AHDL
Training Class

Danny Mok

Altera HK FAE

(dmok@altera.com)

What is AHDL

- ***A***ltera ***H***ardware ***D***escription ***L***anguage
  - develop by Altera
  - integrate into the Altera software Max+Plus II
  - description the hardware in language instead of graphic
    - easy to modify
    - easy to maintane
  - very good for
    - complex combinational logic
      - BCD to 7 Segment converter
      - address decoding
    - state machine
    - more than you want……..

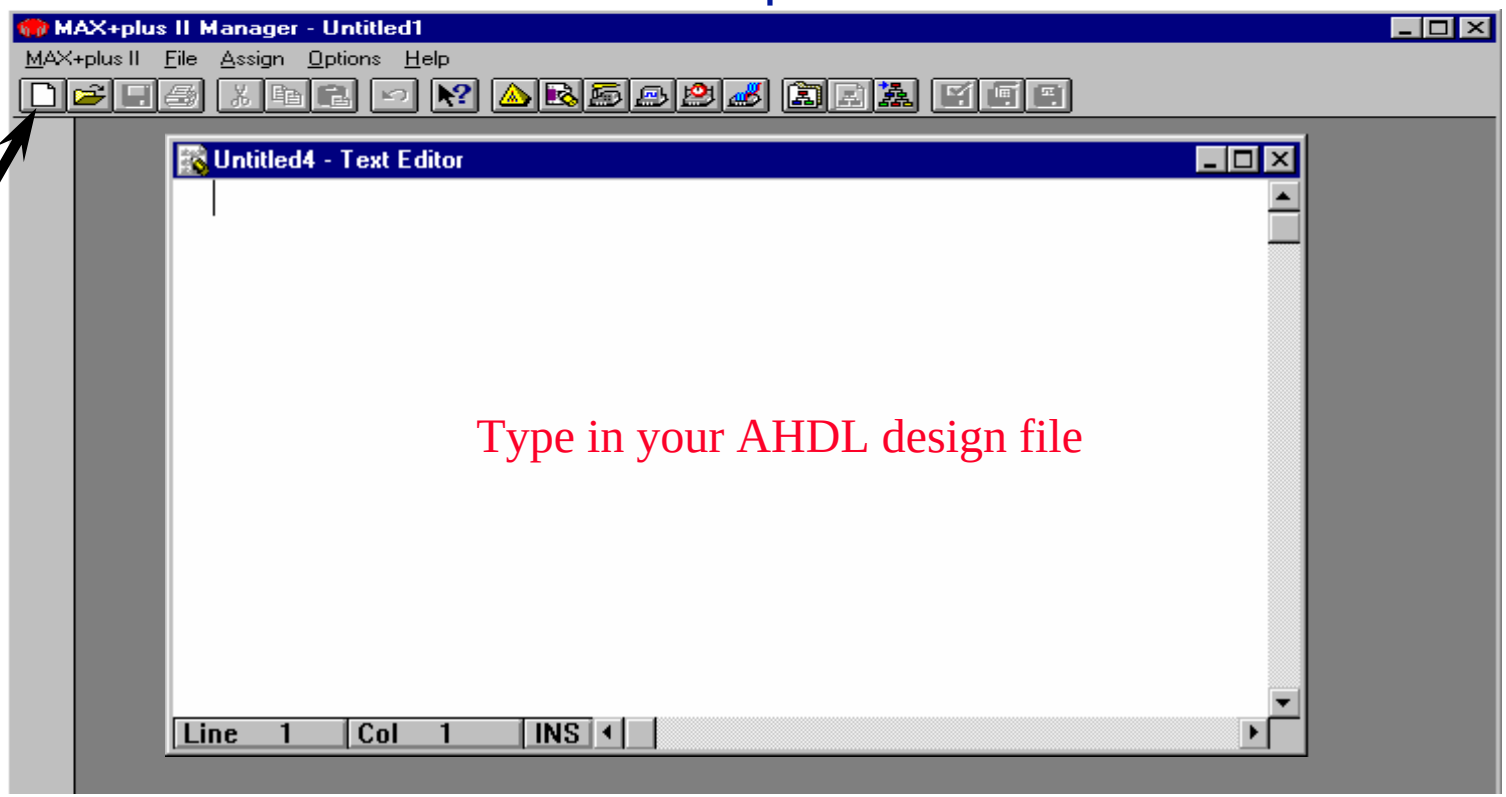
continue...

- As easy as Graphic Entry
- As powerful as HDL (Hardware Description Language)
  - VHDL, Verilog HDL etc.

How to use the ADHL

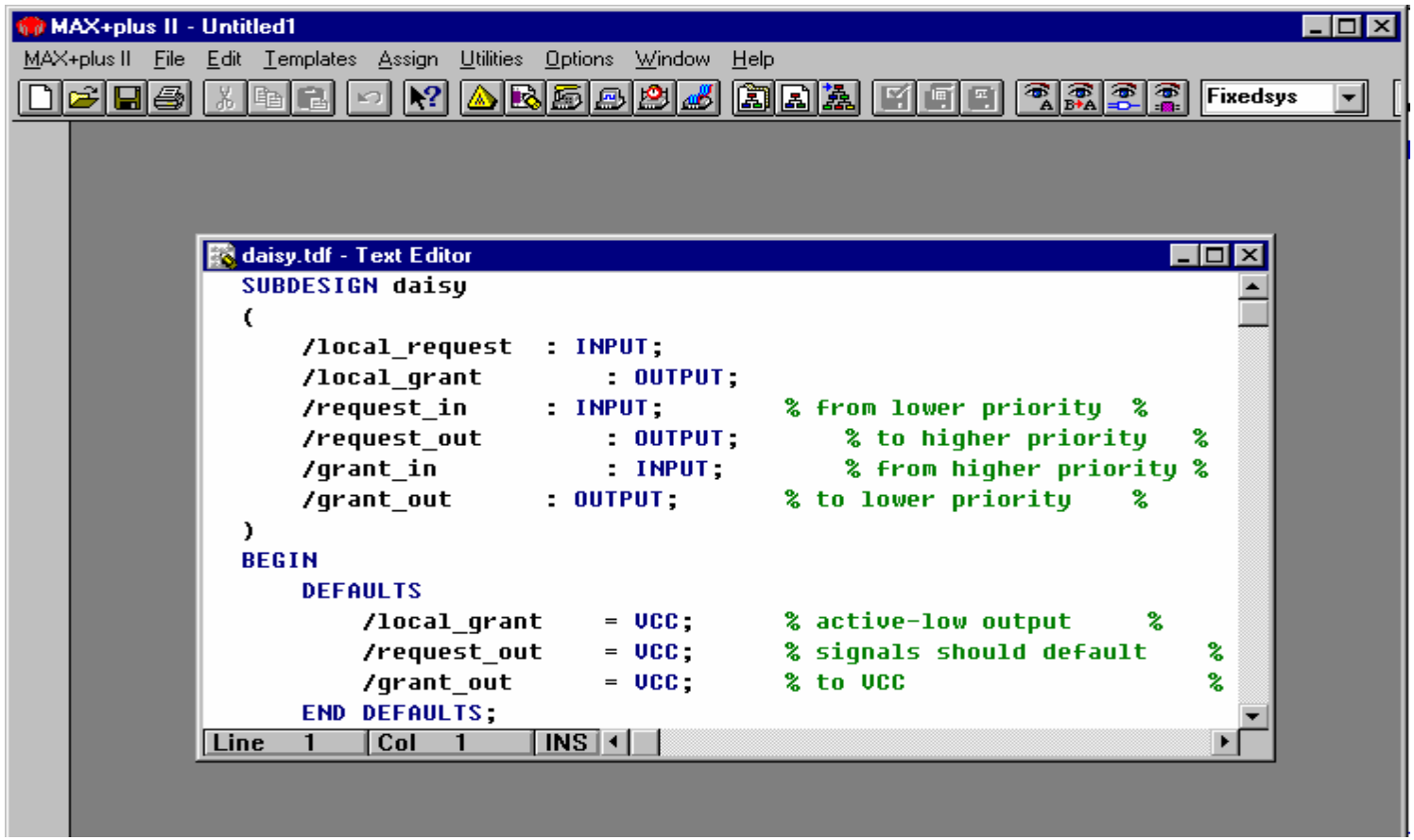
- use any text editor to create the file
  - Altera Software Max+Plus II provide text editor

Click the button



continue…..

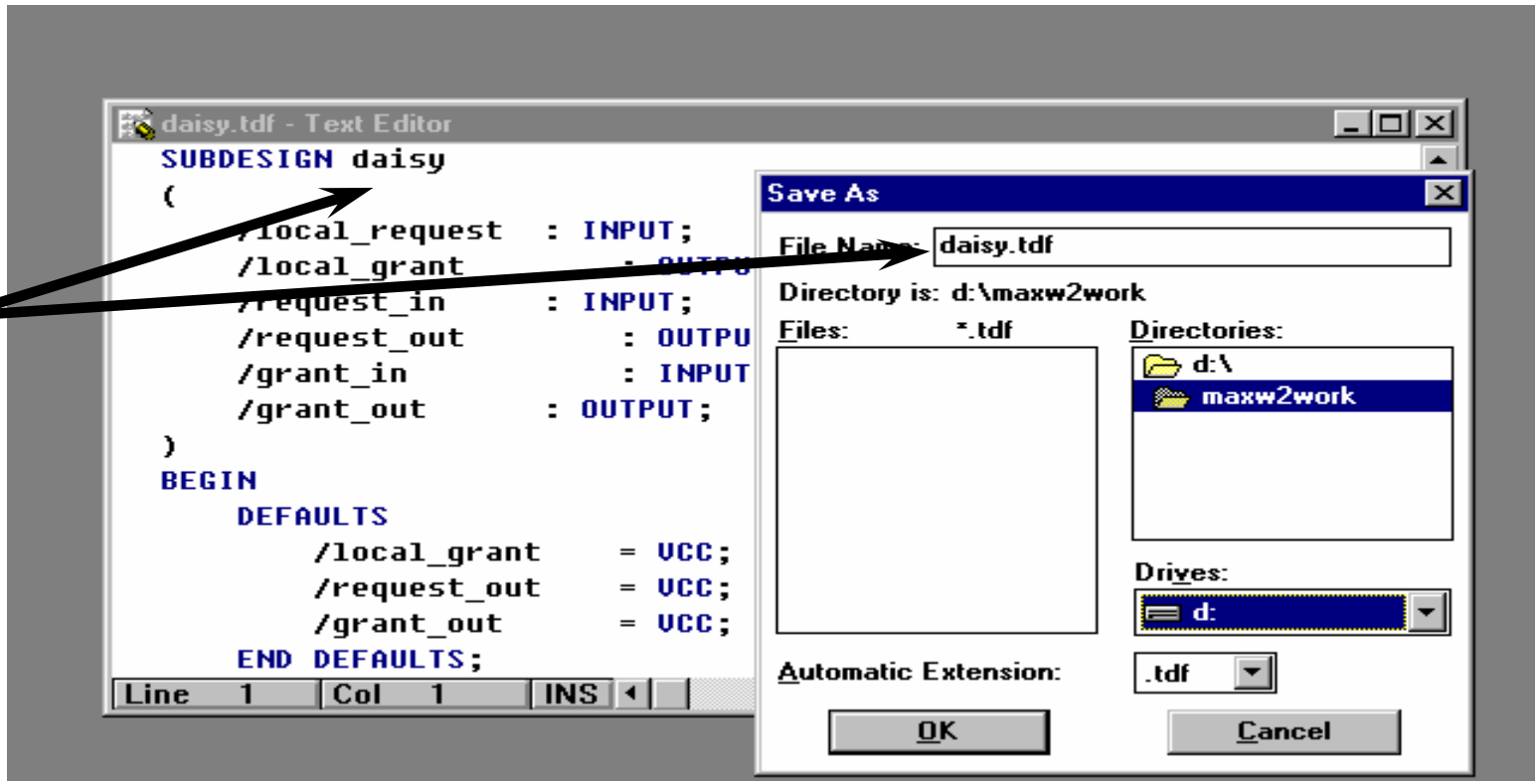
- Create your AHDL file



continue…..

