

Hardware-Software design issues on embedded system



Embedded Systems overview

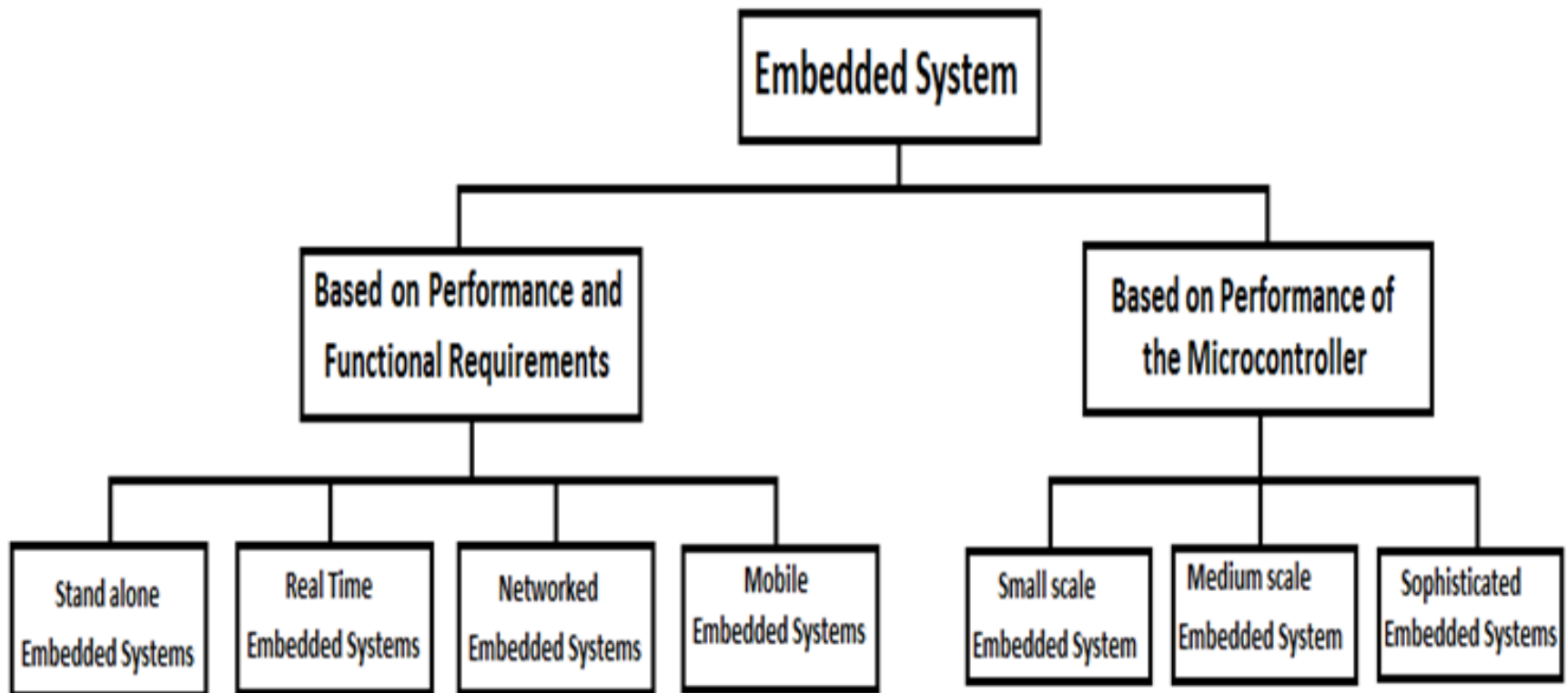
- Is a microcontroller or microprocessor based system which is designed to perform a specific task.
- Embedded means something that is attached to another thing
- For example, a fire alarm is an embedded system; it will sense only smoke
- Embedded systems may work independently or attached to a larger system to work on a few specific functions
- Three **main components of Embedded systems** are:
 - Hardware
 - Software
 - Firmware
 - has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies.
 - RTOS defines the way the system works.
 - It sets the rules during the execution of application program.
 - A small scale embedded system may not have RTOS

Characteristics of an Embedded System

- **Performs specific task**
- **Low Cost**
- **Time Specific:** It performs the tasks within a certain time frame.
- **Low Power:** Embedded Systems don't require much power to operate.
- **High Efficiency:** The efficiency level of embedded systems is so high.
- **Minimal User interface:** These systems require less user interface and are easy to use.
- **Less Human intervention:** Embedded systems require no human intervention or very less human intervention.
- **Highly Stable:** Embedded systems do not change frequently mostly fixed maintaining stability.
- **High Reliability:** Embedded systems are reliable they perform tasks consistently well.

