Introduction to Chapter 4

- Basic logic gate functions will be combined in *combinational* logic circuits.
- Simplification of logic circuits will be done using Boolean algebra and a mapping technique.
- Troubleshooting of combinational circuits will be introduced.
- PLD and HDL control structures will be explained.

4-1 Sum-of-Products Form

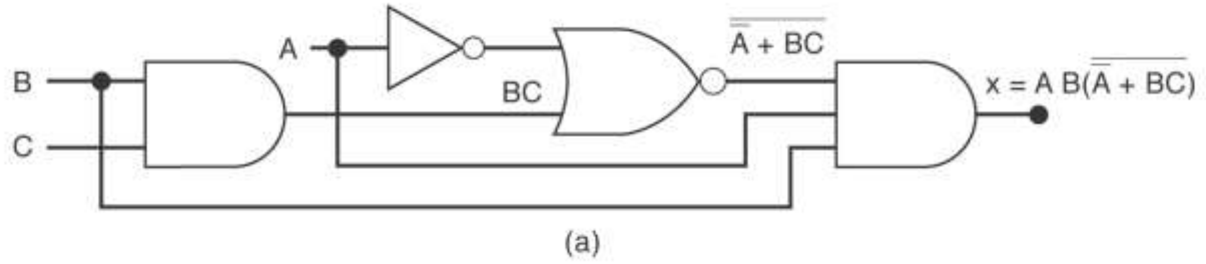
- A Sum-of-products (SOP) expression will appear as two or more AND terms ORed together.

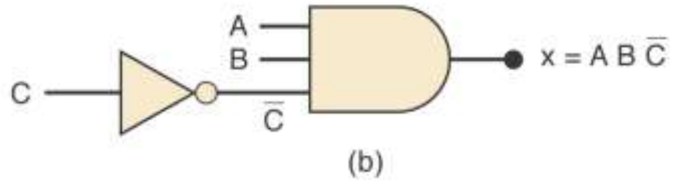
$$ABC + \overline{A}B\overline{C}$$

$$AB + \overline{A}B\overline{C} + \overline{C}\,\overline{D} + D$$

4-2 Simplifying Logic Circuits

- The circuits below both provide the same output, but the lower one is clearly less complex.


A
B
C
BC
$\overline{\overline{A} + BC}$
$x = AB(\overline{\overline{A} + BC})$
(a)



- We will study simplifying logic circuits using Boolean algebra and Karnaugh mapping

4-3 Algebraic Simplification

- Place the expression in SOP form by applying DeMorgan’s theorems and multiplying terms.
- Check the SOP form for common factors and perform factoring where possible.
- Note that this process may involve some trial and error to obtain the simplest result.

4-4 Designing Combinational Logic Circuits

- To solve any logic design problem:
  - Interpret the problem and set up its truth table.
  - Write the AND (product) term for each case where the output equals 1.
  - Combine the terms in SOP form.
  - Simplify the output expression if possible.
  - Implement the circuit for the final, simplified expression.

4-5 Karnaugh Map Method

- A graphical method of simplifying logic equations or truth tables. Also called a K map.
- Theoretically can be used for any number of input variables, but practically limited to 5 or 6 variables.

4-5 Karnaugh Map Method

- The truth table values are placed in the K map as shown in figure 4-11.
- Adjacent K map square differ in only one variable both horizontally and vertically.
- The pattern from top to bottom and left to right must be in the form $\overline{A}\,\overline{B}, \overline{A}B, AB, A\overline{B}$
- A SOP expression can be obtained by ORing all squares that contain a 1.

4-5 Karnaugh Map Method

- Looping adjacent groups of 2, 4, or 8 1s will result in further simplification.
- When the largest possible groups have been looped, only the common terms are placed in the final expression.
- Looping may also be wrapped between top, bottom, and sides.
- Figures 4-13 and 4-14 illustrate looping