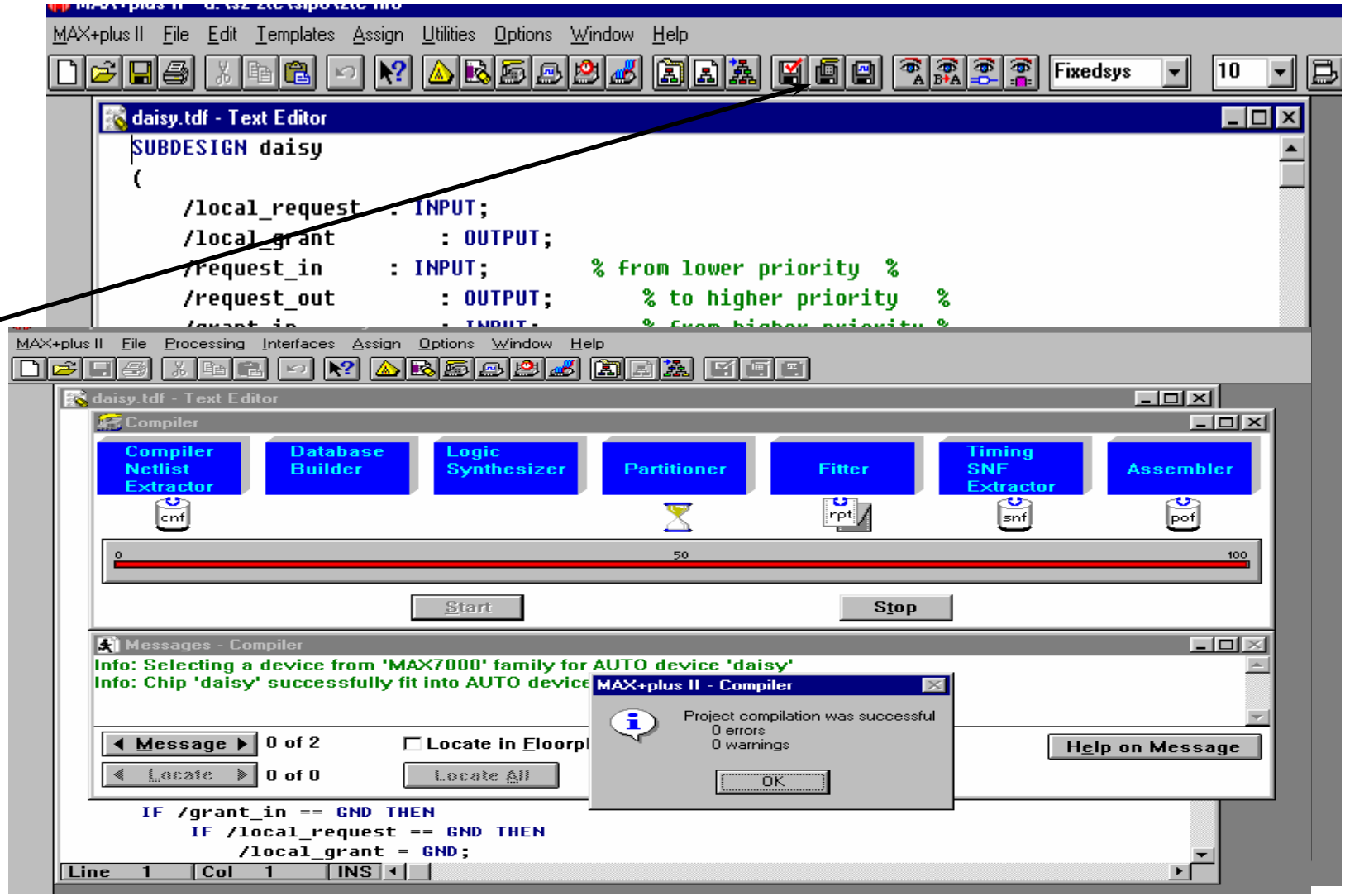
- save your ADHL file as name.TDF

Must be the same



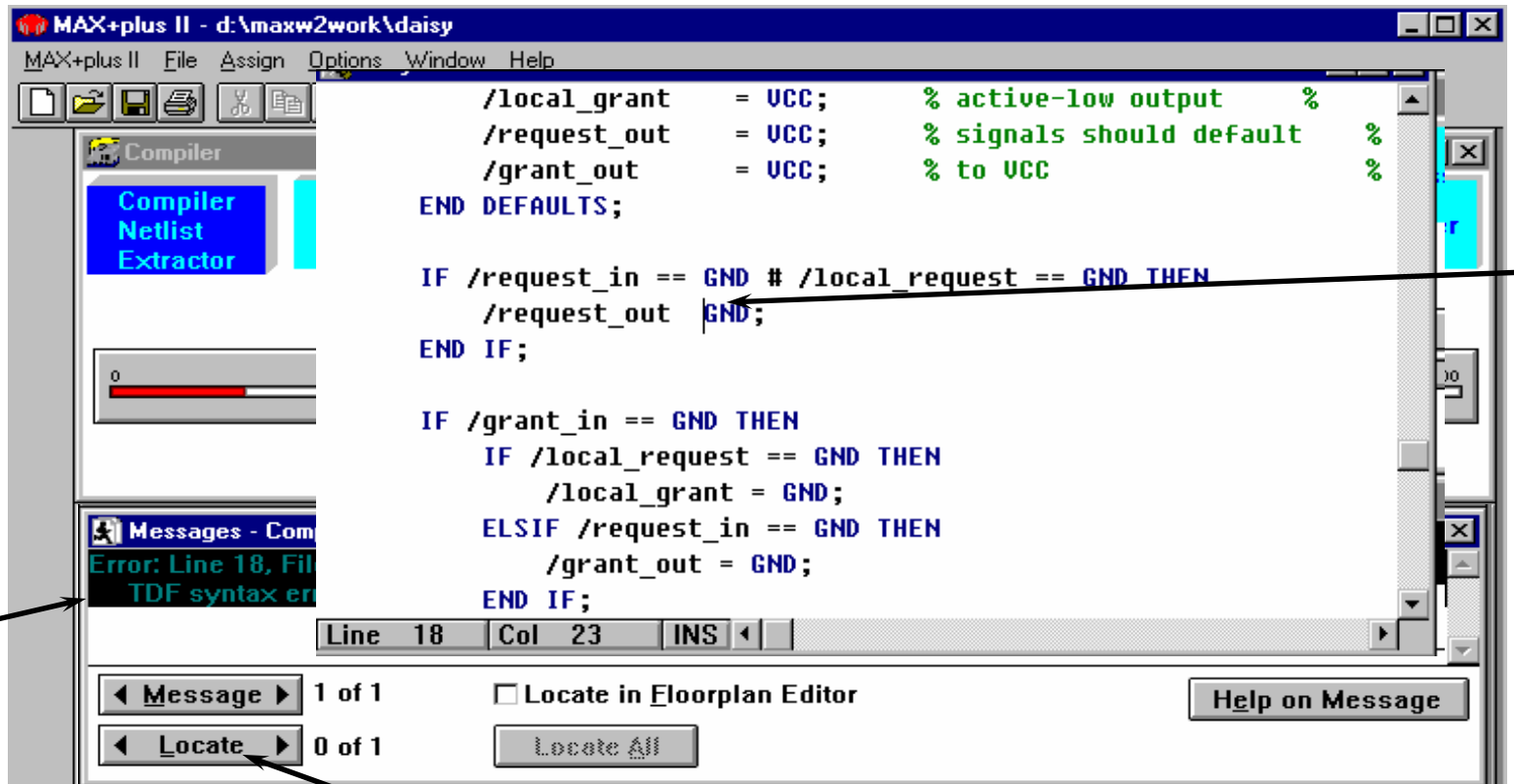
continue...

Click on this icon



Error Location during Compilation

- Easy to locate the error



Error location

Click the error message

Click the Locate button

AHDL Template

I forgot …….

If-then-else
case-end case
loop-end loop
??…???

MAX+plus II

MAX+plus II - d:\maxw2work\daisy - [Untitled6 - Text Editor]

MAX+plus II File Edit Templates Assign Utilities Options Window Help

```
IF _expression THEN
    _statement;
    _statement;
ELSIF _expression THEN
    _statement;
    _statement;
ELSE
    _statement;
    _statement;
END IF;
```

Modify the code

The ALTERA Advantage

General AHDL Format

```
SUBDESIGN decode1
( input_pin_name : INPUT;
  input_bus_name[15..0] : INPUT;
  output_pin_name : OUTPUT;
  output_bus_name : OUTPUT;
)
BEGIN
ouptut_pin_name = input_pin_name;
output_bus_name = input_bus_name;
END;
```

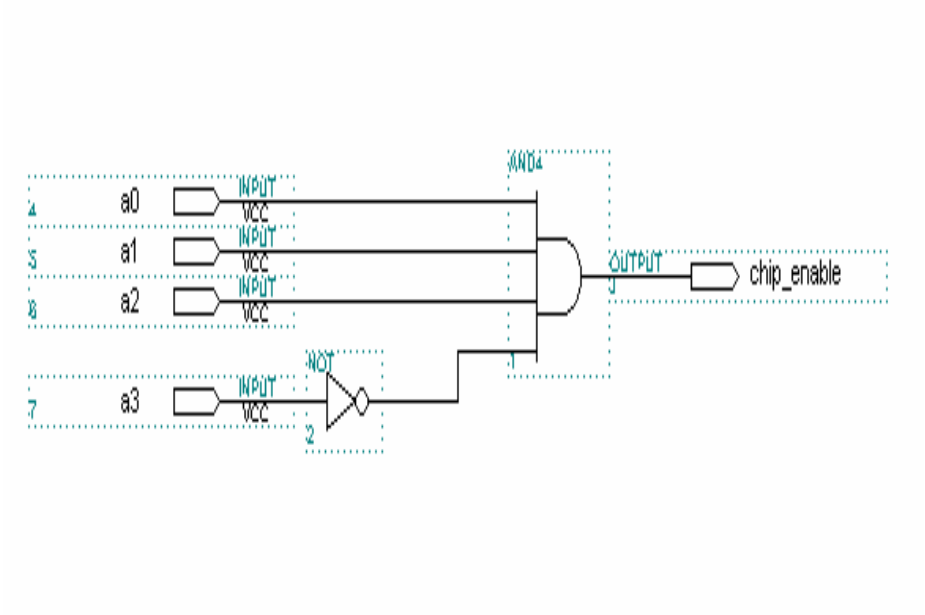
Key Word

Defien I/O port

Logic

AHDL format

Your First AHDL design -- Address Decoder



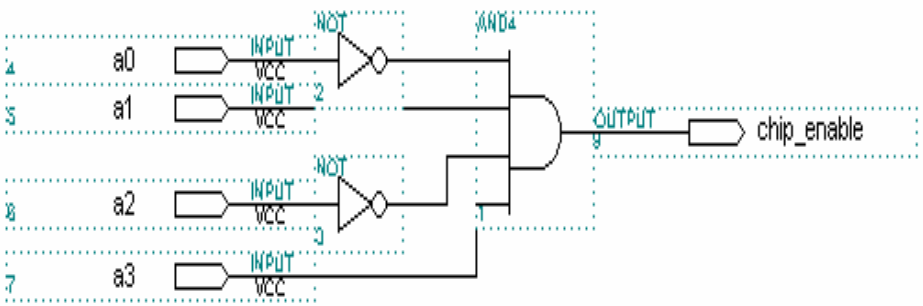

```
SUBDESIGN decode1
( a[3..0] : input;
  chip_enable : output;
)
begin
chip_enable = (a[3..0] == H"7");
end;
```

Chip_enable = a0 & a1 & a2 & !a3

Why I use AHDL instead of Graphic

- Easy to Modify
- Document during the coding
- I want to decode H"A" not H"7"

