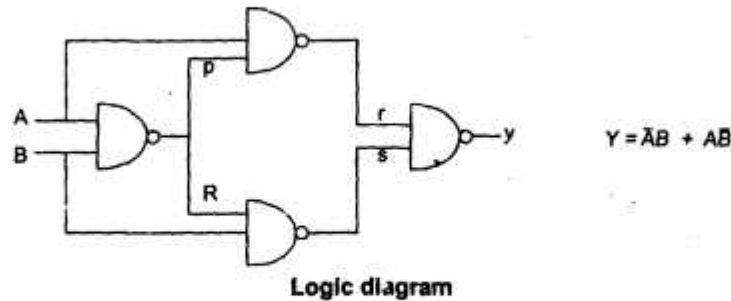


B.E.
Sixth Semester Examination, May-2008
Digital System Design (CSE-308-C)

Note : Attempt any five questions. All questions carry equal marks.

Q. 1. (a) Write a program in structural modelling style for EX-OR gate using four NAND gates.10

Ans.



Program :

```
Library ieee;
use ieee. std_logic__1164_all;
entity ex_or is
port(A, B : in bit; y : out bit);
end ex_or;
architecture ex_or of ex_or is
begin
signal p,q,r,s, : bit;
component nand
port (m, n : in bit : t : out_bit);
end component;
begin
a0 : nand port map (a, b, p);
a1 : nand port map (a, p, r);
a2 : nand port map (b, p, s);
a3 : nand port map (r,s,y);
end ex_or;
```

Q. 1. (b) Explain the concept of representation of data and its coding.

10

Ans. There are various methods to describe a digital circuit by using VHDL. We can describe i.e., represent the data by structure modelling, behaviour modelling and data flow modelling.

VHDL is a language used to describe a hardware. A hardware description of a digital system is called an entity. Entity specifies the external view and one or more internal views of the system..

Entity declaration describes how an entity is connected to outside world. It describes the external view of the entity. Entity declaration specifies the name of the entity

Internal details of an entity are specified by an architectural body by using any of the modelling styles.

In the data flow style of modelling, an entity is described in terms of dataflow by using concurrent signal assignment statements. In the behavioural style of modelling, the functionality of an entity is described without providing any details of how to implement it.

In structural style of modelling, an entity is described in terms of its components and interconnections.

Q. 2. (a) Explain all the data types in VHDL.

10

Ans. All the possible types that can exist in the language can be categorized into the following four major categories :

(a) Scalar Types : Values belonging to these types appear in sequential order.

(b) Composite Types : These are composed of elements of a single type (an array type) or elements of different types (a record type).

(c) Access Types : These provide access to objects of a given type (via pointers).

(d) File Types : These provide access to objects that contain a sequence of values of a given type.

Scalar Types : The values belonging to these types are ordered that is, relational operators can be used on these values.

There are four different kinds of scalar types. These types are :

(i) Enumeration

(ii) Integer

(iii) Physical

(iv) Floating point

(i) Enumeration Types : An enumeration type declaration defines a type that has a set of user-defined values consisting of identifiers and character literals.

Examples :

⇒ type MVL is ('V', '0', '1', 'Z');

The order in which values appear in an enumeration type declaration defines their ordering.

The values in an enumeration type are called enumeration literals.

⇒ type car_state is (stop, slow, medium, fast);

If the same literal is used in two different enumeration type declarations, the literal is said to be overloaded.

(ii) Integer Types : An integer type defines a type whose set of values fall within a specified integer range.

Examples are :

⇒ type INDEX is range 0 to 15;

Values belonging to an integer type are called integer literals.

Examples of integer literal are :

56349 6E2 0 98_71_28

(iii) **Floating Point Types** : A floating point type has a set of values in a given range of real numbers.

Example : type ITL_VOLTAGE is range -5.5 to -14;

Floating point literals differ from integer literals by the presence of the dot (.) character. Thus, 0 is an integer literal which 0.0 is a floating point literal.