- **Use microprocessors or microcontrollers:** Embedded systems use microprocessors or microcontrollers to design and use limited memory.
- **Manufacturable:** The majority of embedded systems are compact and affordable to manufacture. They are based on the size and low complexity of the hardware.

Classification of Embedded System



Real-Time Embedded Systems

- Provides output in a particular/defined time interval.
- Provide quick response in critical situations which gives most priority to time based task performance and generation of output.
- That's why real time embedded systems are used in defense sector, medical and health care sector, and some other industrial applications where output in the right time is given more importance.
- Is divided into two types i.e.

– Soft Real Time Embedded Systems

- In these types of embedded systems time/deadline is not so strictly followed.
- If deadline of the task is passed (means the system didn't give result in the defined time) still result or output is accepted.

– **Hard Real-Time Embedded Systems**

- In these types of embedded systems time/deadline of task is strictly followed.
- Task must be completed in between time frame (defined time interval) otherwise result/output may not be accepted.

- **Examples :**

- Traffic control system
- Military usage in defense sector
- Medical usage in health sector

Stand Alone Embedded Systems

- Stand Alone Embedded Systems are independent systems which can work by themselves they don't depend on a host system.
- It takes input in digital or analog form and provides the output.

- **Examples :**
 - MP3 players
 - Microwave ovens
 - calculator

Networked Embedded Systems

- Are connected to a network which may be wired or wireless to provide output to the attached device.
- They communicate with embedded web server through network.
- **Examples :**
 - Home security systems
 - ATM machine
 - Card swipe machine

Mobile Embedded Systems

- Are small and easy to use and requires less resources.
- They are the most preferred embedded systems.

- In portability point of view mobile embedded systems are also best.
- **Examples :**
 - MP3 player
 - Mobile phones
 - Digital Camera



Based on Performance and micro-controller

It is divided into 3 types as follows

Small Scale Embedded Systems

- Are designed using an 8-bit or 16-bit micro-controller.
- They can be powered by a battery.
- The processor uses very less/limited resources of memory and processing speed.
- Does not act as an independent system they act as any component of computer system but they did not compute and dedicated for a specific task.

Medium Scale Embedded Systems

- Are designed using an 16-bit or 32-bit micro-controller.
- Are faster than that of small Scale Embedded Systems.

•

- Integration of hardware and software is complex in these systems.
- Java, C, C++ are the programming languages are used to develop medium scale embedded systems.
- Different type of software tools like compiler, debugger, simulator etc are used to develop these type of systems.

Sophisticated or Complex Embedded Systems

- Are designed using multiple 32-bit or 64-bit micro-controller.
- These systems are developed to perform large scale complex functions.
- These systems have high hardware and software complexities.
- We use both hardware and software components to design final systems or hardware products.

Custom Single-Purpose Processor Design

Processor

- Is a digital circuit that performs computational tasks.
- The minimum requirement to be a processor is the presence of controller and data path.
- The different processor technologies are as follows:
 - General Purpose Processor (GPP)
 - It is a programmable circuit that can perform varieties of tasks.
 - It consists of program memory and general data path.
 - The data path has large register array and one or more general purpose ALU.
 - Single Purpose Processor (SPP)
 - It is a digital circuit designed to perform exactly one program.

- Digital Camera is a SPP. It only consists of data memory but not program memory.
- Application Specific Processor (ASP)
 - It is a programmable circuit that is optimized for a particular class of applications.
 - It consists of program memory , data memory , custom designed ALU and optimized datapath.

Custom Single Purpose Processor

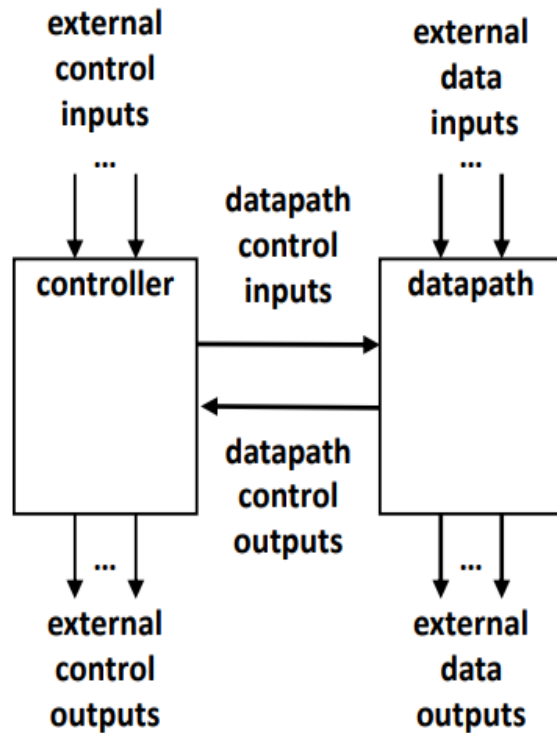
- Faster performance
- Size is smaller
- Lower power management
- High NRE cost
- Longer Time to market
- Less Flexible