Chip_enable = !a0 & a1 & !a2 & a3



Need more effort to modify

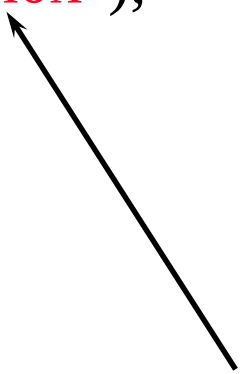
```
SUBDESIGN decode1
( a[3..0] : input;
  chip_enable : output;
)
begin
chip_enable = (a[3..0] == H"A");
end;
```

Self Explain the Function

Only thing to change

More

```
SUBDESIGN decode1
( a[3..0] : input;
  chip_enable : output;
)
begin
chip_enable = (a[3..0] == B"1x0x");
end;
```



Some bit can be ignore for comparsion

Something you need to know

- Addition : +
- Subtraction : -
- Numeric Equality : ==
- Not equal to : !=
- Greater than : >
- Greater than or equal to : >=
- Less than : <
- Less than or equal to : <=
- Logical OR : #
- Logical AND : &

Use Constant Function

- Use Constant if the same number, text string, or arithematic expression is repeated several times in a file
- Advantage
  - Only one statement needs to be changed

Modify One Place will modify all

Define CONSTANT before SUBDESIGN keyword

```
CONSTANT IO_ADDRESS = H"A";

SUBDESIGN decode1
( a[3..0] : input;
  chip_enable : output;
)
begin
chip_enable = (a[3..0] == IO_ADDRESS);
if (a[3..0] == IO_ADDRESS) then
……...
end;
```

More about Constant

- Constant example

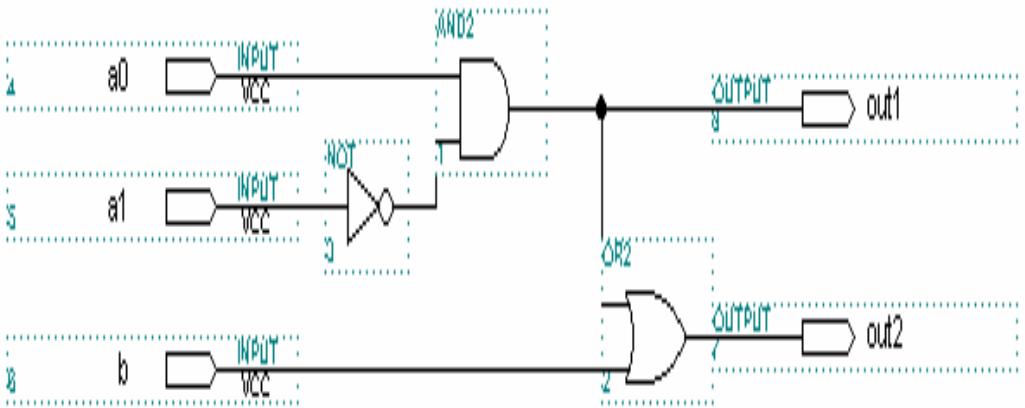
```
Constant IO_ADDRESS = H”370”;
Constant FOO = 1+2*3 - LOG2(256);
Constant FOO_PLUS_one = FOO + 1;
```

Define a constant value

Define an arithematic equation

Depends on pre-define constant

Implement Combinational Logic

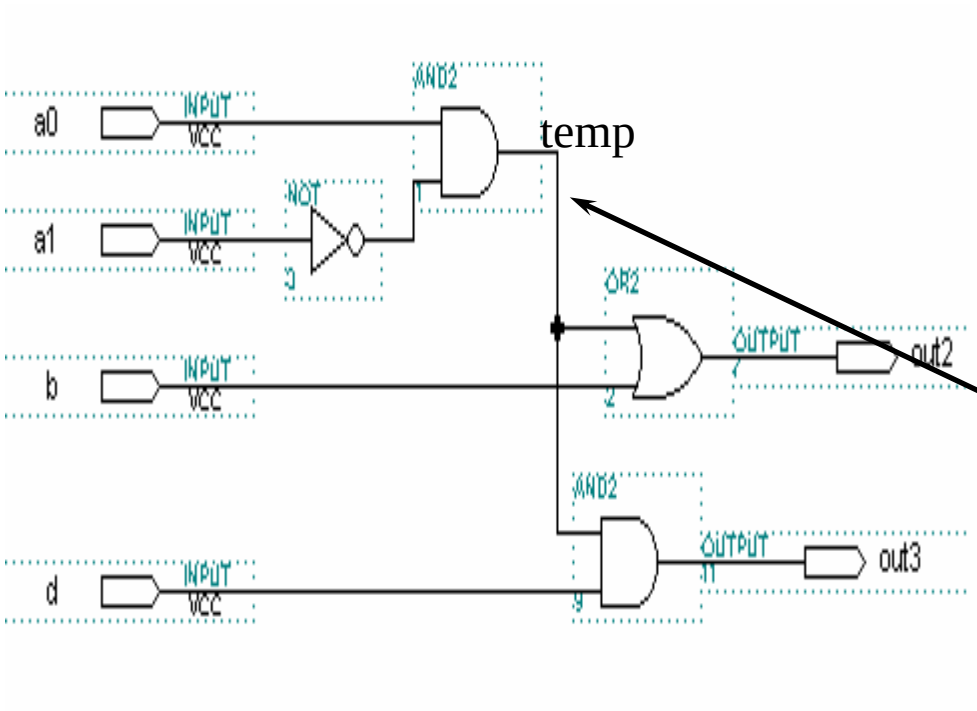


out1 = a0 & !a1
out2 = a0 & !a1 # b

AHDL ?

```
SUBSDESIGN decode1
( a0, a1, b : input;
  out1, out2 : output;
)
begin
out1 = a0 & !a1;
out2 = out1 # b;
end;
```

Define NODEs

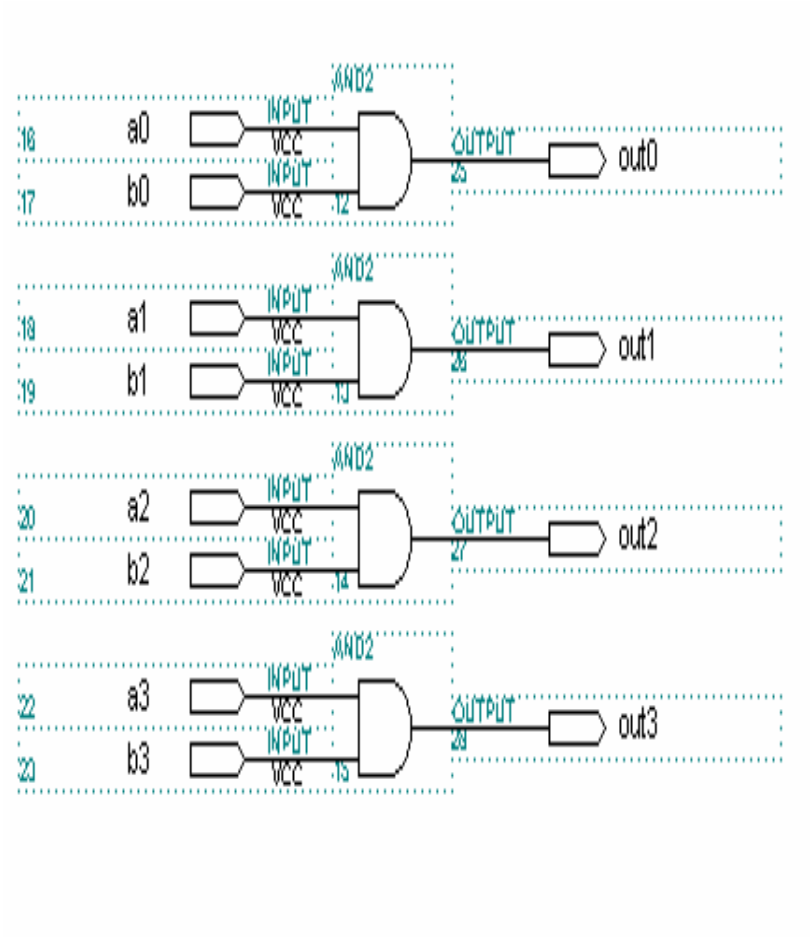


out2 = a0 & !a1 # b
out3 = a0 & !a1 & d

AHDL ?

```
SUBDESIGN decode1
( a0, a1, b, d: input;
  out2, out3 : output;
)
variable
temp : node;
begin
temp = a0 & !a1;
out2 = temp # b;
out3 = temp & d;
end;
```

Bus Operation




```
SUBDESIGN decode1
( a[3..0], b[3..0] : input;
   out[3..0] : output;
)
begin
out0 = a0 & b0;
out1 = a1 & b1;
out2 = a2 & b2;
out3 = a3 & b3;
end;
```

Same function but easier

```
SUBDESIGN decode1
( a[3..0], b[3..0] : input;
  out[3..0]: output;
)
begin
out1[] = a[] & b[];
end;
```

More on Bus Operation

Bus Operation

a[9..0], b[9..0]

- a[] = b[];
- a[7..4] = b[9..6]; a7=b9, a6=b8, a5=b7, a4=b6
- a[9..8] = VCC; a[9..8] connect to VCC
- a[9..8] = 1; a[9..8] = B”01”
- a[9..8] = 2; a[9..8] = B”10”
- a[9..8] = 3; a[9..8] = B”11”
- a[3..0] = GND a[3..0] connect to GND
- a[3..0] = 0; a[3..0] = B”0000”
- temp = b0& b1;
  a[2..1] = temp a2 = temp, a1 = temp

Advance Bus Operation

- Bus
  - b[3..0]
    - b3, b2, b1, b0 (having 4 members)
    - MSB is b3, LSB is b0
- ARRAY BUS
  - a[3..0][2..0]
    - a3_2, a3_1, a3_0, a2_2, a2_1, a2_0, a1_2, a1_1, a1_0, a0_2, a0_1, a0_0 (having 12 members)
    - MSB is a3_2, LSB is a0_0

a[3..2][1..0] = b[];