Floating point literals can also be expressed in an exponential form.

Examples : $\Rightarrow 62.3E-2$

$5.0E+2$

(iv) **Physical Types** : A physical type contains values that represent the measurement of some physical quantity.

Example : \Rightarrow type CURRENT is range 0 to 1E9

units

nA;

$\mu A = 1000nA$;

$mA = 1000 \mu A$;

Amp = 1000mA;

end units.

Composite Types : There are 2 composite types :

(i) Array types

(ii) Record types

(i) **Array Types** : An object of an array type consists of elements that have the same type.

Example : \Rightarrow type ADDRESS_WORD is array (0 to 63) of BIT.

(ii) **Record Types** : An object of a record type is composed of elements of same or different types.

A record type is analogous to the record data type in Pascal and the struct declaration in C.

Example : \Rightarrow type PIN_type is range 0 to 10;

type MODULE is

record

SIZE : INTEGER range 20 to 200;

CRITICAL_DLY : TIME;

NO_INPUTS : PIN_TYPE;

NO_OUTPUTS : PIN_TYPE;

end record;

Access Types : Values belonging to access type are pointers to a dynamically allocated object of some other type.

They are similar to pointers in Pascal and C language.

Example : \Rightarrow type PTR is access MODULE;

\Rightarrow type FIFO is array (0 to 63, 0 to 7) of BIT;

type FIFO_PTR is access FIFO;

PTR is an access type whose values are addressed that point to objects of type MODULE.

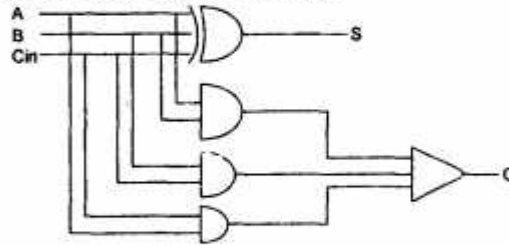
File Types : Objects of file types represent files in the host environment. They provide a mechanism by which a VHDL design communicates with the host environment.

⇒ type file-type-name is file of type-name;

A file can be opened, closed, read, written to, or tested by using special procedures and functions.

Q. 2. (b) Compare and contrast Data flow and structural models using the example of full adder.

Ans. Circuit diagram of full adder is shown in figure



Truth Table of Full Adder :

Inputs			Outputs	
A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Data Flow Model of Full Adder :

Library ieee;

use ieee; std_logic_1164.all;

Entity full_adder is

port (a, b, Cin : in bit; S, C : out bit);

end full_adder;

Architecture full_adder_dataflow of full_adder is

begin

S <= a X or b X or Cin;

C <= (a and b) or (a and cin) or (b and cin);

end full_adder_dataflow;

Structural Model of Full Adder

Library ieee;


```
Use ieee. std_logic_1164.all;
Entity full_adder is
    port (a, b, cin : in bit; S, C : out bit);
end full_adder;
Architecture full_adder_structure of full_adder is component X or 3
    port (in1, in2, in3 : in bit; out1 : out bit);
end component
Component and 2
    port (in4, in5 : in bit; out2 : out bit);
end component
Component or 3
    port (in6, in7, in8 : in bit; out3 : out bit);
end component;
Signal x, y, z : bit;
begin
    X0 : X or 3 portmap (a, b, cin, S);
    a0 : and 2 portmap (a, b, X);
    a1 : and 2 portmap (b, cin, y);
    a2 : and 2 portmap (a, cin, z);
    p0 : or 3 portmap (x, y, z, C);
end full_adder_structure;
```

Q. 3. (a) What are statements ? Explain sequential statements with the help of example. 10

Ans. A statement is used to write the model of an entity.

The declared components are instantiated in the statement part of an architecture body using component instantiation statements.

VHDL provides number of ways to control the execution of statements within a process.

A process statement contains sequential statements that describe the functionality of a portion of an entity in sequential terms.

