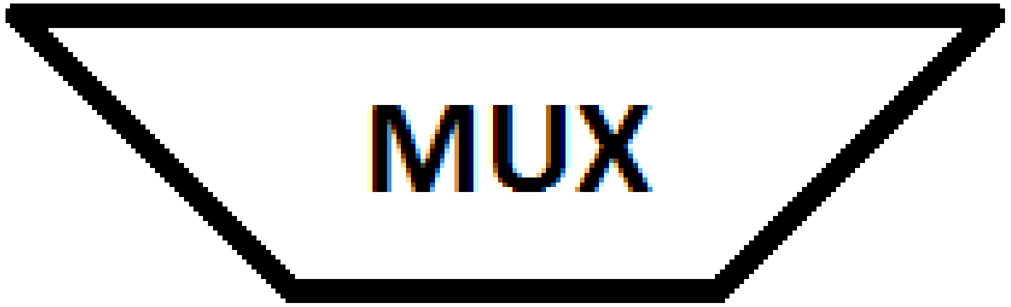


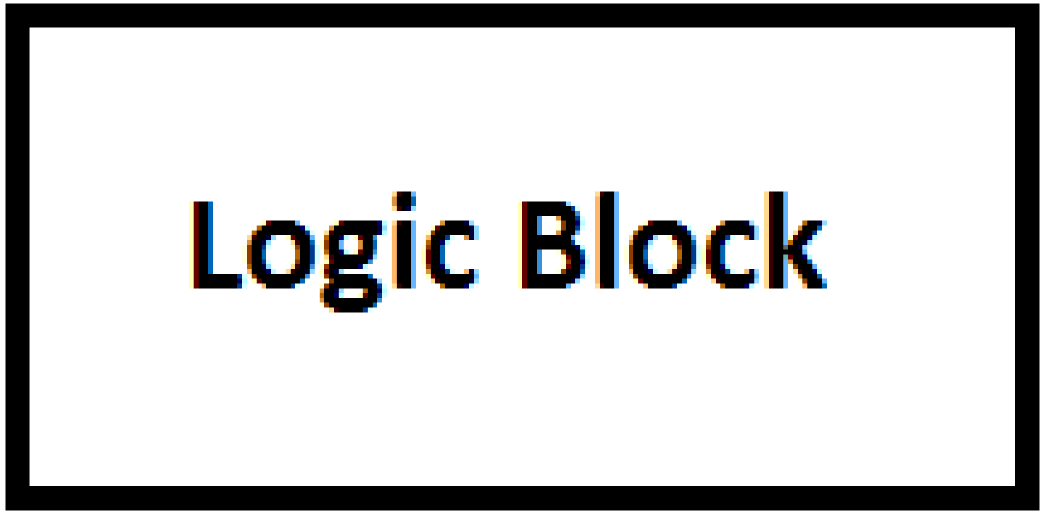
DE2 Board – Level 3 – CPU Only



Common Logic Block Symbols



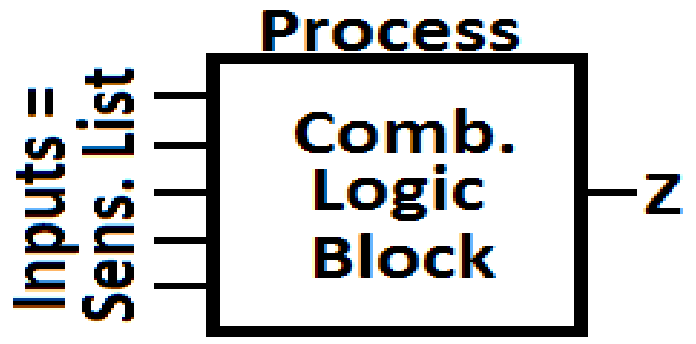


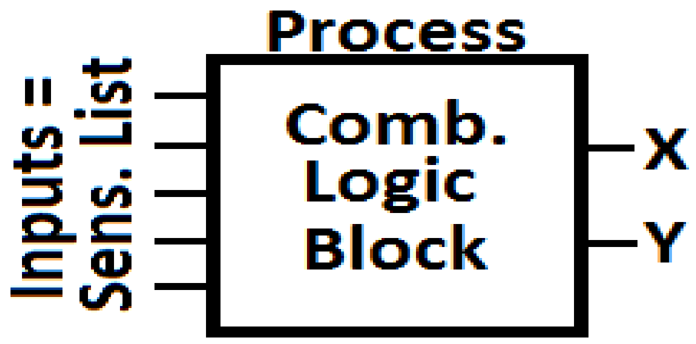


+ logic gate symbols

VHDL and Block Diagrams

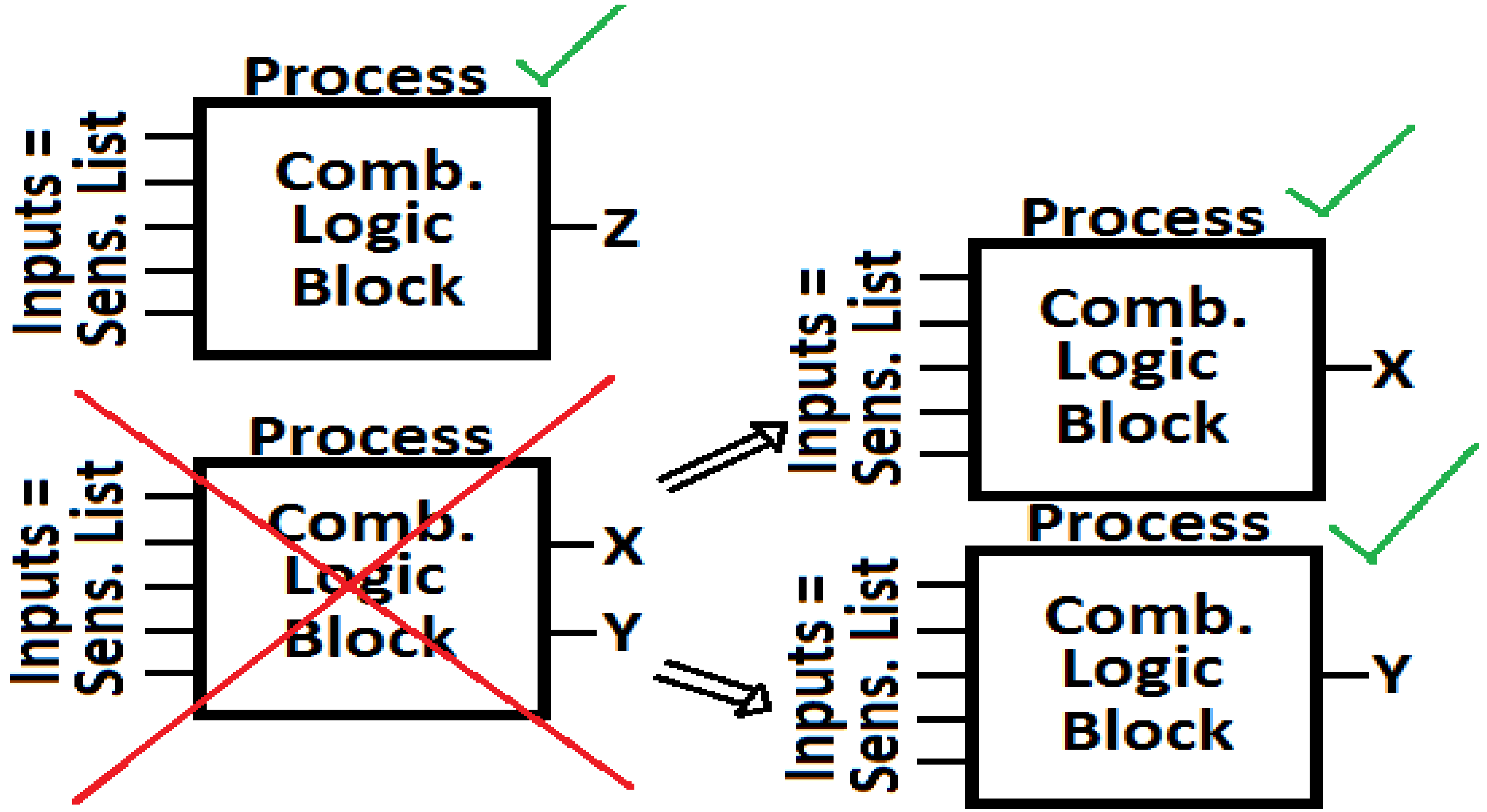
- A VHDL process should describe one (or a group) of blocks on your block diagram
- VHDL processes should have only 1 output (or 2 if the two are almost identical)





VHDL and Block Diagrams

- A VHDL process should describe one (or a group) of blocks on your block diagram
- VHDL processes should have only 1 output (or 2 if the two are almost identical)



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 ;
```