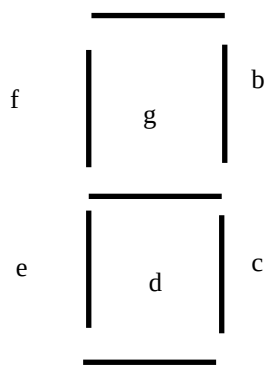
This one change first

a3_1 = b3 a3_0 = b2

a2_1 = b1 a2_0 = b0

Truth Table

i[3..0]	Segment 7
0	0
1	1
2	2
F	F



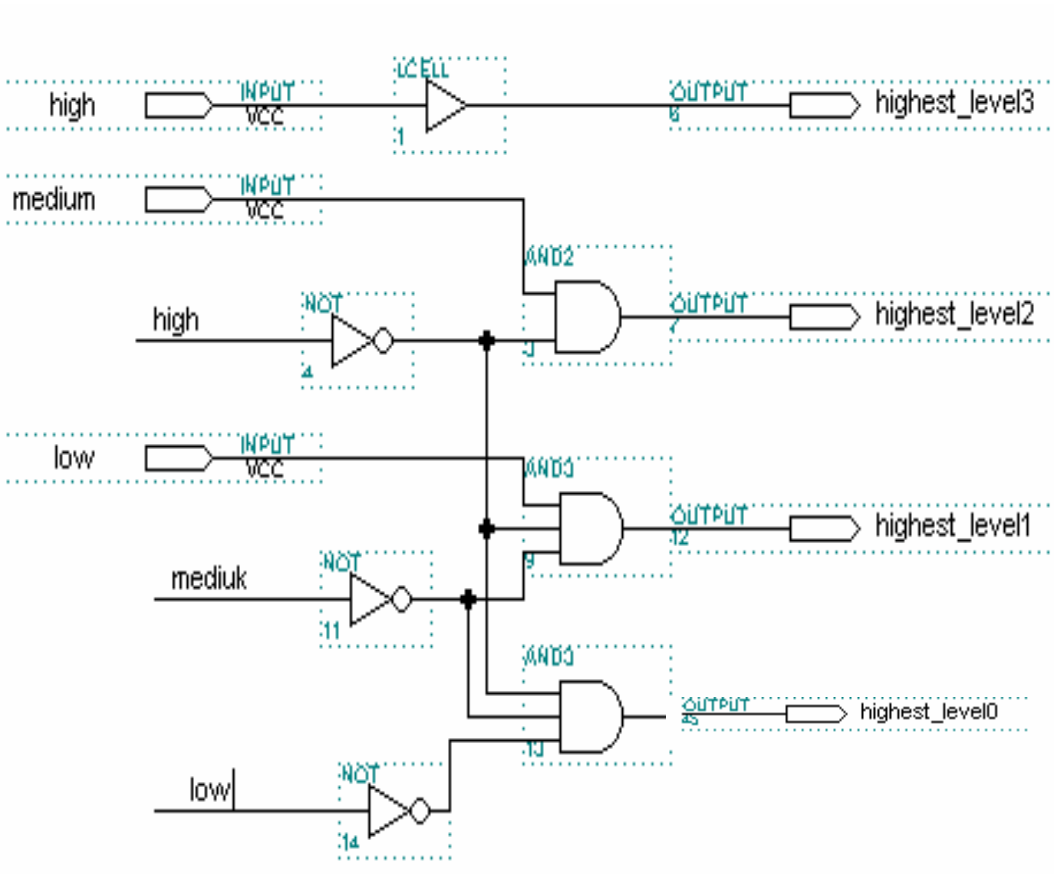

```
SUBDESIGN 7segment
(
    i[3..0]                    : INPUT;
    a, b, c, d, e, f, g : OUTPUT;
)
BEGIN
    TABLE
        i[3..0] =>  a, b, c, d, e, f, g;

        H"0"          =>  1, 1, 1, 1, 1, 1, 0;
        H"1"          =>  0, 1, 1, 0, 0, 0, 0;
        H"2"          =>  1, 1, 0, 1, 1, 0, 1;
        H"3"          =>  1, 1, 1, 1, 0, 0, 1;
        H"4"          =>  0, 1, 1, 0, 0, 1, 1;
        H"5"          =>  1, 0, 1, 1, 0, 1, 1;
        H"6"          =>  1, 0, 1, 1, 1, 1, 1;
        H"7"          =>  1, 1, 1, 0, 0, 0, 0;
        H"8"          =>  1, 1, 1, 1, 1, 1, 1;
        H"9"          =>  1, 1, 1, 1, 0, 1, 1;
        H"A"          =>  1, 1, 1, 0, 1, 1, 1;
        H"B"          =>  0, 0, 1, 1, 1, 1, 1;
        H"C"          =>  1, 0, 0, 1, 1, 1, 0;
        H"D"          =>  0, 1, 1, 1, 1, 0, 1;
        H"E"          =>  1, 0, 0, 1, 1, 1, 1;
        H"F"          =>  1, 0, 0, 0, 1, 1, 1;
    END TABLE;
END;
```

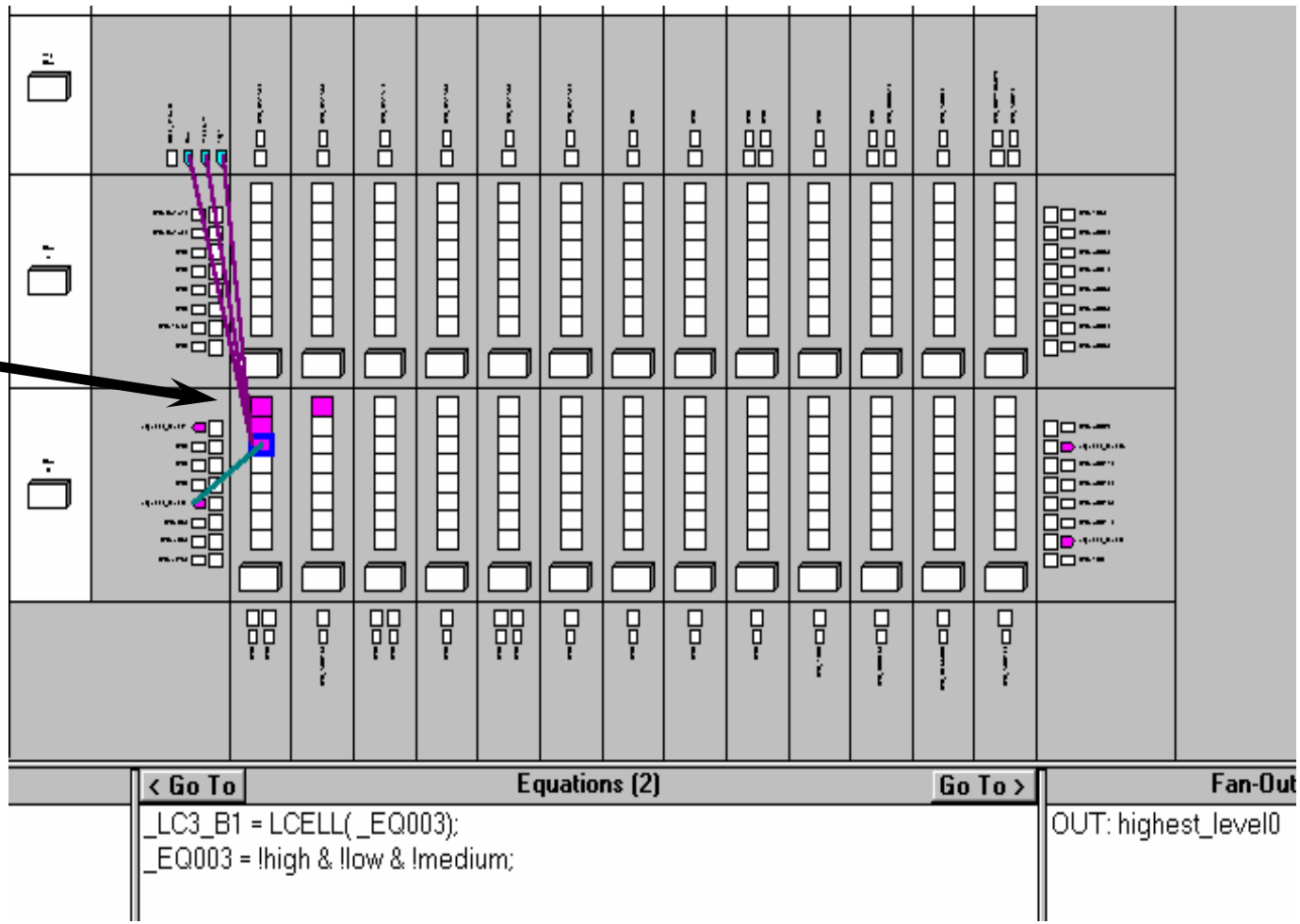
How easy to make any modification

IF-THEN-ELSE

```
SUBDESIGN priority
( low, medium, high : input;
  highest_level[3..0] : output;
)
begin
if ( high == B”1”) then
highest_level[] = B”1000”;
elsif (medium == B”1”) then
highest_level[] = B”0100”;
elsif (low == B”1”) then
highest_level[] = B”0010”;
else
highest_level[] = B”0001”;
end if;
end;
```

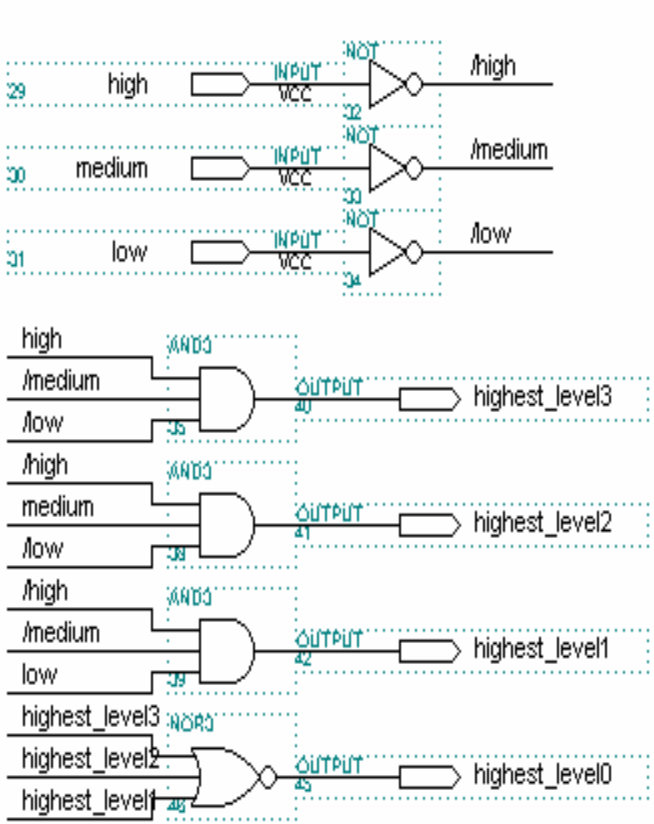


Need 4
LCELL

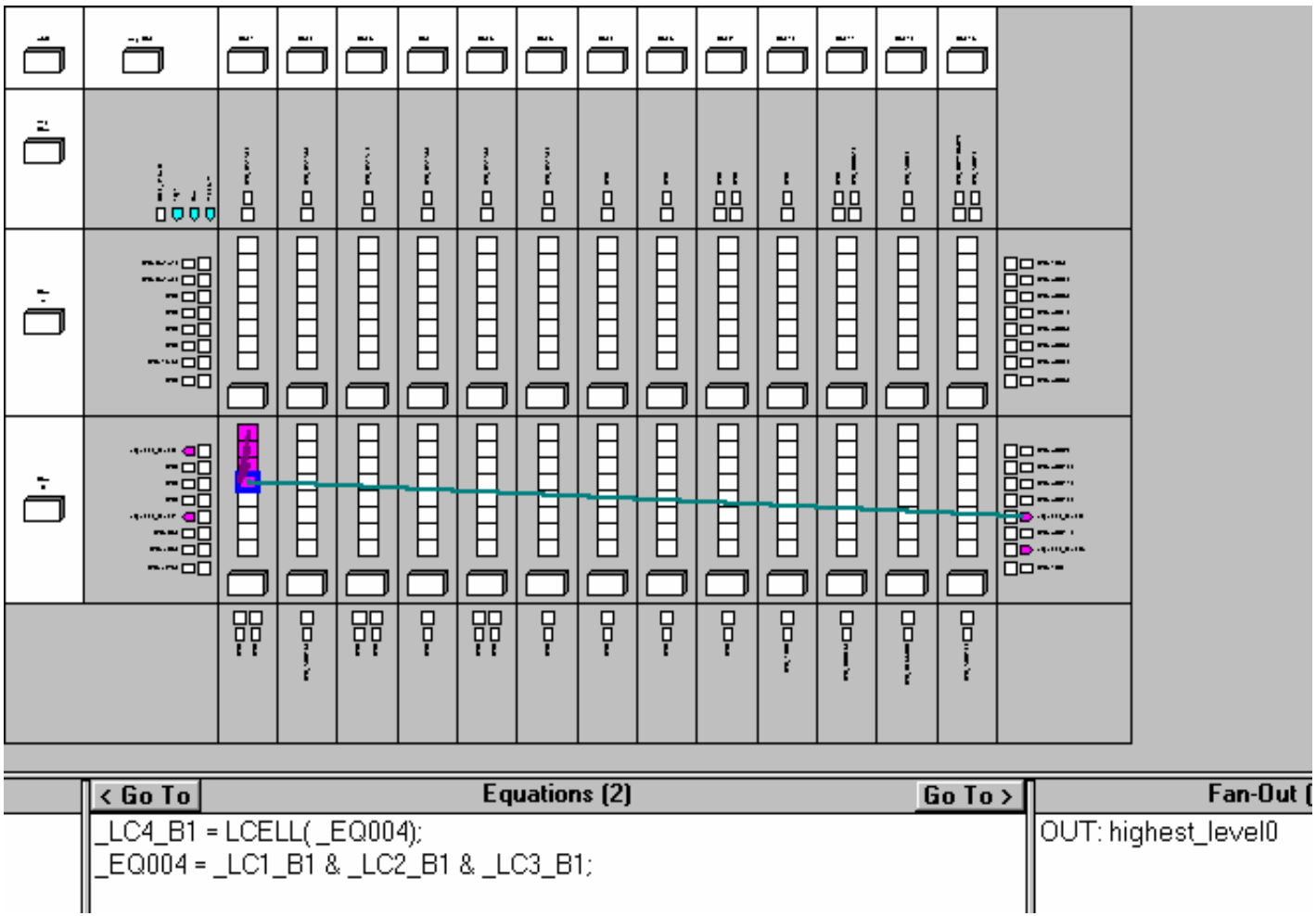


CASE Statement

```
SUBDESIGN decoder
(low, medium, high : input;
  highest_level[3..0] : output;
)
variable
code[2..0] : node;
begin
code2=high;
code1=medium;
code0=low;
case code[] is
when 4 => highest_level[] = B"1000";
when 2 => highest_level[] = B"0100";
when 1 => highest_level[] = B"0010";
when others => highest_level[] = B"0001";
end case;
end;
```

