– Lecture - 3 –

14th Aug. 2019

Hardware Description Language (HDL)

- VHDL
- Verilog
- Basic HDL (BHDL) → in this course

Conventions :-

File extension - Filename.hdl

And.hdl
Xor.hdl
Nand.hdl

.hdl extension is important -

in most case the file name will be with caps as

And.hdl
↳ cap ↳ extension.

Chip structure :-

Header → specifies interface

Body → specifies the implementation.

Syntax Convention :-

- case sensitive
- Keywords are in upper case.

Identifier naming :- letter and digits, starting with letter.

Keyword | CHIP | Decoder { → Identifier

2

comments :- // comment to end of a line

/* comment until closing */

Chip Header (Interface) :-

```
CHIP Chipname {
    IN inputpin1, inputpin2, inputpin3;
    OUT outputpin1, outputpin2;

    // chip Body (Implementation)
}
```

Chip Body: (Implementation)

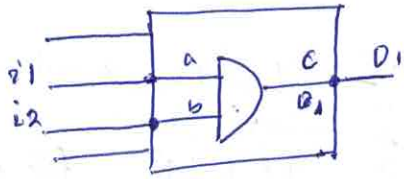
```
PARTS:
    internal chip part-;
    internal chipart;
```

- **Internal chip part :-**

~~Chip-Name~~ Chipname (connection, ... , connection)

Connection:

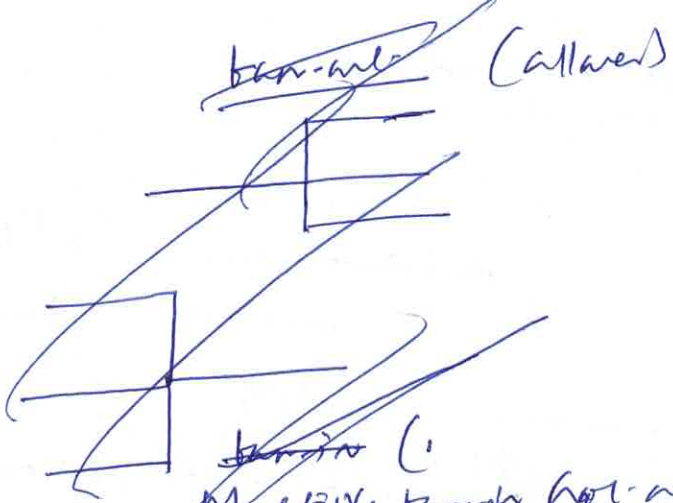
Part's pin name = chip's pin name.



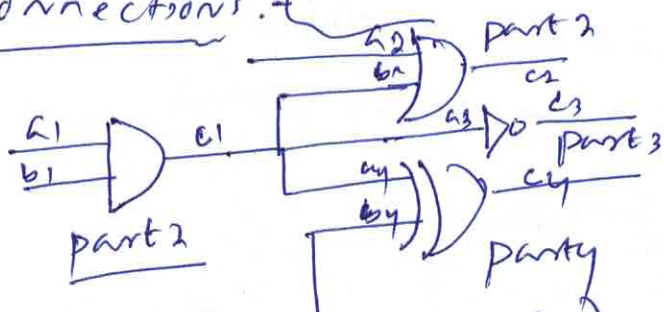
~~a = a~~

a = i1, b = i2

c = O1



Pins & Connections :-



internal pins :- (interconnects)

```
Part1 ( . , out=p);
part2 (          );
part3 (          );
```

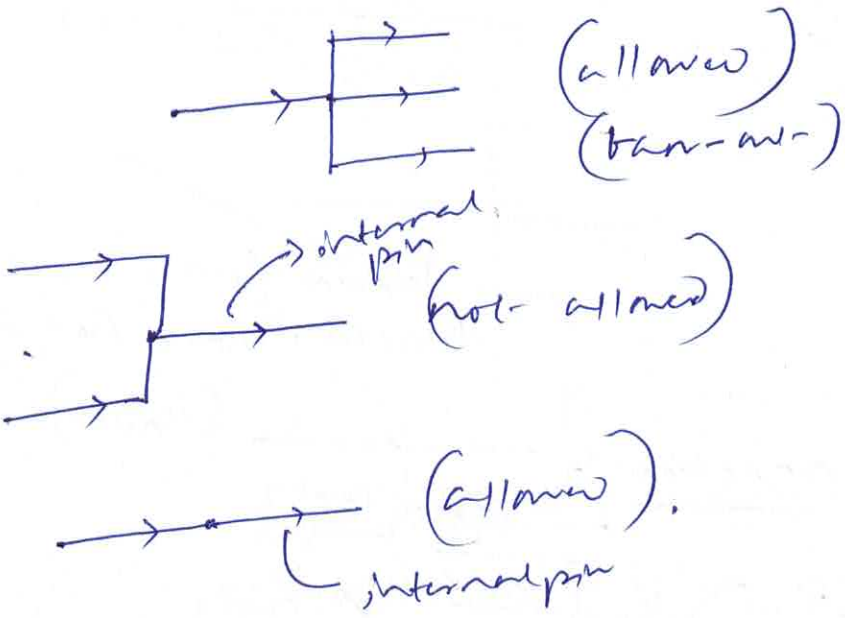
Input pins :- An input pin of a part may be fed by the following sources :-

