EEE205 – Digital Electronics (II)

Lecture 5

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU

In This Session

- Representing Data in AHDL
- Operators
- Truth Tables Using AHDL
- Decision Control Structures in AHDL

Identifiers

- Composed of legal characters
  - Letters (a–z, A–Z)
  - Digits (0–9)
  - Slash (/)
  - Underscore (_)
- Can begin with a digit
- Must not be a reserved word
- Identifiers and keywords are **case-insensitive**.

Representing Data in AHDL

- Binary, hexadecimal, and decimal values are represented in AHDL as shown, with a prefix to indicate the number system.

Number System	AHDL	Decimal Equivalent
Binary	B”101”	5
Hexadecimal	H”101”	257
Decimal	101	101

- Constants: **VCC** and **GND**

Representing Data in AHDL

Bit Arrays or **Bit Vectors** are used to describe a port or a node with more than one data bit.

Declaration

- Syntax – a name is followed by the index range.
- Example:

```
p1 [7..0]     :INPUT;  -- DEFINE AN 8-BIT INPUT
PORT
```

- Individual bits can be accessed by specifying the bits, e.g. p1[5].

Representing Data in AHDL

Bit Array Assignment

- To assign the 8-bit port p1 to a node named temp:

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

- The empty braces mean that all bits in the array are being connected.
- Individual bits could also be connected, e.g. `temp[0]=p1[0];`

Operators

Boolean Operators

Operator	Alternate	Description
!	NOT	Inverter
&	AND	AND
! &	NAND	AND with Inverted Output
#	OR	OR
! #	NOR	OR with Inverted Output
$	XOR	Exclusive OR
! $	XNOR	Exclusive OR with Inverted Output

Arithmetic Operators

Operator	Description
– (unary)	Two's Complement Negation
+	Unsigned/Two's Complement Addition
–	Unsigned/Two's Complement Subtraction

Operators

Comparison Operators

Operator	Type	Description
==	Logical	Equal to
!=	Logical	Not equal to
<	Arithmetic	Less than
<=	Arithmetic	Less than or equal to
>	Arithmetic	Greater than
>=	Arithmetic	Greater than or equal to

Operators

The Concatenation Operator

- Combine nodes (and other groups) into a group by placing the node/group names in parentheses

  g[3..0] = (a, b, c, d);

  (a, b, c, d) = g[3..0];

- Can also be used to assign multiple ports in a single statement

  (a, b) = B"10";

Truth Tables Using AHDL

- Circuits can be designed directly from truth tables in AHDL.
- The key point is to use the **TABLE** keyword.

a	b	c	y
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

Truth Tables Using AHDL

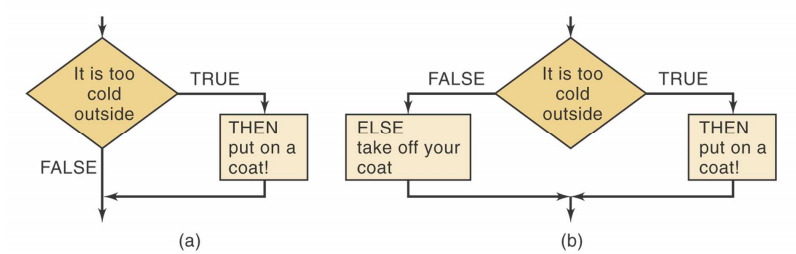
```
SUBDESIGN FIG4-50
(
     a,b,c  :INPUT;    --define inputs to block
     y      :OUTPUT;   --define block output
)
BEGIN
     TABLE
          (a,b,c)    =>  y; --column headings
          (0,0,0)    =>  0;
          (0,0,1)    =>  0;
          (0,1,0)    =>  0;
          (0,1,1)    =>  1;
          (1,0,0)    =>  0;
          (1,0,1)    =>  1;
          (1,1,0)    =>  1;
          (1,1,1)    =>  1;
     END TABLE;
END;
```


Truth Tables Using AHDL

```
SUBDESIGN FIG4-50
(
     a,b,c  :INPUT;    --define inputs to block
     y      :OUTPUT;   --define block output
)
VARIABLE in_bits[2..0] :NODE;
BEGIN
     in_bits[] = (a,b,c);  --concatenating
     TABLE
          in_bits[]  =>  y; --column headings
          b"000"     =>  0;
          b"001"     =>  0;
          b"010"     =>  0;
          b"011"     =>  1;
          b"100"     =>  0;
          b"101"     =>  1;
          b"110"     =>  1;
          b"111"     =>  1;
     END TABLE;
END;
```

Decision Control Structures in AHDL

- IF/THEN/ELSE statements provide a framework for making logical decisions in a system.
  - **IF/THEN** is used when there is a choice between doing something and doing nothing.
  - **IF/THEN/ELSE** is used when there is a choice between two possible actions.



Decision Control Structures in AHDL

- The IF/THEN/ELSE in AHDL:

```
SUBDESIGN FIG4_54
(
   digital_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;
```


Decision Control Structures in AHDL

- The **CASE** construct determines the value of an expression and then goes through a list of values (cases) to determine what action to take, e.g. to implement truth table:

p	q	r	s
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Decision Control Structures in AHDL

- The CASE construct in AHDL:

```
SUBDESIGN 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;
```