Designing custom single purpose processor

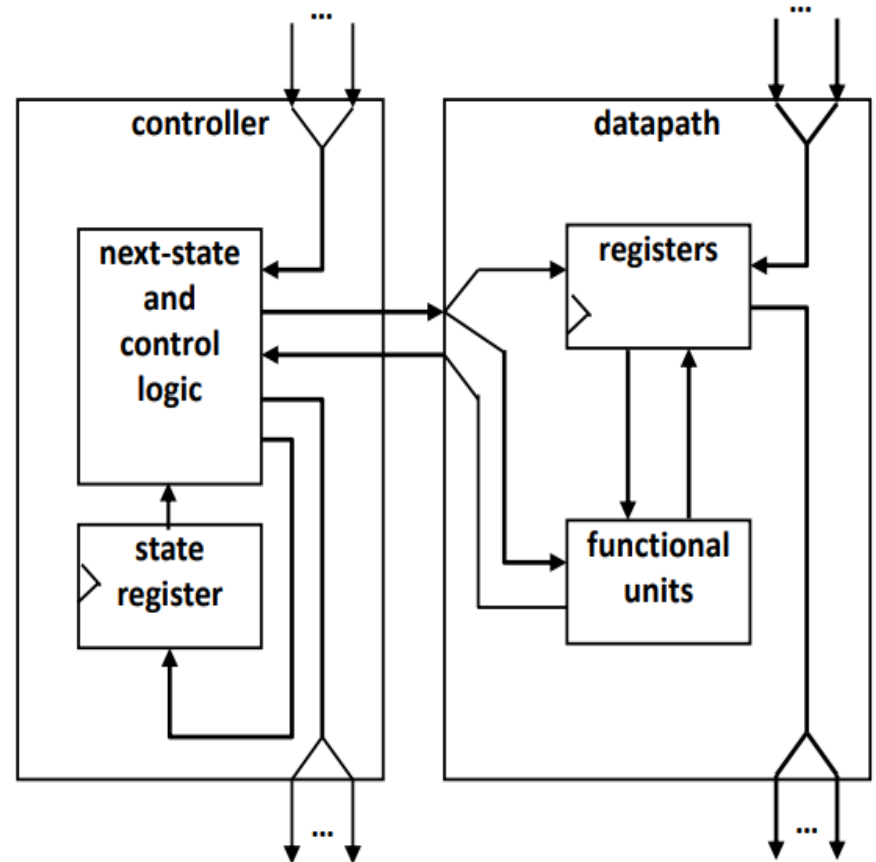
- Develop an algorithm or function that computes the desired output.
- Convert algorithm into a complex state diagram or FSM (finite state machine with data)
- Divide functionality with datapath part and controller part.
- Datapath consists of interconnection of combinational and sequential components. Controller consists of pure FSM
- Complete controller design by implementing FSM with combinational logic.