Types : (a) Conditional statements

(b) Iterative statements

(a) Conditional Statements :

(i) if statement

(ii) case statement

(b) Iterative Statements :

(i) simple loop statement

(ii) for loop statement

(iii) while loop statement

DETAIL :

(i) If Statement : If statement is used to select collection of statements to execute if certain condition is true. The condition is Boolean i.e., on evaluation it returns value 'TRUE' or 'FALSE'

The if can be in following three forms :

(i) if end if

(ii) if else end if

(iii) if elseif else end if

(i) if condition then

statements

end if

(ii) if condition then

statements

else

statements

end if

(iii) if condition then

statements

elseif condition then

statements

elseif condition then

statements

end if;

(ii) Case Statement : Case statement selects the collection of statements based on the value of expression.

The expression must be of a discrete type or a one-dimensional array type.

Syntax :

case expression is

when value 1 \Rightarrow

sequential statement

when value 2 \Rightarrow

sequential statement

when value 3 \Rightarrow

sequential statement

end case;

(i) Simple Loop :

Syntax :

sequential statements

end loop;

(ii) For Loop : It includes the specification of how many times the body of the loop is to be executed.

Syntax :

```
for identifier in discrete range loop
sequential statements
end loop;
```

(iii) **While Loop** : While loop test a condition before each iteration, If the condition is true, iteration proceeds. If it is false, the loop statement is terminated.

Syntax :

```
while Boolean-expression loop
sequential statements
end loop;
```

Q. 3. (b) Explain what is overloading ? How it is removed ?

10

Ans. Overloading is classified in three different categories :

- (i) Subprogram overloading
- (ii) Operator overloading
- (iii) Literal overloading

(i) **Subprogram Overloading** : When two subprograms are existing with same name, then the subprogram name and corresponding. Subprograms are said to be overloaded.

Example : function number (Data : integer) return integer;
function number (Binary : bit) return bit;

Which a function call is made then the function for which call is made can be identified by the actual parameter list because they have different parameter types.

A call to overloaded subprogram is an error if it is not possible to identify the exact subprogram being called using any one of following methods :

- (i) Subprogram name
- (ii) Number of actual parameters.
- (iii) Types of actual parameters
- (iv) Return result type.

(ii) **Operator Overloading** : When an operator symbol is forced to behave differently then the operator is said to be overloaded.

For example :

“and” operator is predefined for operands of Bit or Boolean type and for one dimensional array of Bit or Boolean type.

If operands of “and” operator are of four-value type. In this case we have to write a function so that “and” perform logical and operation on operands of four-value type. Now this “and” operator is said to be overloaded.

Once we declared overloaded function, the operators are now called by any of the following methods :

- (i) Standard operator notation.
- (ii) Standard function call notation.

(iii) **Literal Overloading** : The values of an enumeration type are called enumeration literals. If the same literal is used in two different enumeration type declarations then the literal is said to be overloaded.

In such case, Whenever overloaded literal is used, the type of the literal is determined from its surrounding context.

Q. 4. (a) What is the need of component declaration ? In which style of modelling we use this declaration and why ? 10

Ans. A component is declared in declarative part of architecture body. Component declaration specifies the name of component, name of interfacing ports, their model (in, out, inout) and type of ports.

In **structural modelling** style we use component declaration.

In structural modelling, an entity is modelled as a set of interconnected components. Structural model does not tell about the functionality of an entity rather than it only tells how various components are interconnected to each other to form the entity. A structural description is easier to be synthesized.

The structural models have two parts, declarative part and statement part. In the architecture declaration part components are declared and in the statement part. The components declared in declarative part are instantiated by using component instantiation statements.