1) input pin of chip
2) an internal pin (interconnect)
3) constant ~~True or False~~ True (1) or False (0)

output pins :- An output pin of a part could feed to

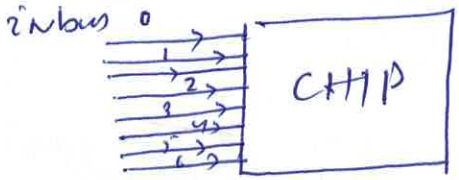
1) output pin of a chip
2) an internal pin (interconnect)

Restriction on connection :-



- Buses -

A bus is a multi bit pin of type input-, output- or internal.

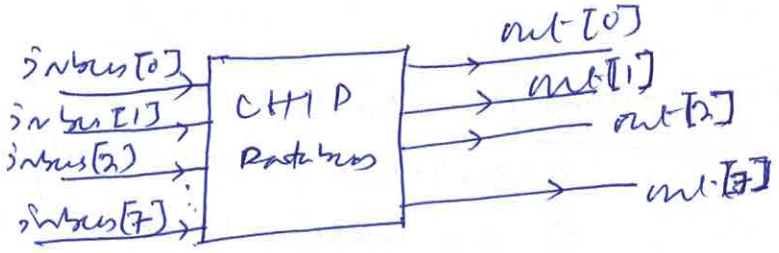


Declaration :-

~~CHIP Mybus {}~~

```
CHIP Databus {
    IN inbus[8];
    OUT outbus[8];
    // body of data bus.
}
```

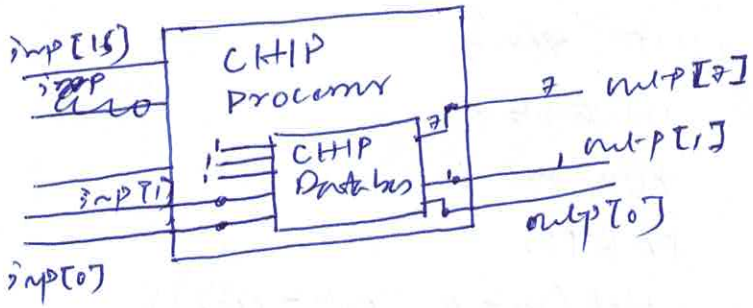
How to use :-



Chip instantiation (in presence of bus)

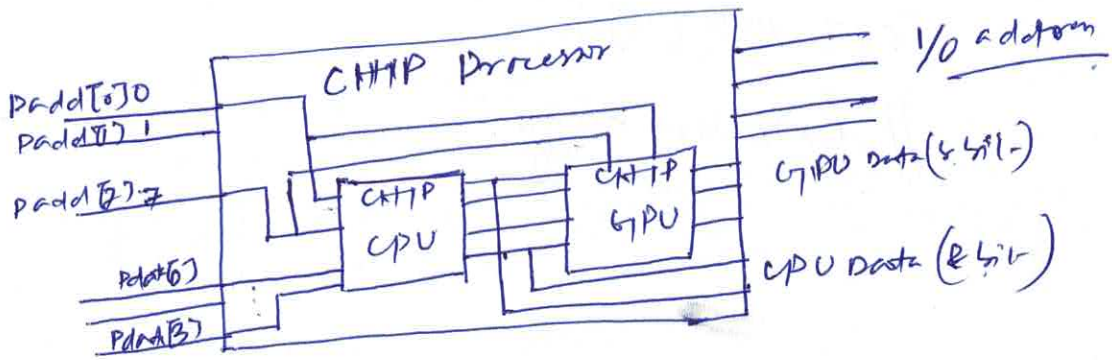
```
CHIP Databus {
    IN in[8];
    OUT out[8];
    // Data bus body.
}
```

```
CHIP Processor {
    IN inp[16];  // chip's pin
    OUT outp[16]; // chip's pin
    // start of body for processor
    PARTS:
    // some other gates.
    Databus (in[0] = inp[0], in[1] = inp[1],
             in[2..7] = true,
             out[0] = outp[0]
             out[1..7] = outp[1..7])
}
```