PANA ACADEMY

Single purpose processor design



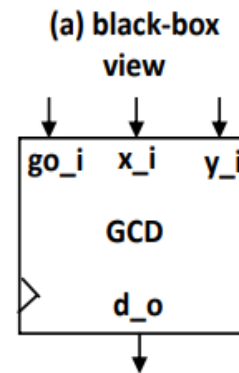
controller and datapath



a view inside the controller and datapath

Example: Greatest Common Divisor

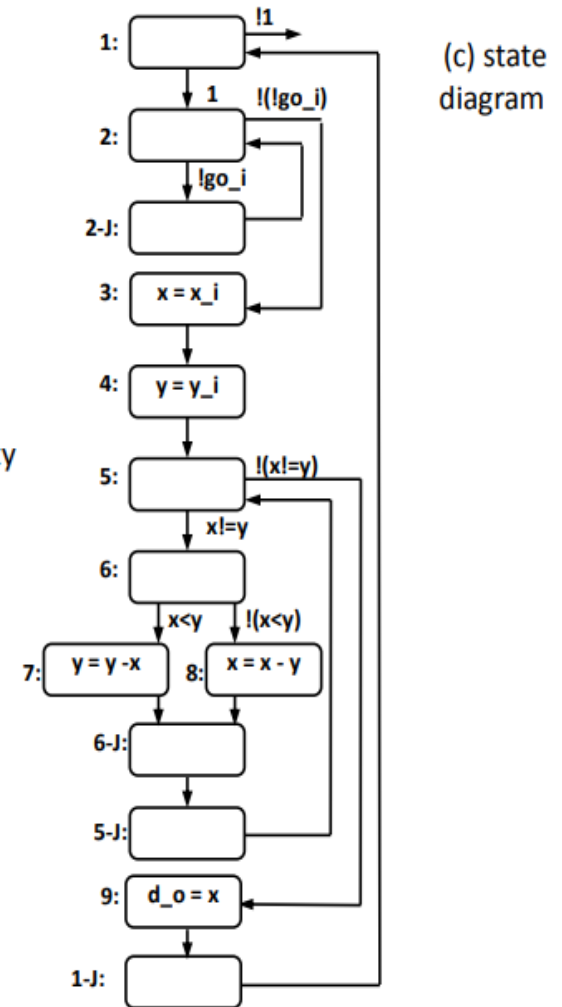
- **First create algorithm**
- **Convert algorithm to “complex” state machine**
 - Known as FSMD: finite-state machine with datapath
 - Can use templates to perform such conversion



(b) desired functionality

```

0: int x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while (x != y) {
6:     if (x < y)
7:       y = y - x;
8:     else
7:       x = x - y;
9:   }
9:   d_o = x;
}
    
```



Optimizing Custom Single Purpose Processor

Optimization

- Optimization is the technique of improving the design metrics so as to get the best possible values of various design metrics.
- The optimization opportunities in custom SPP are as follows:-

Optimization of original program:

- The algorithms are analyzed in terms of time complexity and space complexity , and hence we try to develop more efficient alternative algorithms.
- It involves decreasing of number of computations and size of variables if possible.
- Also involves time and space complexity
- Operations used
 - Multiplication and division very expensive

Optimizing FSM

Areas of possible improvements

- merge states
 - states with constants on transitions can be eliminated, transition taken is already known
 - states with independent operations can be merged
- separate states
 - states which require complex operations ($a*b*c*d$) can be broken into smaller states to reduce hardware size
- scheduling.

Optimizing the datapath

- Sharing of functional units
- one-to-one mapping, as done previously, is not necessary
- if same operation occurs in different states, they can share a single functional unit

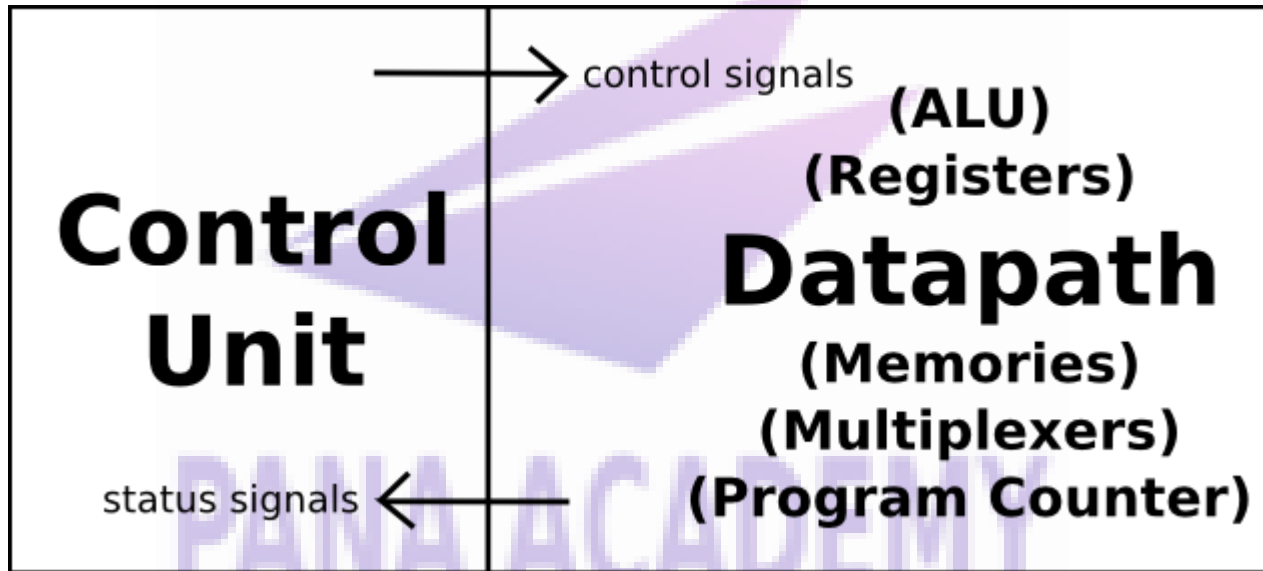
- Multi-functional units
- ALUs support a variety of operations, it can be shared among operations occurring in different states