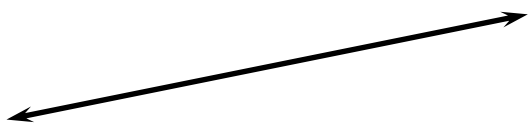


What is the usage of this statement

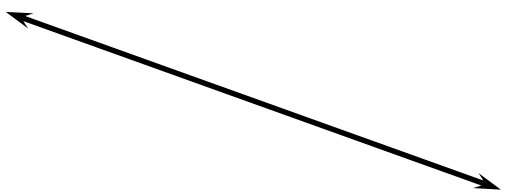


For Loop

```
CONSTANT num_of_bit = 8;
SUBDESIGN numbit
( a[num_of_bit..0] : input;
  b[num_of_bit..0] : output;
)
begin
b[8] = a[0];
b[7] = a[1];
b[6] = a[2];
b[5] = a[3];
b[4] = a[4];
b[3] = a[5];
b[2] = a[6];
b[1] = a[7];
b[0] = a[8];
end;
```



easier with same function

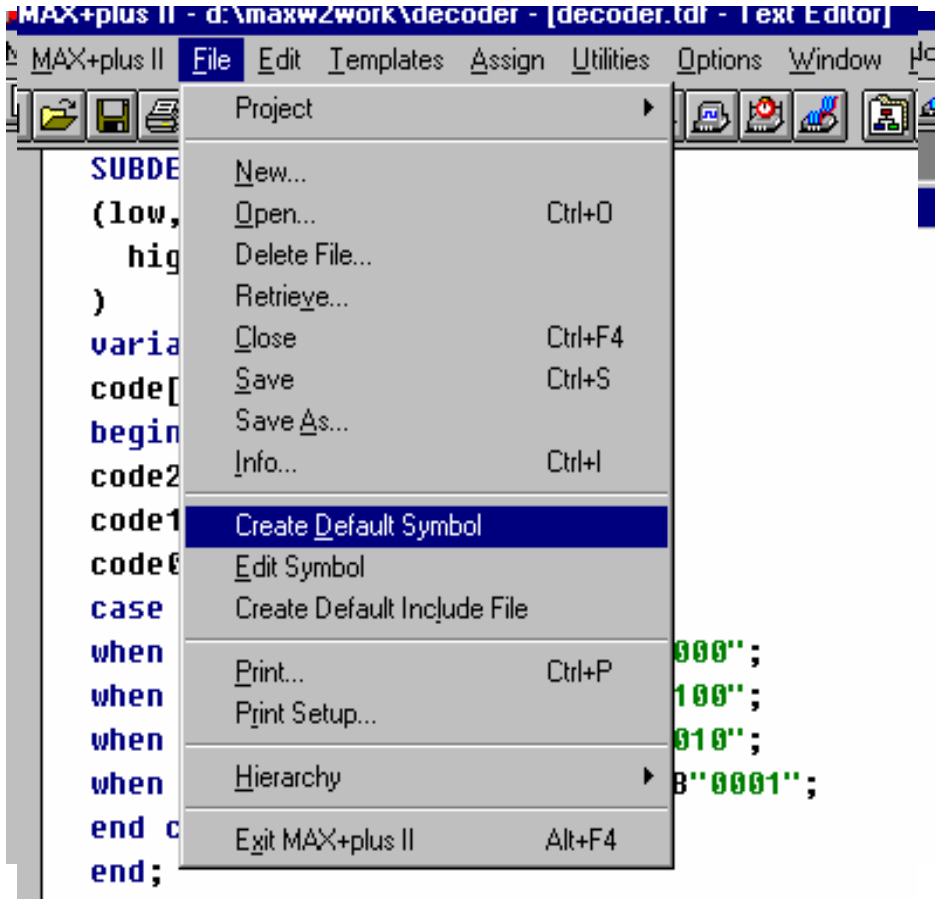


```
CONSTANT num_of_bit = 8;
SUBDESIGN numbit
( a[num_of_bit..0] : input;
  b[num_of_bit..0] : output;
)
begin
for i in 0 to num_of_bit generate
b[num_of_bit - i] = a[i];
end generate;
end;
```

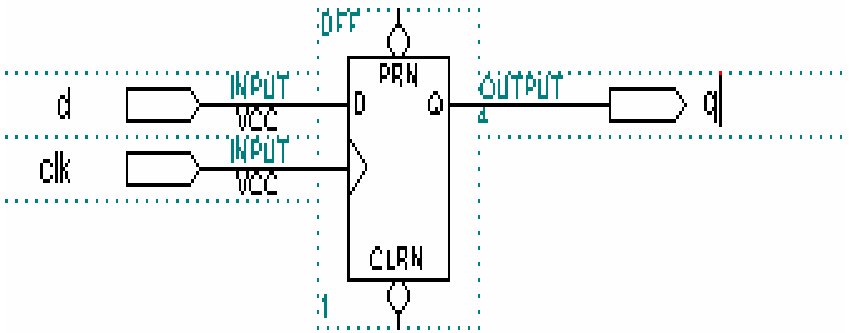
```
CONSTANT num_of_bit = 8;
SUBDESIGN numbit
(a[num_of_bit..0] : input;
 b[num_of_bit..0] : otuput;
)
begin
b[num_of_bit..0] = a[0..num_of_bit];
end;
```

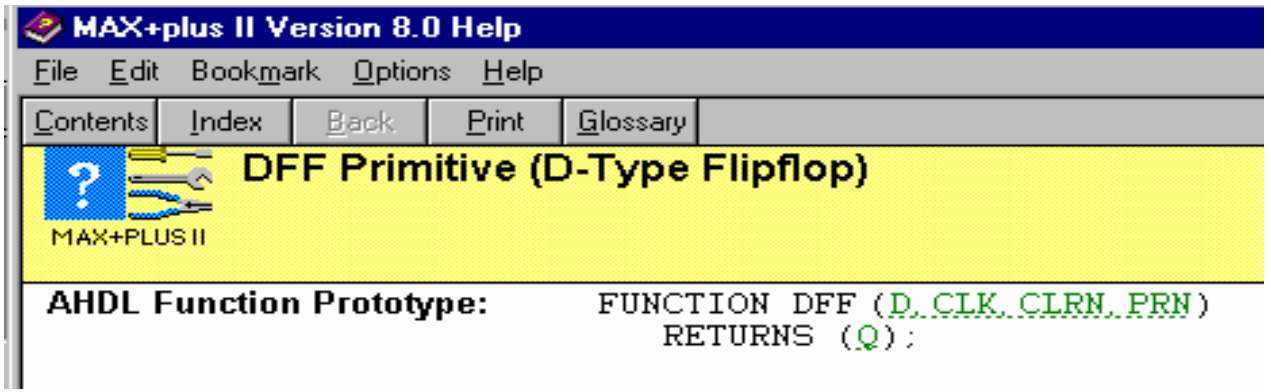
AHDL with Graphic

- Use the File menu to create Symbol for the AHDL design
- The symbol can be used for graphic entry



Register Logic





Method 1

```
SUBDESIGN flip_flop
( d, clk : input;
  q : output;)
begin
q = dff(d,clk, ,);
end;
```

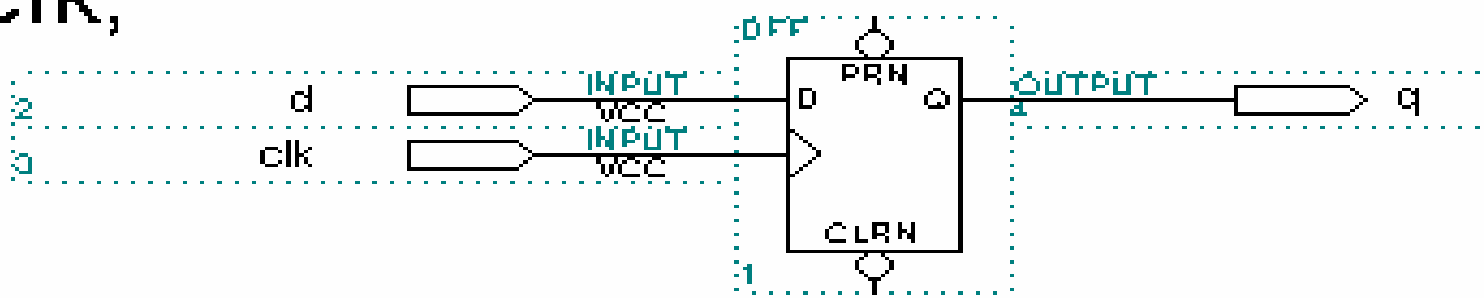
Method 2

```
SUBDESIGN flip_flop
( d, clk : input;
  q : output;)
variable
  temp : dff;
begin
temp.d = d;
temp.clk = clk;
q = temp.q;
end;
```

I want a D-Flipflop

More Detail

```
temp : dff
temp.d = d;
temp.clk = clk;
q = temp.q
```



More on Register

- How to do the Bus with Register
- How to do the other type of Register
  - DFFE (D-FF with enable)
  - TFF/TFFE
  - JKFF/JKFFE
  - SRFF/SRFFE

Register Buses

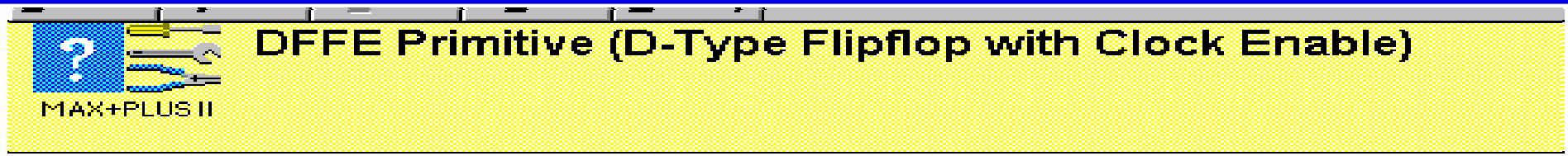
```
SUBDESIGN bus_reg
( clk, d[7..0] : input;
  q[7..0] : output;
)
variable
   ff[7..0] : dff;
begin
ff[].clk = clk;
ff[].d = d[];
q[] = ff[].q;
end;
```

```
ff[0].clk = clk;
ff[1].clk = clk;
ff[2].clk = clk;
ff[3].clk = clk;
ff[4].clk = clk;
ff[5].clk = clk;
ff[6].clk = clk;
ff[7].clk = clk;
```

```
ff[0].d = d[0];
ff[1].d = d[1];
ff[2].d = d[2];
ff[3].d = d[3];
ff[4].d = d[4];
ff[5].d = d[5];
ff[6].d = d[6];
ff[7].d = d[7];
```

```
q[0] = ff[0].q;
q[1] = ff[1].q;
q[2] = ff[2].q;
q[3] = ff[3].q;
q[4] = ff[4].q;
q[5] = ff[5].q;
q[6] = ff[6].q;
q[7] = ff[7].q;
```