The syntax of component declaration is :

component component_name

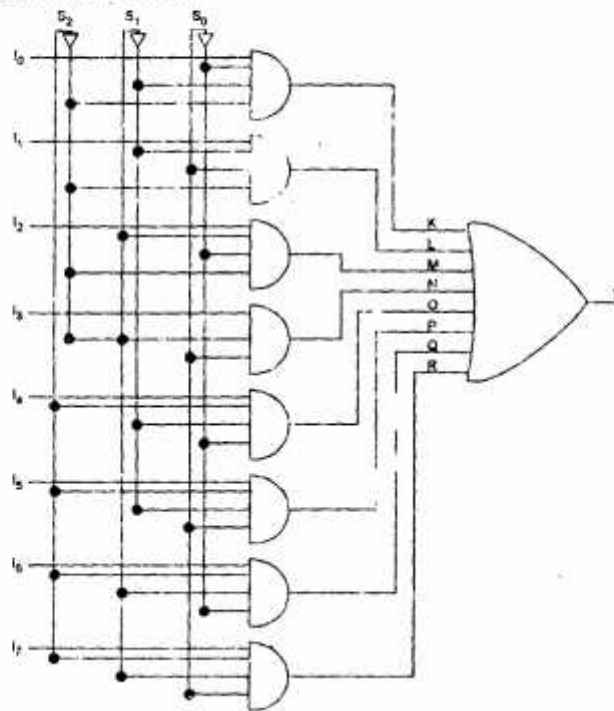
port(name, mode and type of interfacing ports);

end component.

Q. 4. (b) Model a 8 : 1 mux.

10

Ans. Circuit diagram of 8 : 1 mux.



Truth Tabel of 8 : 1 Mux :

S_2	S_1 (Select inputs)	S_0	Y (Output)
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

Program :

Library ieee;

use ieee.std_logic_1164-all;

entity mux is

port (i0, i1, i2, i3, i4, i5, i6, i7, S_0 , S_1 , S_2 : in bit; if : out bit);

end mux;

architecture mux_structure of mux is

component and 4

port (in1, in2, in3, in4 : in bit : out 1 : out bit);

end component;

component or 8

port (in4, in5, in6, in7, in8, in9, in10, in11 : in bit; out 2 : out bit);

end component;

Signal S_0 bar, S_1 bar, S_2 bar, k, l, m, n, o, p, q, r : bit;

begin

a0 : and 4 port map (i0, S_2 bar, S_1 bar, S_0 bar, k);

a1 : and 4 port map (i1, S_2 bar, S_1 , S_0 , l);

a2 : and 4 port map (i2, S_2 bar, S_1 , S_0 bar, m);

a3 : and 4 port map (i3, S_2 bar, S_1 , S_0 , n);

a4 : and 4 port map (i4, S_2 , S_1 bar, S_0 bar, o);

a5 : and 4 port map (i5, S_2 , S_1 bar, S_0 , p);

a6 : and 4 port map (i6, S_2 , S_1 , S_0 bar, q);

a7 : and 4 port map (i7, S_2 , S_1 , S_0 , r);

b0 : or 8 port map (k, l, m, n, o, p, q, r, y);

end mux_structure;

Q. 5. (a) Explain and model the working of ring counter.

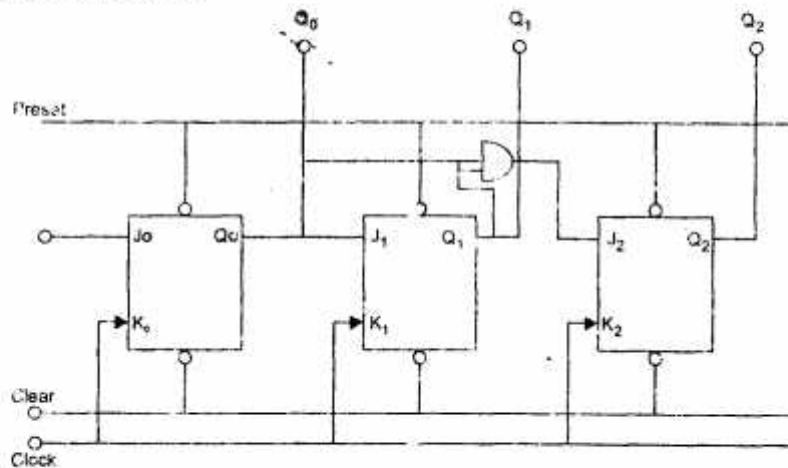
Ans. If the serial output of shift register is connected back to the serial input, then an injected pulse will keep on circulating. This circuit is referred to as ring counter. The pulses (clock pulses) are applied.