Optimizing FSM

- State encoding
 - task of assigning a unique bit pattern to each state in an FSM
 - size of state register and combinational logic vary
 - can be treated as an ordering problem
- State minimization
 - task of merging equivalent states into a single state
 - state equivalent if for all possible input combinations the two states generate the same outputs and transitions to the next same state

Basic Architecture of Software Design and Operation

- It consists of general datapath
- Control unit does not store algorithms
- Algorithm is programmed into memory



Operations

- **Fetch Instruction:** It gets the next instructions as indicated by location in memory pointed by PC and stores into IR.
- **Decode Instruction:** It determines the actual operations performed by the instructions at IR.
- **Fetch Operands:** It gets the operand data needed for instruction from memory to appropriate registers of datapath.
- **Execute:** The actual arithmetic or logical operations is performed by moving data through ALU.
- **State Results:** It writes the data from datapath register into memory .

PANA ACADEMY

Programmer's View

- Programmer does not need detailed understanding of architecture.
- They just need to understand which instructions can be executed.
- Generally there are two levels of instructions assembly level and structured languages.
- Instruction set is the legal set of instructions that can be processed by a processor.
- Addressing modes indicates how the data for any operation is referenced in the instruction.

Programmer's Consideration

- Program and data memory space
- Registers
- Input Output (I/O)
- Interrupts
- Operating System

Development Environment

- Development processor: the processor on which programs are written.
- Target Processor: The processor that will run the program.
- If Development and target processor are different , the code can be run by downloading to target processor or by simulation.
- Simulation can be done by using HDL and Instruction set simulator(ISS)

Pipelining

- Pipelining is the mechanism to increase instruction throughput of a microprocessor.
- It assumes the independent operations to be performed simultaneously.
- Superscalar microprocessor executes two or more scalar operations in parallel and requires two or more ALU.

- VLIW (Very Long Instruction Word) architecture is a static superscalar microprocessor that encodes several operations in single machine instructions.

Application Specific Instruction Set Processor

- Is the processor targeted for a particular domain.
- This architecture is domain specific but can be programmed .
- For example Microcontroller , Digital Signal Processor
- It is designed to exploit special characteristics in the target application in order to meet performance, cost and energy requirements
- Is considered as balance between two extremes: ASICs and general purpose processors
- Application specific integrated circuit (ASIC) is a custom integrated circuit designed and optimized to fit a specific purpose and product

Benefits of an ASIP solution

- Maintain a level of flexibility/programmability through an instruction set
- Overcome the problems of conventional RISC/DSP architectures
 - Fixed level of parallelism which may prove inefficient for real-time applications of high computational complexity
 - Prohibitively high energy consumption
 - Time-critical tasks that require the incorporation of dedicated hardware modules
- Shortening of debugging/verification time
- Key enabler is FPGA technology (rapid prototyping/reconfigurable platforms)

VHDL

V : VHSIC (Very High Speed Integrated Circuit), H : Hardware, D : Description, L : Language

- Stands for Very High-Speed Integration Circuit HDL (Hardware Description Language).
- It is an IEEE (Institute of Electrical and Electronics Engineers) standard hardware description language that is used to describe and simulate the behavior of complex digital circuits.
- The most popular examples of VHDL are **Odd Parity Generator, Pulse Generator, Priority Encoder, Behavioral Model** for 16 words, 8bit RAM, etc.

VHDL supports the following features:

- Design methodologies and their features.
- Sequential and concurrent activities.
- Design exchange, Standardization, Documentation, Readability
- Large-scale design
- A wide range of descriptive capability

VHDL is used for the following purposes:

- For Describing hardware
- As a modeling language
- For a simulation of hardware
- For early performance estimation of system architecture
- For the synthesis of hardware

Basic Elements of VHDL

- There are the following three basic elements of VHDL:

1. Entity

- The Entity is used to specify the input and output ports of the circuit.
- An Entity usually has one or more ports that can be inputs (in), outputs (out), input-outputs (inout), or buffer.
- An Entity may also include a set of generic values that are used to declare properties of the circuit.

