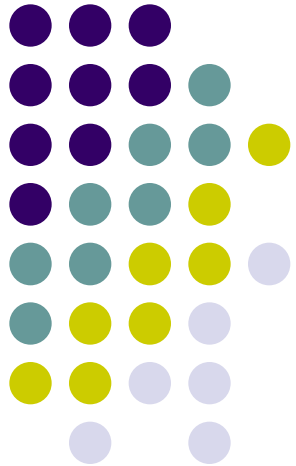
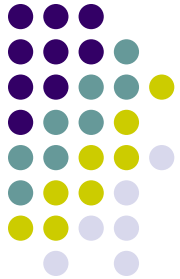
AHDL: Introduction



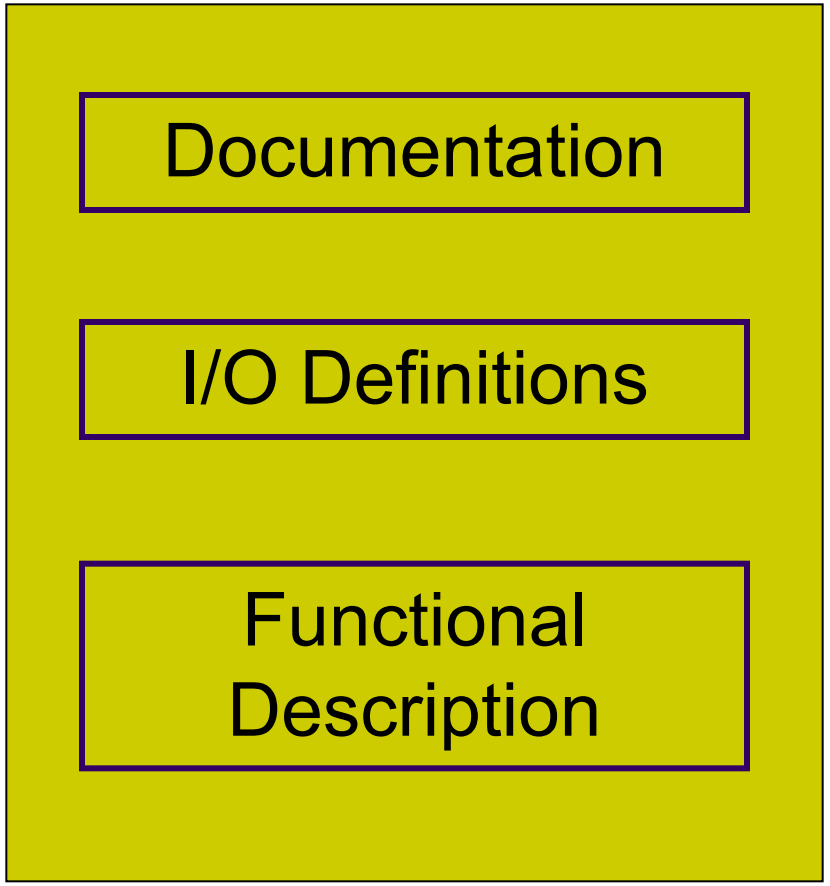
Hardware Description Languages



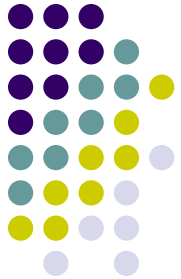
- Altera hardware description language (AHDL)
- Very high speed integrated circuit (VHSIC) hardware description language (VHDL)
- Verilog HDL
- Tool: MAX+PLUS II supports both AHDL and VHDL.

HDL Format and Syntax

- The basic format of any hardware circuit description involves two vital elements:
  - The definition of what goes in and what comes out of it. (I/O specs)
  - The definition of how outputs respond to the inputs. (operation)



AHDL



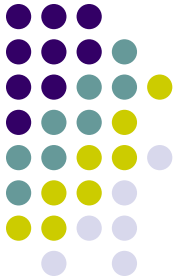
- The textbook includes information on both AHDL and VHDL.
- We will be discussing only the AHDL format and syntax for the sake of simplicity.

Simple Boolean Expression

```
SUBDESIGN and_gate
(
   a,b : INPUT;
   y : OUTPUT;
)
BEGIN
   y = a & b;
END;
```

- Keywords:
  - SUBDESIGN: gives a name to the circuit block. (Not case-sensitive)
  - INPUT
  - OUTPUT
  - BEGIN
  - END

Basic Boolean Operators



- & AND
- # OR
- ! NOT
- $ XOR

Intermediate Variables in AHDL

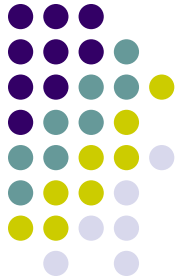
```
%   (Figure 3-49) %
SUBDESIGN fig3_50
(
    a, b, c : INPUT;        -- define input to block
    y : OUTPUT;             --define block output
)
VARIABLE
    m : NODE;   --name an intermediate signal
BEGIN
    m = a & b;    --generate buried product term
    y = m # c;    --generate sum on output
END;
```

Keyword: VARIABLE

Comments:

- Between two %'s
- After two dashes (--)

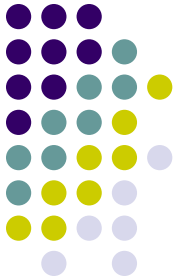
Representing Data in AHDL



- Numbers
  - Binary: B”101”
  - Hexadecimal H”101”
  - Decimal 101
- Bit Array/Vectors
  - p1 [7..0] :INPUT; --define an 8-bit input port
  - Assignment:

```
VARIABLE temp [7..0] : NODE;
BEGIN
  temp[]=p1[];
END
```