A circuit used for counting the pulses is known as a counter. The number of states in an N stage ring counter is N, whereas it is 2N in case of Molbius counter.

These counters are referred to as modulo N (or divided by N) and modulo 2N counters respectively.

The ring counter and the twisted ring counter don't make efficient use of flip-flops. A flip-flop has 2 states, therefore a group of N flip-flops will have 2^N states. This means it is possible to make a modulo 2^N counter using N flip-flops.

Behavioural Architecture :



```
library ieee;
use ieee.std_logic_1164.all;
entity upcount is
port (clock, resets, E : in std_logic; q : out std_logic_vector (2 down to 0));
end upcount;
architecture behaviour of upcount is
signal count : std_logic_vector (2 down to 0);
begin
process (clock, resets)
begin
if resets = '0' then
count <= '0000';
else if (clock event and data = '11') then
if E = '1' then
count <= count + 1;
else
count <= count;
end if;
end if;
end if;
end process;
q <= count;
end behaviour;
```

```

end if
end if
    end process;
    Q <= count
end behaviour

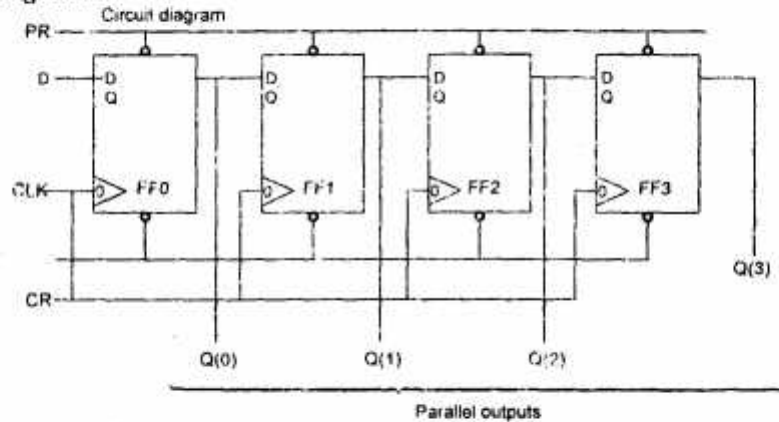
```

Q. 5. (b) Write the simulation for SIPO shift register.

10

Ans. 4 Bit Serial in Parallel Out Register

Circuit Diagram :



D = Serial Input

Behavioural Model (of 4bit SIPO Register)

Library ieee;

Use ieee.std_logic_1164.all;

entity sipo is

port (d, pr, cr, clk : in bit; q : buffer bit_vector (0 to 3));

end sipo;

architecture sipo-behavioural of sipo is

begin

process (pr, cr, clk)

begin

if (pr = '1' and cr = '1' and clk = '0') then

q(0) <= d;

q(1) <= q(0);

q(2) <= q(1);

q(3) <= q(2);

else if (pr = '1' and cr = '0') then

q(0) <= '0';

q(1) <= '0';

```

q(2) <= '0';
q(3) <= '0';
else if (pr = '0' and cr = '1') then
  q(0) <= '1';
  q(1) <= '1';
  q(2) <= '1';
  q(3) <= '1';
else if (pr = '0' and cr = '0') then
  q(0) <= '1';
  q(1) <= '1';
  q(2) <= '1';
  q(3) <= '1';
end if;
end process;
end sipo_behavioural;

```

Q. 6. (a) Explain the architecture of simple micro computer.

10

Ans. A micro computer is a general purpose system, i.e., a system in organized in such a way that it can perform a variety of computations.

A simple μ computer system composed of a processor, memory and input/output subsystems-and the connections between them.

The memory bus and the I/O bus are composed of the following signals :

An Address Bus which identifies the memory location or I/O port being accessed.