Entity Declaration

It can be declared using the following syntax:

Simplified syntax

```
entity entity_name is
ort (
port_1_name : mode data_type;
ort_2_name : mode data_type;
.....
Port_n_name : mode data_type
);
end entity_name;
```

Example: entiy orgate is

```
port (
a : in std_logic; b : in std_logic;
c : out std_logic ); end orgate;
```

Using generic

- If an entity is generic, then it must be declared before the ports. Generic does not have a mode, so it can only pass information into the entity.

Syntax:

```
entity entity_name is
  generic (
    generic_1_name : data_type;
    generic_2_name : data_type;
    .....
    generic_n_name : data_type
  );
  port (
    port_1_name : mode data_type;
    port_2_name : mode data_type;
    .....
    Port_n_name : mode data_type
  );
end entity_name;
```

Example:

```
entity Logic_Gates is
  generic (Delay : Time := 10ns);
  port (
    Input1 : in std_logic;
    Input2 : in std_logic;
    Output : out std_logic
  );
end Logic_Gates;
```

Rules for writing Port name:

- - Port name consist of letters, digits, and underscores.
- - It always begins with a letter.
- - Port name is case insensitive.

• Modes of Port

in	Input port
out	Output port
inout	Bidirectional port
buffer	Buffered output port

2. Architecture

- Architecture is the actual description of the design, which is used to describe how the circuit operates. It can contain both concurrent and sequential statements.

Architecture Declaration

- An architecture can be declared using the following syntax:
- architecture architecture_name of entity_name is
- begin
- (concurrent statements)
- end architecture_name;

Example:

```
architecture synthesis of andgate is
begin
    c <= a AND b;
end synthesis;
```

3. Configuration

- A configuration defines how the design hierarchy is linked together. It is also used to associate architecture with an entity.

Configuration Declaration

```
configuration configuration_name of entity_name is
--configuration declarations
for architecture_name
    for instance_label : component_name
        use entity library_name.entity_name(architecture_name);
    end for;
--
end for;
end [configuration] [configuration_name];
```

Example:

configuration demo_config of even_detector_testbench is

for tb_archi

for uut : even_detector

use entity work.even_detector (sop_archi);

end for;

end for;

end demo_config;

Types of Modeling styles in VHDL

- There are 4 types of modeling styles in VHDL:

1. Data flow modeling (Design Equations)

- Data flow modeling can be described based on the Boolean expression.
- It shows how the data flows from input to output. It works on Concurrent execution.

2. Behavioral modeling (Explains Behaviour)

- Behavioral modeling is used to execute statements sequentially.
- It shows that how the system performs according to the current statement.
- Behavioral modeling may contain Process statements, Sequential statements, Signal assignment statements, and wait statements.

3. Structural modeling (Connection of sub modules)

- Structural modeling is used to specify the functionality and structure of the circuit.
- Structural modeling contain signal declarations, component instances, and port maps in component instance.

VHDL objects

- VHDL uses the following three types of objects:

1. Constants

- Constant is an object which can only hold a single value that cannot be changed during the whole code.
- **Example:** constant number_of_bytes integer:=8;

2. Variables

- A variable also holds a single value of a given type.
- The value of the variable may be changed during the simulation by using variable assignment operator.
- Variables are used in the processes and subprograms.
- Variables are assigned by the assignment operator ":=".
- **Example:**
- variable index: integer :=0;

3. Signals

- Signals can be declared in architecture and used anywhere within the architecture.
- Signals are assigned by the assignment operator "<=".
- **Example:** Signal sig1: std_logic;
Sig1 <= '1'

Pipelining in VHDL

- Step-by-step guide to building a 3-stage pipelined processor for arithmetic operations

Instruction Fetch (IF) Stage:

- This stage fetches the instruction from memory and prepares it for decoding.

Instruction Decode (ID) Stage:

- This stage decodes the fetched instruction and extracts the operation to be executed.

Execution (EX) Stage:

- This stage performs the actual arithmetic operation based on the decoded instruction.

Data Hazards and Forwarding Logic

- In real-world scenarios, data hazards may occur when instructions depend on the results of previous instructions still in the pipeline. To handle these hazards, forwarding logic is required to forward data to the correct stages.

Overflow and data representation using VHDL

Now let's do this binary addition, what will we notice?