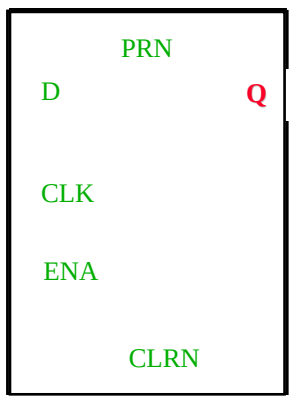
D-Flip-Flop with Enable/Preset/Clear



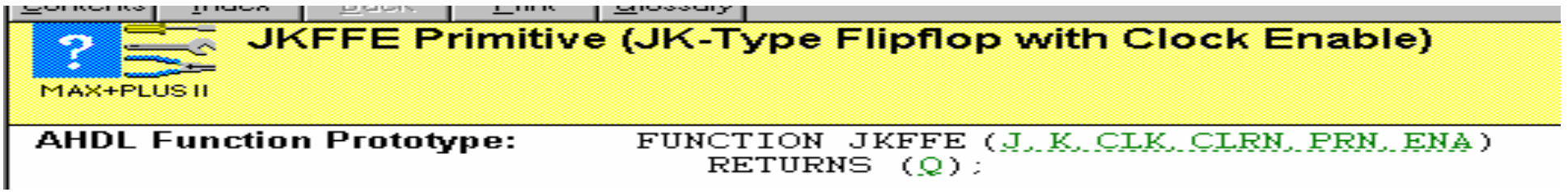
AHDL Function Prototype:

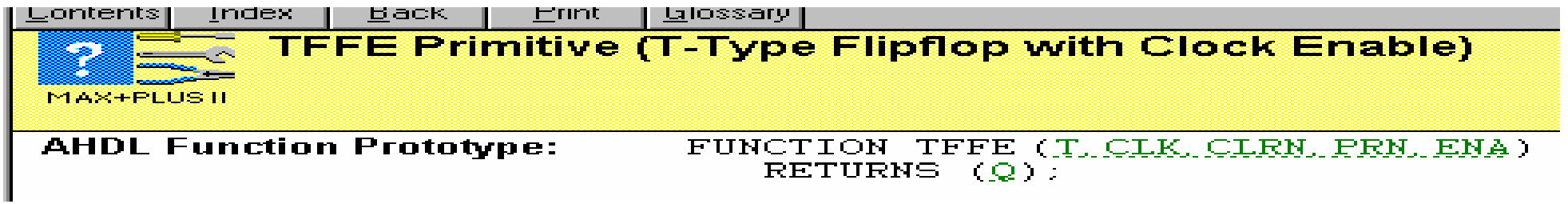
```
FUNCTION DFFE (D, CLK, CLRN, PRN, ENA)
    RETURNS (Q);
```

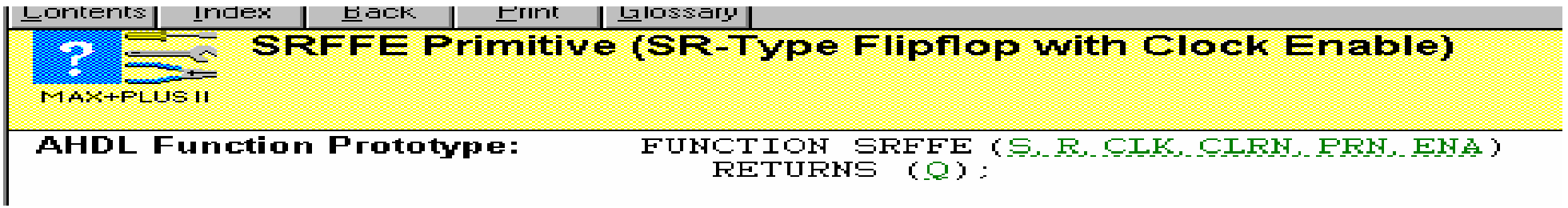
```
SUBDESIGN flip_flop_enable
( clock, data, enable, preset, clear : input;
  qout : output;
)
variable
   temp : dffe;
begin
temp.d = data;
temp.clk = clock;
temp.clrn = clear;
temp.prn = preset;
temp.ena = enable;
qout = temp.q;
end;
```



Other Type of Flip-Flop







How to use Help Menu

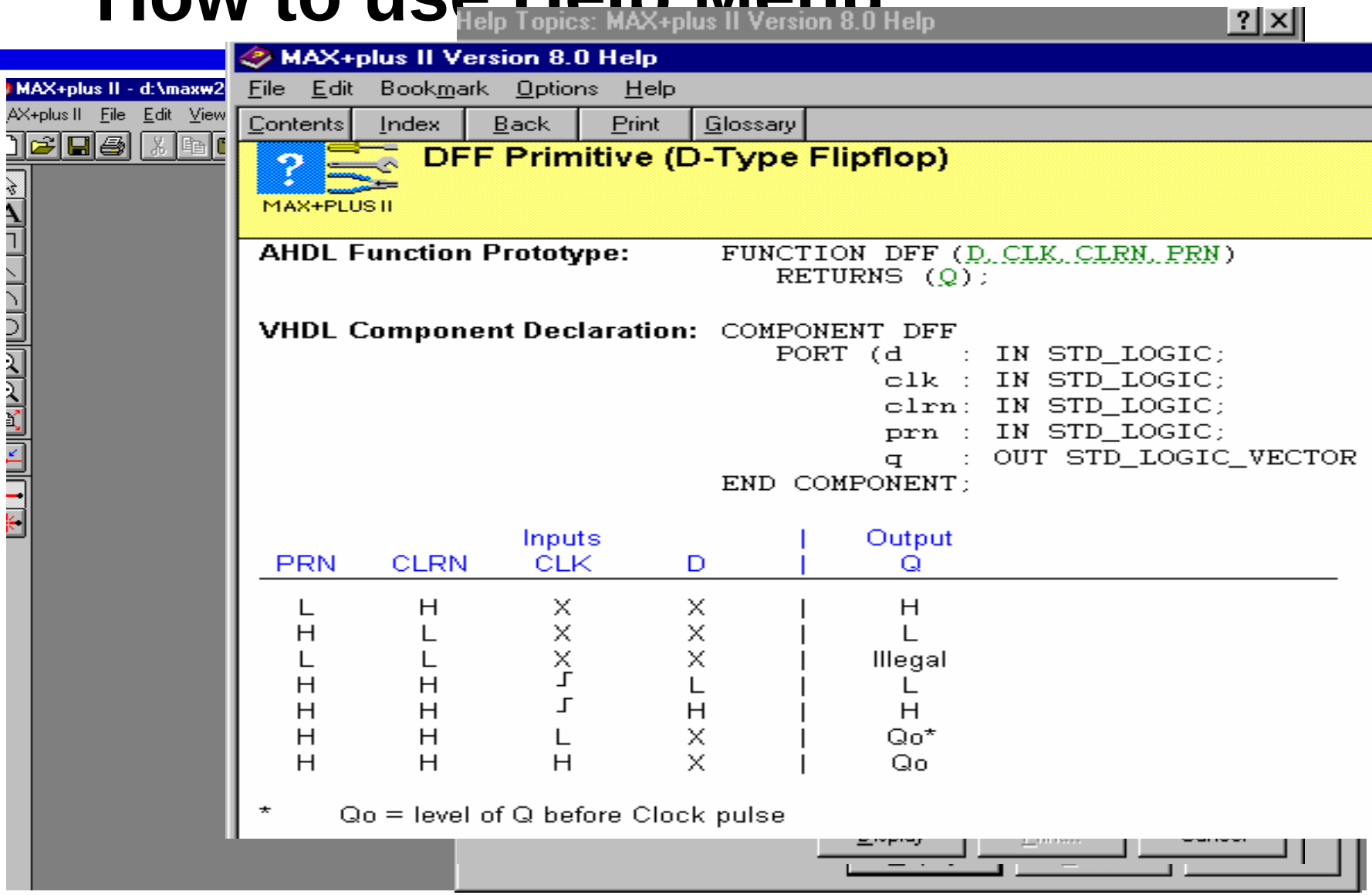
Q : I don’t know how to use Altera DFF/JKFFE……, what can I do ?

A : Altera Help Menu is a good place to find information

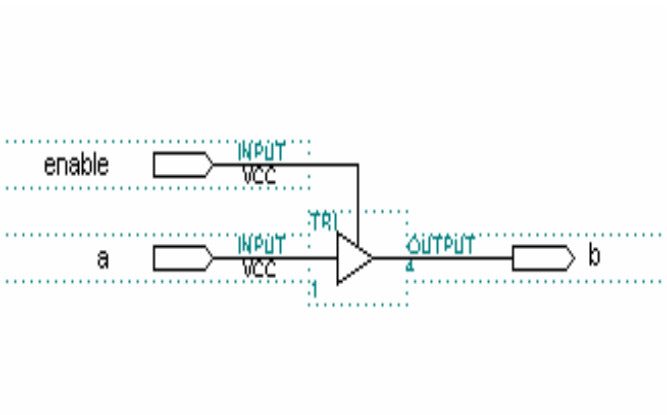
Q : How do I use the Help Menu ?

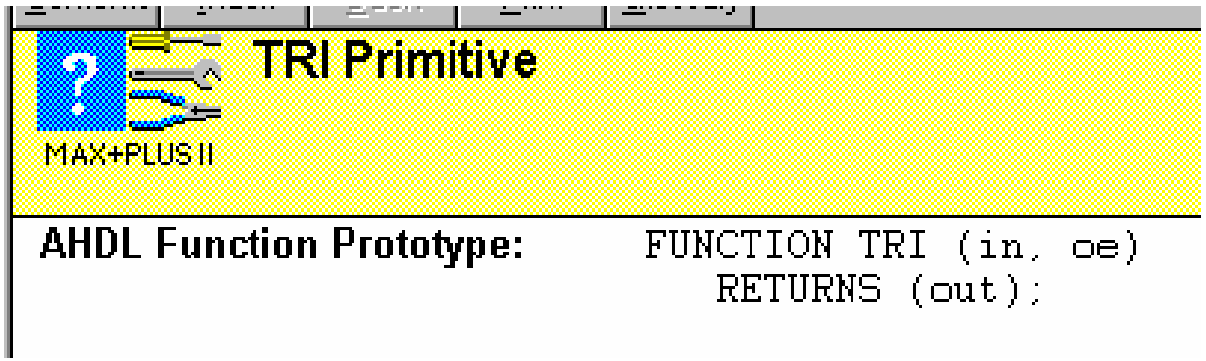
A : It is easy…… and Fun

How to use Help Menu



Tri-state Buffer





Method 1

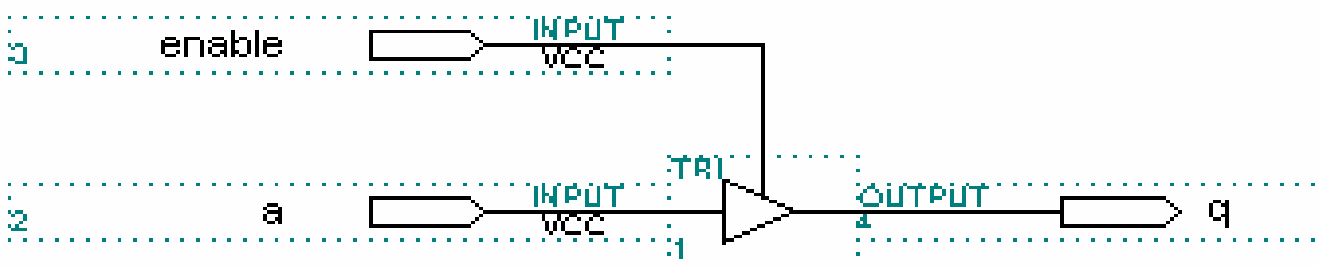
```
SUBDESIGN tri_state
(a, enable : input;
  b : output;)
begin
b = tri(a, enable);
end;
```