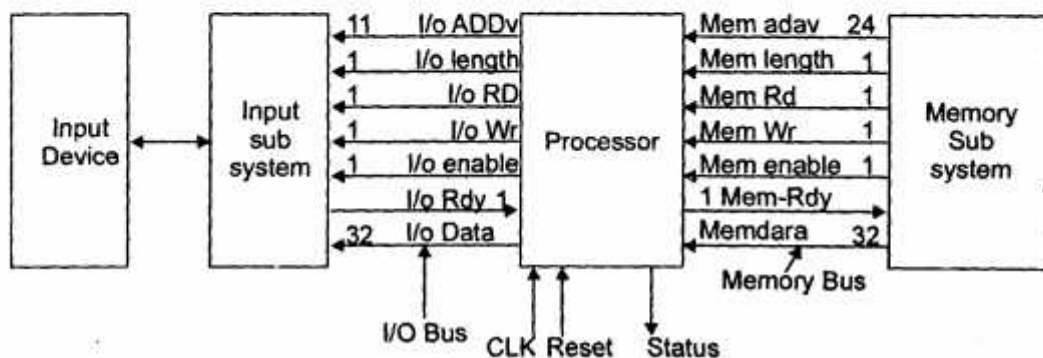
The memory address bus has 24 lines (to address upto 2^{24} location) whereas the I/O address bus has 11 lines (to address upto 2^{11} ports).

Control Signals, which indicate the type of access being performed (read/write), the length of the operand, and an enabling signal.

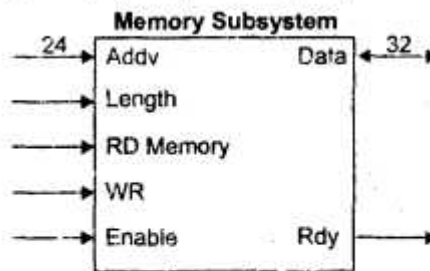
Status Signal, which indicates the state (busy, ready) of the memory or I/O subsystem.

A Data Bus : Which transfers the data to/from the processor and the memory or I/O subsystem.

These data buses have 32 lines for 32 bit data items.

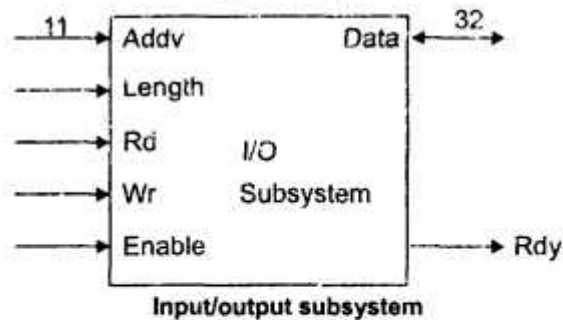


Structure of μ Computer (XMC) : (Example μ computer)



Input/Output Subsystem : The I/O subsystem contains the interfaces to devices that allow transferring data in/out to/from the computer

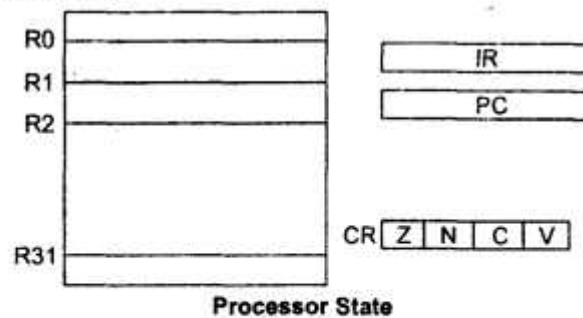
These interfaces are addressed as an array of 2048 ports.