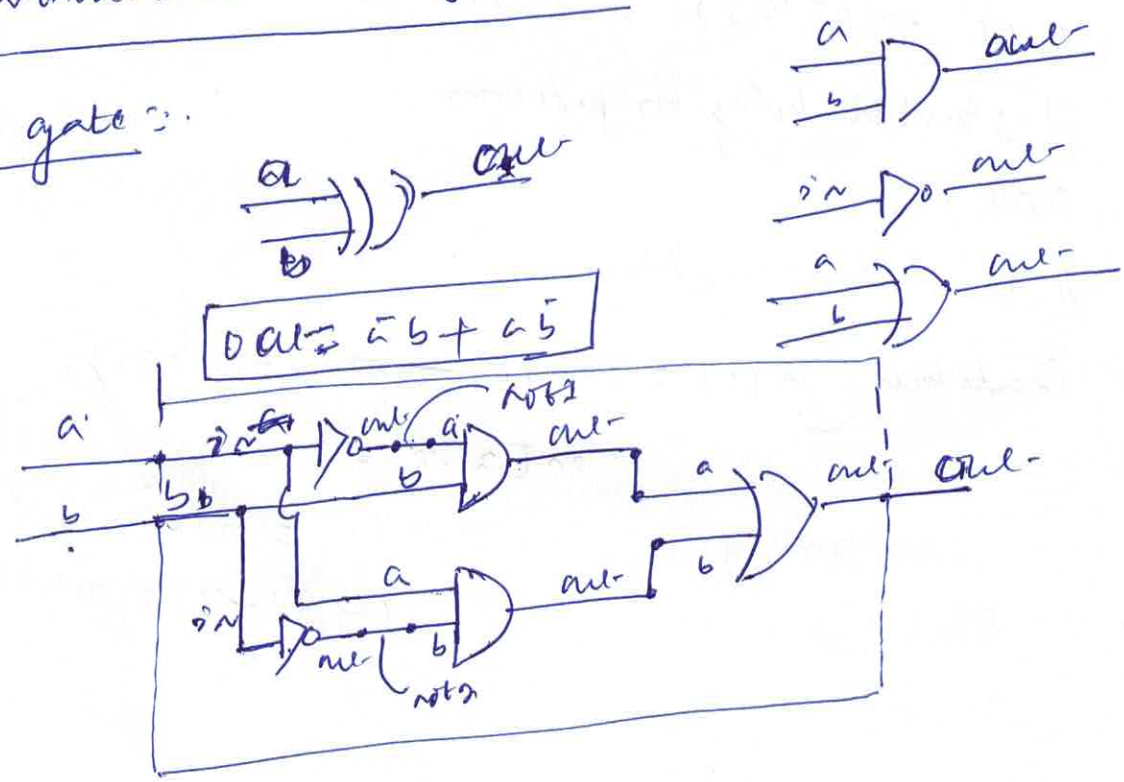
Home Exercise

Describe the following connections using HDL.



Combinational Building blocks :-

XOR gate :-



HDL Code

```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not (in = a, out = Not1);
  Not (in = b, out = Not2);
```

And (a = not1, b = b, out = and1);
And (a = a, b = not2, out = and2);
Or (a = and1, b = and2, out = out);

}

Simulation of XOR gates :-

Simulation :- To verify the correctness of a given chip
Language - Test Scripting Language (TSL)

Xor.tst

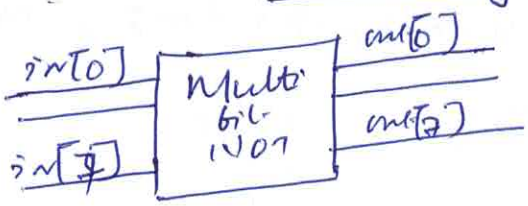
→ API the i/p/o/p pins to be simulated and directed to console

load Xor.hdl, output-list a, b, out;
set a 0, b 0, ~~eval~~
eval, output;
set a 0, b 1,
eval, output;
set a 1, b 0,
eval, output;
set a 1, b 1,
eval output;