Method 2

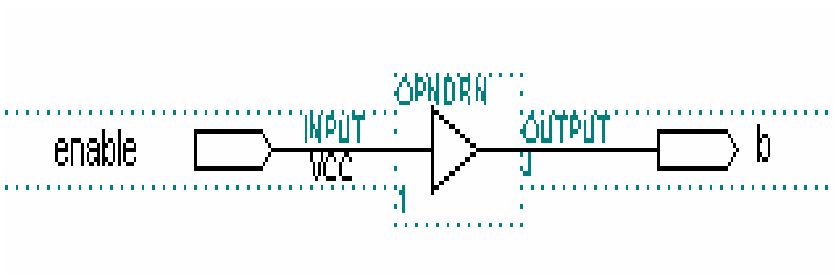
```
SUBDESIGN tri_state
( a, enable : input;
  b : output;)
variable
  temp : tri;
begin
temp.in = a;
temp.oe = enable;
b = temp.out;
end;
```

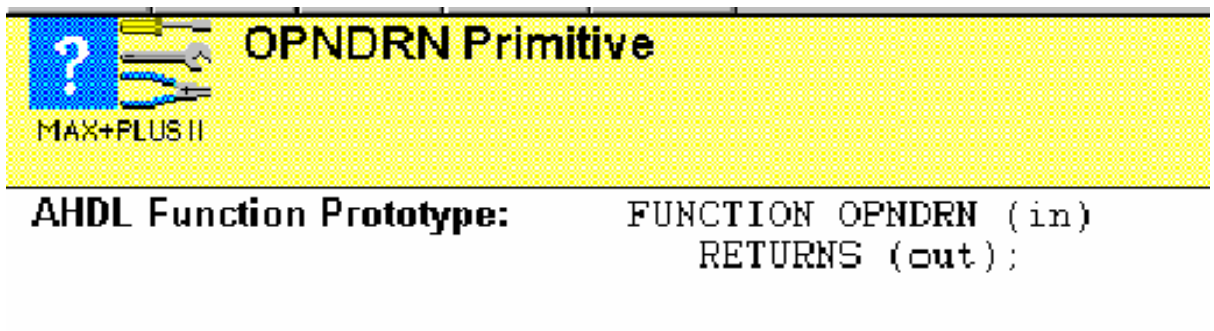
More Detail

```
temp : tri
temp.in = a
temp.oe = enable
q = temp.out
```



OPNDRN - Open Drain Buffer





Method 1

```
SUBDESIGN opn_drn
(enable : input;
  b : output;)
begin
b = opndrn(enable);
end;
```

Method 2

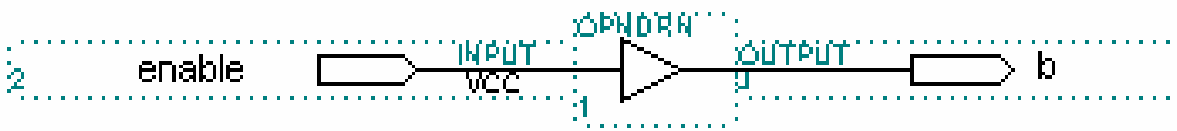
```
SUBDESIGN tri_state
( enable : input;
  b : output;)
variable
  temp : opndrn;
begin
temp.in= enable;
b = temp.out;
end;
```

More Detail

temp : opndrn

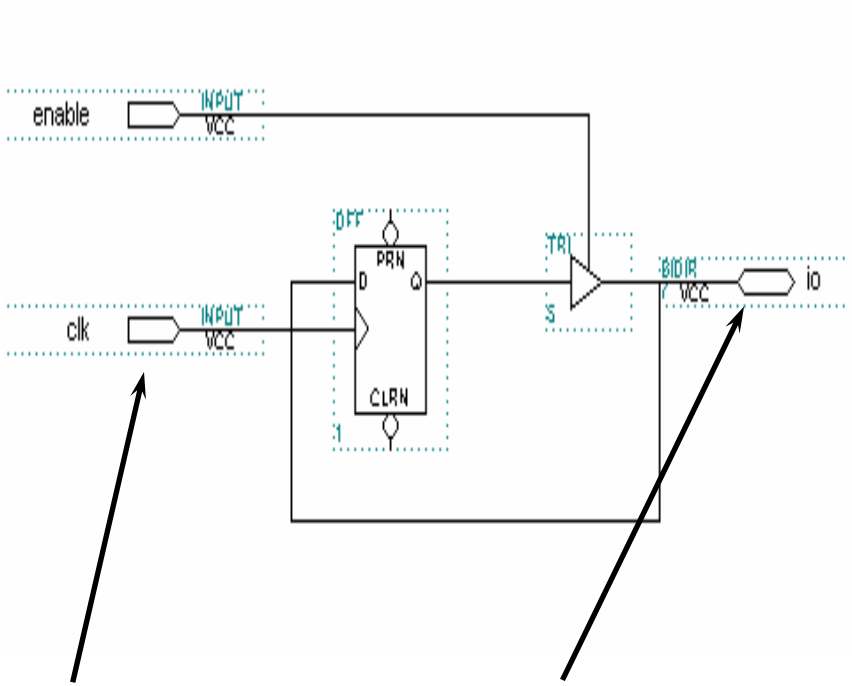
temp.in = enable

b = temp.out



Using AHDL as EASY as Schematic
but
Using AHDL is more POWERFUL

Exercise



input

bidir

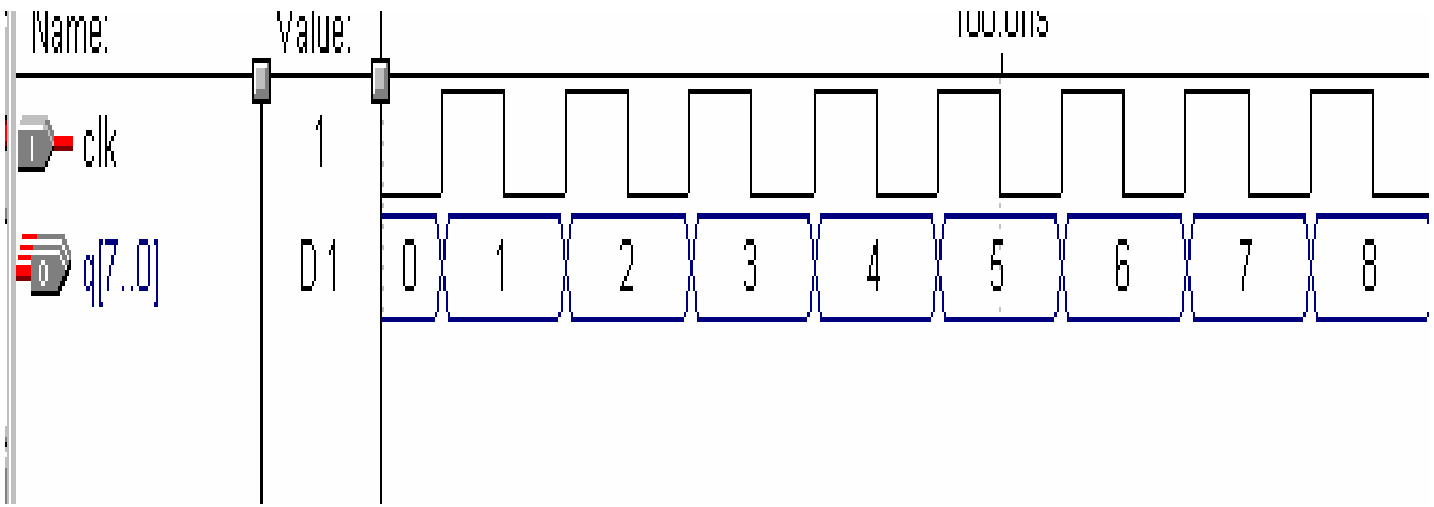
clk, enable : input;
io : bidir;

AHDL

```
SUBDESIGN tri_io
( clk, enable : input;
  io : bidir;)
variable
temp1 : dff;
temp2 : tri;
begin
temp1.d = io;
temp1.clk = clk;
temp2.in = temp1.q;
temp2.oe = enable;
io = temp2.out;
end;
```

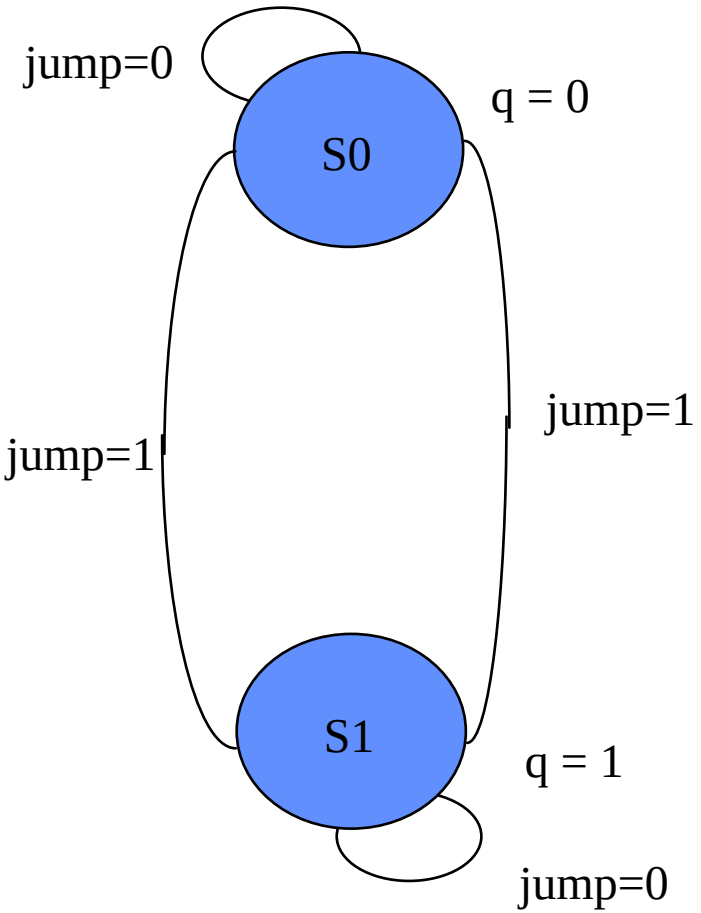
Design 8 bits Counter is Easy

```
SUBDESIGN 8bits
(clk : input;
  q[7..0] : output;
)
variable
   temp[7..0] : dff;
begin
temp[].clk = clk;
temp[].d = temp[].q +1  ;
q[] = temp[].q;
end;
```



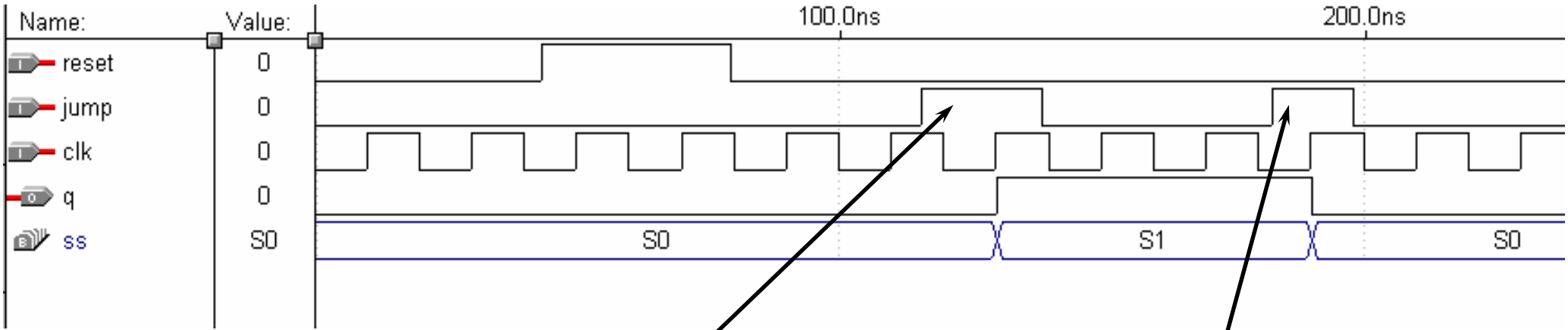
State Machine

State Machine Diagram




```
SUBDESIGN simple
( clk, reset, jump : input;
  q : output;
)
variable
   ss : MACHINE WITH STATES (S0,S1);
begin
ss.clk = clk;
ss.reset = reset;
case ss is
   when s0 =>
                  q = gnd;
                  if (jump) then
                  ss = s1;
                  end if;
   when s1 =>
                  q = vcc;
                  if (jump) then
                  ss = s0;
                  end if;
end case;
end;
```