Example								
128	64	32	16	8	4	2	1	
								Carries
1	0	0	1	0	1	0	1	
1	1	0	1	1	0	0	0	+

We get

1	0	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---	---



Each 1 or 0 is called a **bit** short for **binary** digit

This is shortened to **b**

The calculation you just did resulted in a **9 bit** number

- Computers are designed to deal with a set of number of bits at once
- For example 8, 16, 32 or 64 bits
- Due to which the bit of left would be lost
- This is called overflow

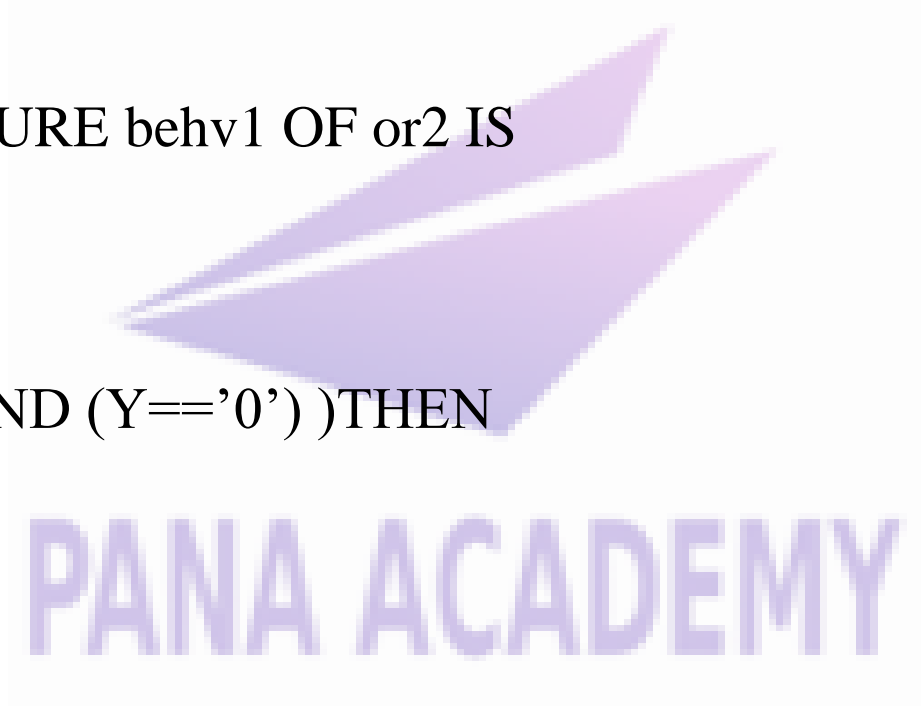
Design of combinational and sequential logic using VHDL

NOT Gate

```
library ieee;
use ieee.std_logic_1164.all;
ENTITY not 1 IS
    PORT(x:IN STD_LOGIC; F:OUT STD_LOGIC; );
END not1;
ARCHITECTURE behv1 of not1 IS
BEGIN
    PROCESS(X)
    BEGIN
        IF(X=='1') THEN F<='0';
        ELSE F<='1';
        ENDIF;
    ENDPROCESS;
END behv1;
```

OR Gate

```
Library ieee;  
use ieee.std_logic_1164.all;  
ENTITY or2 IS  
    PORT (x,y:IN STD_LOGIC; F:OUT STD_LOGIC; );  
END or2;  
ARCHITECTURE behv1 OF or2 IS  
BEGIN  
    process(x,y)  
    BEGIN  
        IF ((X='0') AND (Y=='0')) THEN  
            F<='0';  
        ELSE  
            F<='1';  
        ENDIF;  
    END process;  
END behv1;
```

The logo for PANA ACADEMY is located in the background. It features a stylized purple graphic consisting of two overlapping, elongated shapes that resemble a wing or a stylized letter 'P'. Below this graphic, the words "PANA ACADEMY" are written in a bold, purple, sans-serif font.

XOR Gate

Library ieee;

Use iee.std_logic_1164.all

ENTITY xor2 is

PORT (x,y :IN STD_LOGIC; F: OUT STD_LOGIC);

END xor 2;

ARCHITECTURE behv1 OF xor2 IS

BEGIN

IF((x=='0') AND (Y=='0')) THEN F<='0' ;

ELSE IF ((X='1') AND (Y=='1')) THEN F<='0';

ELSE

F<='1';

END IF ;

END PROCESS;

END behv1;

4:1 MUX

library IEEE;

use IEEE.STD_LOGIC_1164.all;

entity mux_4to1 is

port(

A,B,C,D : in STD_LOGIC;

S0,S1: in STD_LOGIC;