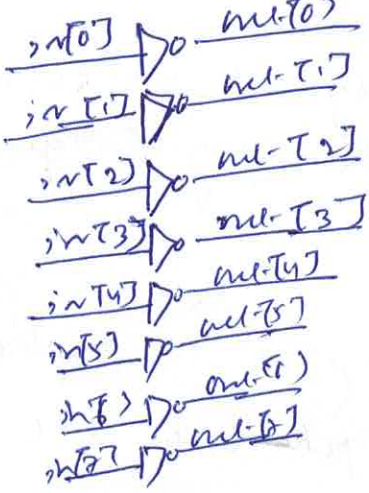
Multi bit gates :-

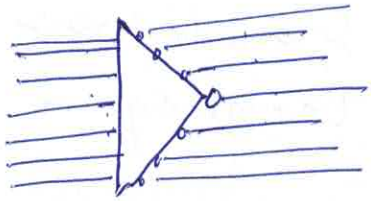
Multi bit NOT

Block diagram



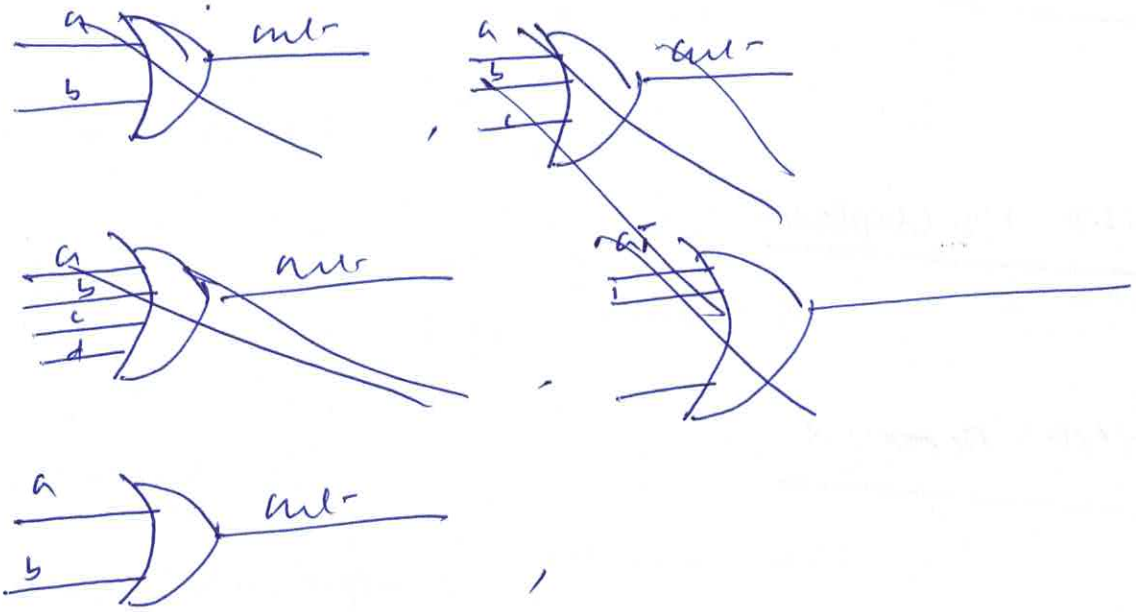
Functionality

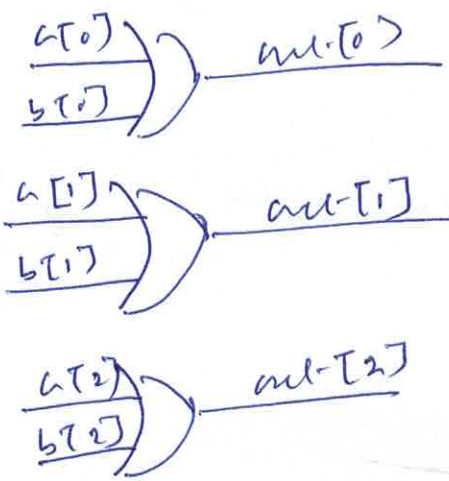


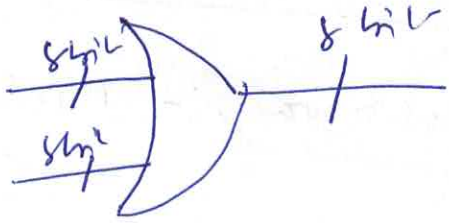


```
CHIP MBNot {

    IN in[8];
    OUT out[8];

    PARTS:
    Not (in = in[0], out = out[0]);
    Not (in = in[1], out = out[1]);
    =
    =
    Not (in = in[7], out = out[7]);

}
```

Multi bit OR :-







HDL Code :-

```
CHIP OR16 {
    ||||
    ||||
    ||||
    ||||
}
```

Multi bit AND :-

Multibit XOR :-

Multibit Multiplexer :-

Multibit Demux :-

Multiway gates -

Multiway :- Multiple input with only one output.

Multiway OR

Multiway AND

Multiway XOR

Multiway MUX

Multiway DeMUX :-