Processor : Contains :

- (i) 32 general purpose registers
- (ii) A 24 bit program counter
- (iii) A 4 bits condition register
- (iv) A 32 bits instruction register

General Purpose Register :



Q. 6. (b) Implement a 8085 processor with the help of a VHDL.

10

Ans. The structure of processor is described in μ VHDL as follows :

library ieee;

```

use ieee.std_logic_1164.all;
architecture structural of processor is
    signal InStr : word T;
    signal ZE, NG, CY, OV : std_logic;
    signal AddrA, addrB, addrC : std_logic_vector (4 down to 0);
    signal ALUop : std_logic_vector (3 down to 0);
    signal WrC, WrPc, WrCR, WrIR : std_logic;
    signal Mem_ALU, PC_RA, IR_RB : std_logic;
    signal ALU_PC, ZE_SE, Sin Sout : std_logic;
begin
    P1 : entity Data_subsystem
        port map (Mem addr, Mem data, IO Addr, IO Data, InStr, ZE, NG, CY, OV, addrA, addrB, addrC,
            ALUOP, WrC, WrPC, WrCR, WrIR, Mem_ALU, PC_RA, IR_RB, ALU_PC, ZE_SE, Sin S out,
            CLK, Reset);
    P2 : entity ctrl_subsystem
        port map (Instr, ZE, NG, CY, OV, AddrA, addrB, addrC, ALUop, WrC, WrPC, WrCR, WrIR,
            Mem_ALU, PC_RA, IR_RB, ALU_PC, ZE_SE, SinSout, MemRd, MemWr, Mem length,
            MemEnable, MemRdy, IORd, IOWr, IO length, IO Enable, IO Rdy, Status, CLK, Reset);
end structural;

```

Q. 7. (a) Explain the need and working of PLD's.

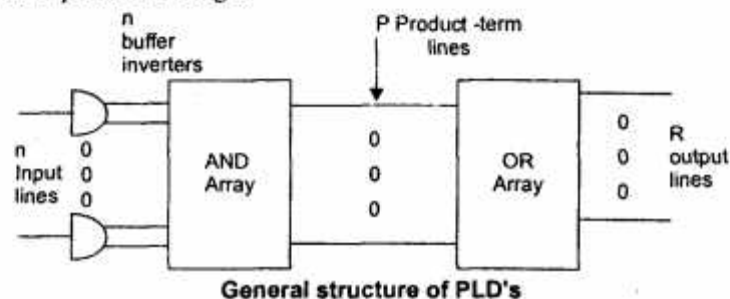
Ans. A programmable logic device or PLD is an electronic component used to build digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed. The PLDs allow designers more flexibilities because these can be programmed in seconds.

The design deficiencies and modifications can be reprogrammed and carried out in short time thereby reducing the possibility of huge cost over runs.

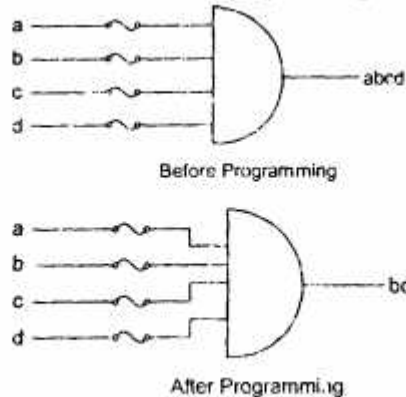
A PLD consists of arrays of AND and OR gates.

A system designer implement a logic design with a device programmer that blows fuses on the PLDs to control gate operation.

System designers can use development software that converts basic code into instructions a device programmer needs to implement a design.



Programming by blowing fuse, In PLDs, we are programming functions by connected fuse.



Q. 7. (b) Explain the architecture and working of PEEL.

Ans. A similar device called PEEL (Programmable Electrically Erasable Logic) was introduced by international CMOS technology (ICT) corporation.

ICT Inc offers the most flexible PLD solutions for lower pin count application, FICT's programmable electrically erasable logic, PEEL. products includes PEEL devices, PEEL array and PEEL development tools.

The PEEL product features include the most extensive PLD architecture offering in 20-44 pin packages. The technology is CMOS electrically erasable logic, the product line is a direct 'JEDEC' file compatible replacement and a super set for the popular PAL's, GAL's and EPLD's.