Note : All State Machine Variable must be associated with a CLOCK



if (jump) then
ss = s1;
end if;

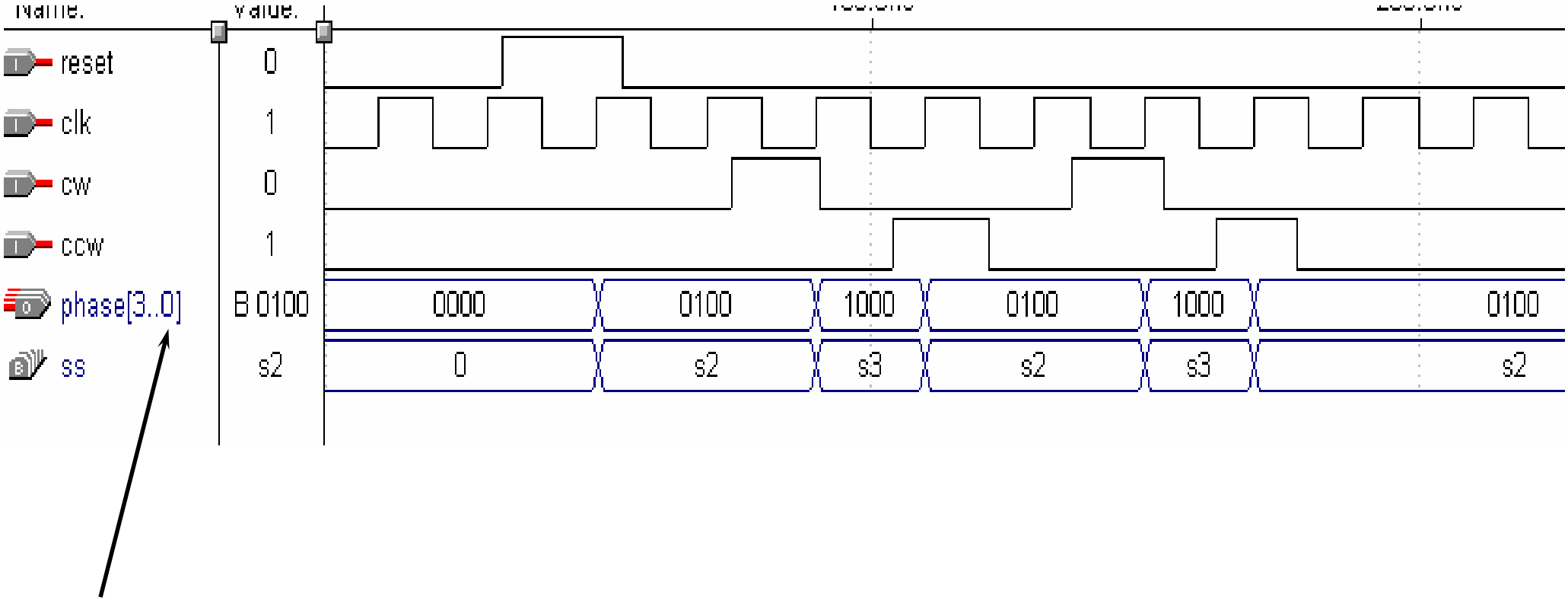
if (jump) then
ss = s0;
end if;

More about State Machine

```
SUBDESIGN stepper
( reset, ccw, cw, clk : input;
  phase[3..0] : output;)
variable
  ss : MACHINE OF BITS (temp[3..0])
       WITH STATES ( s0 = B"0001",
                     s1 = B"0010",
                     s2 = B"0100",
                     s3 = B"1000");
begin
ss.clk = clk;
if (reset) then
ss = s2;
end if;
phase[] = temp[];
```

```
TABLE
    ss,    ccw,    cw  =>   ss;
    s0,    1,      x   =>   s3;
    s0,    x,      1   =>   s1;
    s1,    1,      x   =>   s0;
    s1,    x,      1   =>   s2;
    s2,    1,      x   =>   s1;
    s2,    x,      1   =>   s3;
    s3,    1,      x   =>   s2;
    s3,    x,      1   =>   s0;
END TABLE;
end;
```

Note : No need to declare what is TEMP
It is automatic declare as DFF



User can control the State Bit

Exercise

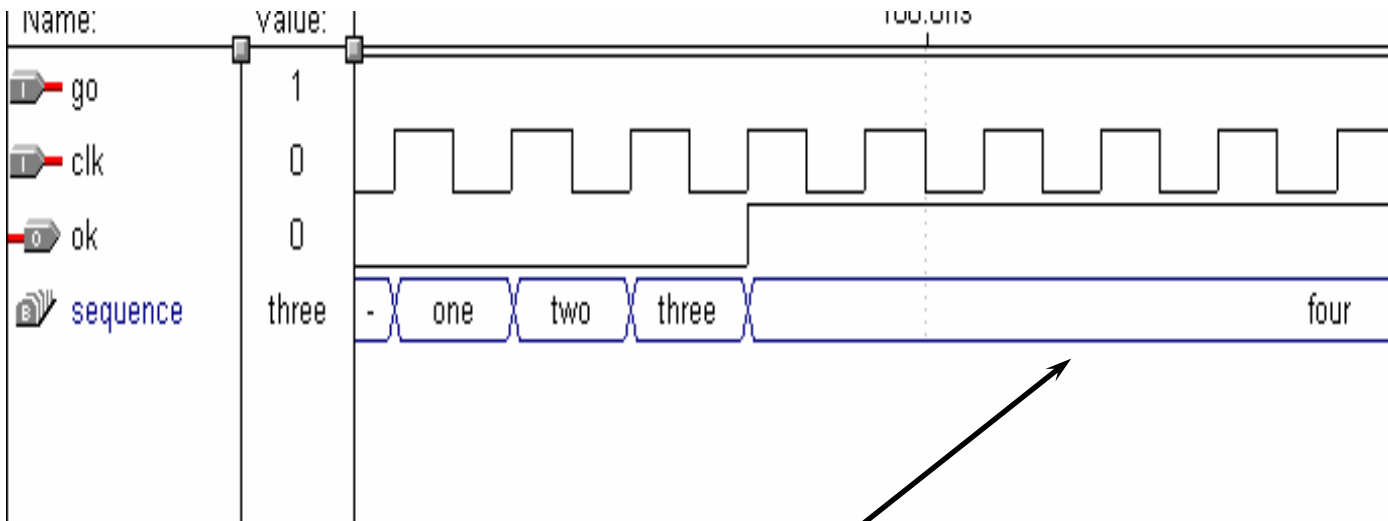
```
SUBDESIGN stepper
( reset, ccw, cw, clk : input;
  phase[3..0] : output;)
variable
  ss : MACHINE OF BITS (temp[3..0])
       WITH STATES ( s0 = B"0001",
                     s1 = B"0010",
                     s2 = B"0100",
                     s3 = B"1000");
begin
ss.clk = clk;
if (reset) then
ss = s2;
end if;
phase[] = temp[];
```

```
TABLE
    ss,    ccw,    cw  =>   ss;
    s0,    1,      x   =>   s3;
    s0,    x,      1   =>   s1;
    s1,    1,      x   =>   s0;
    s1,    x,      1   =>   s2;
    s2,    1,      x   =>   s1;
    s2,    x,      1   =>   s3;
    s3,    1,      x   =>   s2;
    s3,    x,      1   =>   s0;
END TABLE;
end;
```

Can you Modify this Truth Table to CASE statement

State Machine without Recover State

```
SUBDESIGN recover
( clk, go : input;
  ok : output;)
variable
  sequence : MACHINE OF BITS (q[2..0])
             with STATES ( idle, one, two, three, four, illegal1, illegal2, illegal3);
begin
sequence.clk = clk;
case sequence is
when idle => if (go) then
               sequence = one;
               end if;
when one => sequence = two;
when two => sequence = three;
when three => sequence = four;
end case;
ok = (sequence == four);
end;
```



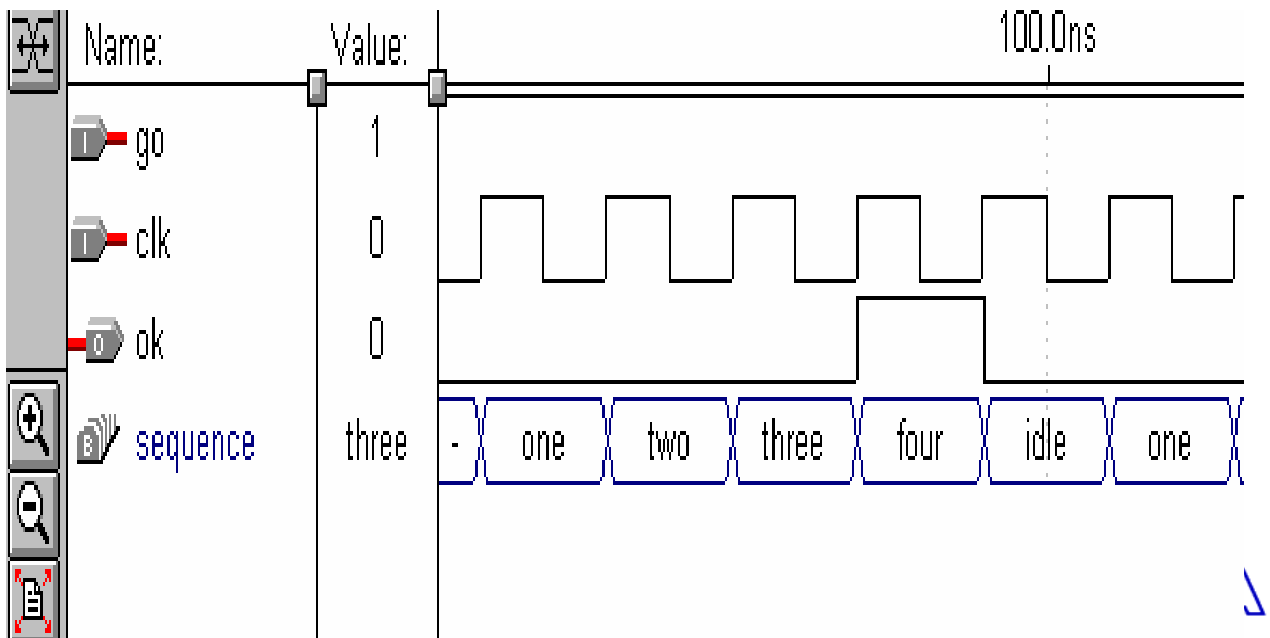
State Machine stuck at FOUR

Better have Recover within State Machine

```
SUBDESIGN recover
( clk, go : input;
  ok : output;)
variable
  sequence : MACHINE OF BITS (q[2..0])
             with STATES ( idle, one, two, three, four, illegal1, illegal2, illegal3);
begin
sequence.clk = clk;
case sequence is
when idle => if (go) then
               sequence = one;
               end if;
when one => sequence = two;
when two => sequence = three;
when three => sequence = four;
when OTHERS => sequence = idle;
end case;
ok = (sequence == four);
end;
```

Three bits have Eight State

Only Five State is interesting,
but better have this RECOVER options



Conclusion

- When the gate count and design getting for complex
  - AHDL
  - VHDL/Verilog
- When the design target for high speed and optimize
  - mixture of AHDL/Graphic/VHDL or Verilog
- Engineers is more talent than software

Right Tools
help you achieve the
Successful Product