Define a Truth Table in AHDL



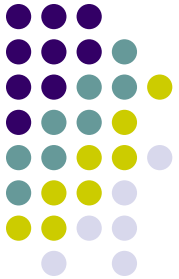
```
SUBDESIGN FIG4_50
(
    a,b,c :INPUT;    --define inputs to block
    y :OUTPUT;       --define block output
)
BEGIN
    TABLE
        (a,b,c)    => y;      --column headings
        (0,0,0)    => 0;
        (0,0,1)    => 0;
        (0,1,0)    => 0;
        (0,1,1)    => 1;
        (1,0,0)    => 0;
        (1,0,1)    => 1;
        (1,1,0)    => 1;
        (1,1,1)    => 1;
    END TABLE;
END;
```

Keywords:

TABLE

END TABLE

Decision Control: IF/THEN/ELSE



```
SUBDESIGN FIG4_54
(
    digtal_value[3..0] :INPUT;    -- define inputs to block
    z :OUTPUT; --define block output
)

BEGIN
    IF digital_value[] > 6 THEN
        z = VCC;          -- output a 1
    ELSE          z = GND;          -- output a 0
    END IF;
END;
```

Keywords:

IF THEN

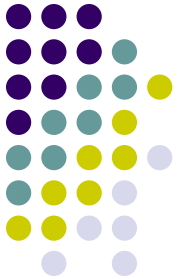
ELSE

END IF

ELSEIF

```
SUBDESIGN fig4_58
(
    digtal_value[3..0] :INTPUT;  --define inputs to block
    too_cold, just_right, too_hot :OUTPUT;       --define outputs
)
VARIABLE
status[2..0] :NODE;      --holds state of too_cold, just_right, too_hot
BEGIN
    IF digital_value[] <= 8 THEN status[] = b"100";
    ELSEIF digital_value[] > 8 AND digital_value[] < 11 THEN
          status[] = b"010";
    ELSE          status[] = b"001";
    END IF;
    (too_cold, just_right, too_hot) = status[];      --update output bits
END;
```

CASE/WHEN



```
SUBDEDSIGN fig4_60
(
    p,q,r :INPUT;          --define inputs to block
    s : OUTPUT;            --define outputs
)
VARIABLE
    status[2..0] :NODE;
BEGIN
    status[] = (p, q, r);  --link input bits in order
    CASE status[] IS
        WHEN b"100"  => s = GND;
        WHEN b"101"  => s = GND;
        WHEN b"110"  => s = GND;
        WHEN OTHERS        => s = VCC;
    END CASE
END;
```

Keywords:

CASE

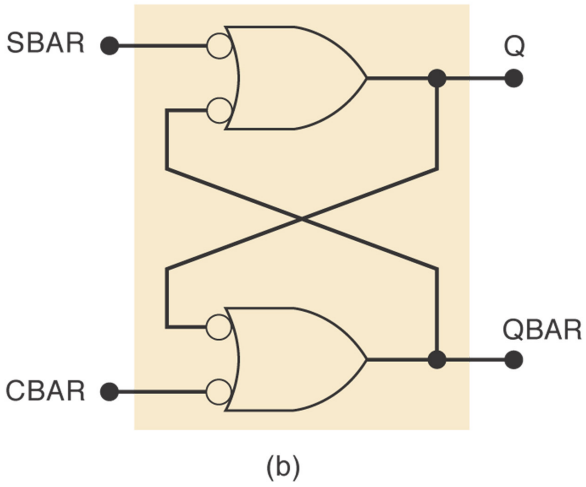
WHEN

OTHERS

END CASE

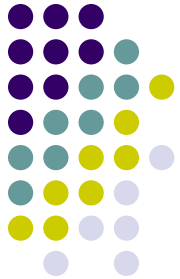
The NAND Latch

```
SUBDESIGN fig5_61
(
    sbar, cbar : INPUT; -- active low inputs
    q, qbar : OUTPUT; -- allow feedback
)
BEGIN
    q = !sbar # !qbar;    -- Boolean equations
    qbar = !cbar # !q;
END;
```



(b)

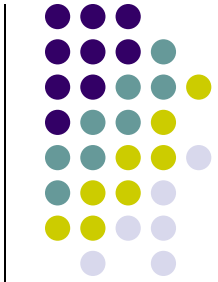
AHDL Flip-Flops



- JK, D, SR and latch registers are available for use in AHDL.

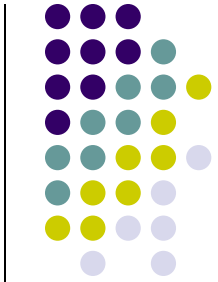
Standard Part Function	Primitive Identifier
Clock input	clk
Asynchronous preset (active-LOW)	prn
Asynchronous clear (active-LOW)	clrn
J,K,S,R,D inputs	j,k,s,r,d
Level triggered ENABLE input	ena
Q output	q

Single JK Flip-Flop



```
SUBDESIGN fig5_64
(
    jin,kin,clkin,preset,clear      :INPUT;
    qout                            :OUTPUT;
)
VARIABLE
    ff1    :JKFF; -- define the FF as JKFF type
BEGIN
        ff1.prn=preset;  --optional, default to vcc
        ff1.clrn=clear;
        ff1.j=jin;
        ff1.k=kin;
        ff1.clk=clkin;
        qout=ff1.q;
END;
```

MOD 8 Ripple Up Counter



```
SUBDESIGN fig5_71
(
    clock : INPUT;
    q[2..0] : OUTPUT;
)
VARIABLE
    q[2..0] : JKFF;          --defines three JK FFs
BEGIN
    q[2..0].j = VCC;         -- toggle mode J=K=1 for all FFs
    q[2..0].k = VCC;
    q[0].clk = !clock;
    q[1].clk = !q[0].q;
    q[2].clk = !q[1].q;      -- connect clocks in ripple form
END;
```