The product line offers flexible architectures with extended logic for new design or design enhancements.

The product pin-to-pin speeds are characterised to 5ns and faster, ICT Inc offers a broad variety of packaging options the product are available in DIP, PLCC, SDIC and TSSOP packages.

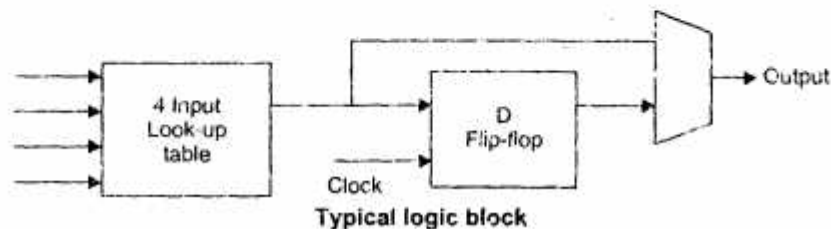
Q. 8. Write short notes on :

(i) FPGA, (ii) Various Macro Cells

Ans. (i) FPGA : A field-programmable gate array is a semiconductor device containing programmable logic components called "logic blocks" and programmable interconnects.

Logic blocks can be programmed to perform the function of basic logic gates such as AND and XOR, or more complex combinational functions such as decoders or simple mathematical functions.

In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

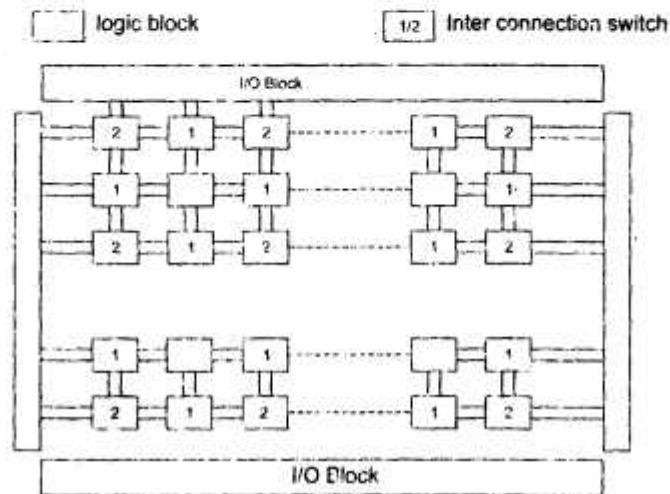


A hierarchy of programmable interconnects allows logic blocks to be interconnected as needed by the system designer, somewhat like a one-chip programmable breadboard.

Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA is manufactured, to implement any logical function. Hence the name field programmable.

Applications : Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defense system, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bio informatics, Computer hardware evaluation and a growing range of other areas.

FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture.



(b) Various Macro Cells : Developments in the design of programmable array logic have led to the introduction of configurable outputs enhancing the output capabilities of such devices.

The configurable (also called generic) device architecture is provided by equipping the device with enhanced special circuitry, known as output macro cells.

A macro cell has circuitry with fuses which can be configured for a variety of output options, giving flexibility to the device.

A configurable PAL can replace a large number of simpler PAL type devices with a "one size fits all" devices and allows designs to be implemented that are challenging or simply impossible for the simpler PAL devices to handle.

Example : Can be 22V10 configurable PAL. It has 22 inputs, 10 outputs, and 120 product terms.

Out of 22 inputs, 12 are dedicated inputs and 10 may be used for inputs as well as outputs.

The generic array logic (GAL) device is another type of configurable PAL. GAL devices are intended as pin-for-pin replacements for a wide variety of PAL devices. It is designed to be compatible, all the way to the fuse level, for any simple PAL which can be directly implemented in the GAL device.

Example : GAL 16V8 replaces most 20 pin PAL devices, while the 20V8 replaces most 24-pin PAL devices. Its macrocell can be configured in one of the three basic configurations. In this device, the OR gate is considered to be a part of the macro cell to obtain various types of I/O configurations found in the PAL devices that the GAL is designed to replace.