Z: out STD_LOGIC

);

end mux_4to1;

architecture bhv of mux_4to1 is

begin

process (A,B,C,D,S0,S1) is

begin

if (S0 = '0' and S1 = '0') then Z <= A;

elsif (S0 = '1' and S1 = '0') then Z <= B;

elsif (S0 = '0' and S1 = '1') then Z <= C;

else Z <= D;

end if;

end process;

end bhv;

- Which of the following is not a characteristic of combinational circuits?
 - a) The output of combinational circuit depends on present input
 - b) There is no use of clock signal in combinational circuits
 - c) The output of combinational circuit depends on previous output
 - d) There is no storage element in combinational circuit
- For using a process to implement a combinational circuit, which signals should be in the sensitivity list?
 - a) Inputs of the circuit
 - b) Outputs of the circuit
 - c) Both of the Inputs and Outputs
 - d) No signal should be in the sensitivity list
- A 4 to 16 decoder can be used as a code converter. What will be the inputs and outputs of the converter respectively?
 - a) Binary, Octal
 - b) Octal, Binary
 - c) Hexadecimal, Binary
 - d) Binary, Hexadecimal

Which of the following entity declares the ports of a 3 by 8 decoder?

- a)
- **ENTITY** decoder **IS PORT**(inp : **IN** STD_LOGIC_VECTOR(3 **DOWNTO** 0); Outp: **OUT** STD_LOGIC_VECTOR(8 **DOWNTO** 0));
END decoder;
- b)
- **ENTITY** decoder **IS PORT**(inp : **IN** STD_LOGIC_VECTOR(8 **DOWNTO** 0); Outp: **OUT** STD_LOGIC_VECTOR(3 **DOWNTO** 0));
END decoder;
- c)
- **ENTITY** decoder **IS PORT**(inp : **IN** STD_LOGIC_VECTOR(7 **DOWNTO** 0); Outp: **OUT** STD_LOGIC_VECTOR(2 **DOWNTO** 0));
END decoder;
- d)
- **ENTITY** decoder **IS PORT**(inp : **IN** STD_LOGIC_VECTOR(2 **DOWNTO** 0); Outp: **OUT** STD_LOGIC_VECTOR(7 **DOWNTO** 0));
END decoder;

- What does the architecture of an entity define?
 - a) External interface
 - b) Internal functionality
 - c) Ports of the entity
 - d) Specifications
- Which of the following is the correct architecture for a simple Nand gate?
 - a) ARCHITECTURE my_arch OF nand_gate IS BEGIN x <= a NAND b; END my_arch;
 - b) BEGIN ARCHITECTURE my_arch OF nand_gate IS x <= a NAND b; END behavioral;
 - c) BEGIN ARCHITECTURE behavioral OF nand_gate IS x <= a NAND b; END my_arch;
 - d) ARCHITECTURE nand OF nand_gate IS BEGIN x <= a NAND b; END nand;
- Which of the following can't be declared in the declaration part of the architecture?
 - a) Signals
 - b) Subprograms
 - c) Components
 - d) Libraries

- Which of the following task swapping method is a better choice in the embedded systems design?
 - a) time slice
 - b) RMS
 - c) cooperative multitasking
 - d) pre-emptive
- How an embedded system communicate with the outside world?
 - a) Memory
 - b) Output
 - c) Peripherals
 - d) Input
- Which of the following helps in reducing the energy consumption of the embedded system?
 - a) emulator
 - b) debugger
 - c) simulator
 - d) compilers
- Which design considers both the hardware and software during the embedded design?
 - Memory Design
 - Software/ hardware codesign
 - Platform-based design
 - Peripheral design

- How is the protection and security for an embedded system made?
 - a) Security chips
 - b) Memory disk security
 - c) IPR (intellectual property right)
 - d) OTP
- Which type of memory is suitable for low volume production of embedded systems?
 - a) Non-volatile
 - b) RAM
 - c) Volatile
 - d) ROM
- Which command takes the object file and searches library files to find the routine calls?
 - Emulator
 - Simulator
 - Linker
 - Debugger

- Which simulator/ debugger is capable of displaying output signal waveform resulting from stimuli applied to the inputs?
 - a) VHDL emulator
 - b) VHDL simulator
 - c) VHDL locator
 - d) VHDL debugger

The logo for PANA ACADEMY features a stylized, abstract graphic above the text. The graphic consists of several overlapping, translucent purple shapes that form a sense of motion or a wing-like structure. Below this graphic, the words "PANA ACADEMY" are written in a bold, sans-serif, purple font.

PANA ACADEMY