4-5 Karnaugh Map Method

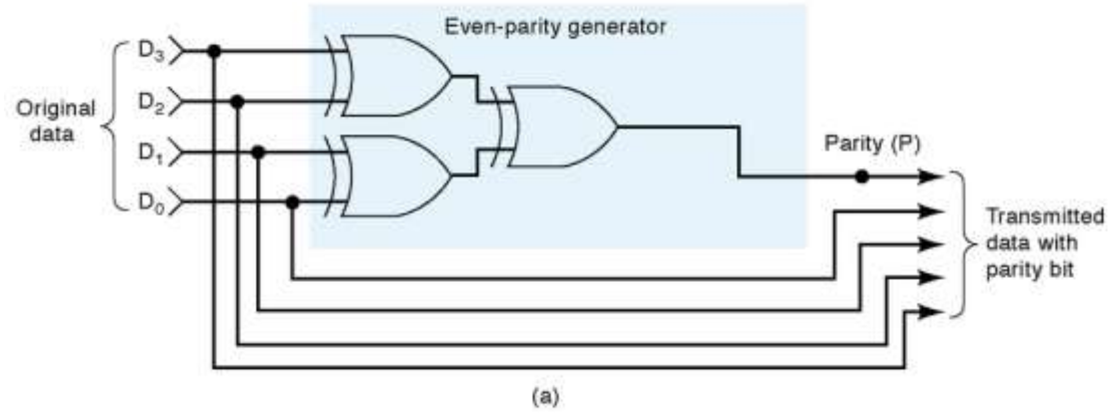
- Complete K map simplification process:
  - Construct the K map, place 1s as indicated in the truth table.
  - Loop 1s that are not adjacent to any other 1s. Isolated 1s.
  - Loop 1s that are adjacent to only one other 1. Loop any pair containing such a 1.
  - Loop 1s in octets even if they have already been looped.
  - Loop quads that have one or more 1s not already looped.
  - Loop any pairs necessary to include 1^{st} not already looped.
  - Form the OR sum of terms generated by each loop.

4-6 Exclusive OR and Exclusive NOR Circuits

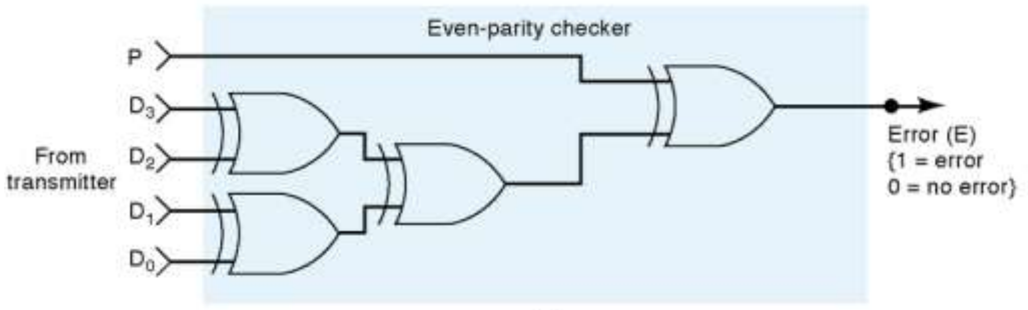
- The exclusive OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.
- The exclusive NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.
- XOR and XNOR outputs are opposite.

4-7 Parity Generator and Checker

- XOR and XNOR gates are useful in circuits for parity generation and checking.



(a)



(b)

4-8 Enable/Disable Circuits

- A circuit is enabled when it allows the passage of an input signal to the output.
- A circuit is disabled when it prevents the passage of an input signal to the output.
- Situations requiring enable/disable circuits occur frequently in digital circuit design.

4-9 Basic Characteristics of Digital ICs

- IC “chips” consist of resistors, diodes, and transistors fabricated on a piece of semiconductor material called a substrate.
- Digital ICs may be categorized according to the number of logic gates on the substrate:
  - SSI – less than 12
  - MSI – 12 to 99
  - LSI – 100 to 9999
  - VLSI – 10,000 to 99,999
  - ULSI – 100,000 to 999,999
  - GSI – 1,000,000 or more

4-9 Basic Characteristics of Digital ICs

- The first package we will examine is the dual in line package (DIP). The pin numbering scheme is described in figure 4-29.
- PLDs have a different numbering scheme that is also described in figure 4-29

4-9 Basic Characteristics of Digital ICs

- ICs are also categorized by the type of components used in their circuits.
  - Bipolar ICs use NPN and PNP transistors
  - Unipolar ICs use FET transistors.
- The transistor-transistor logic (TTL) and the complementary metal-oxide semiconductor (CMOS) families will both be examined.

4-9 Basic Characteristics of Digital ICs

- The TTL family consists of subfamilies as listed in table 4-6.
  - The 74 series devices are all part of the standard TTL series.
  - The 74LS series devices are all part of the low power Schottky TTL series.
  - The differences between devices is limited to electrical characteristics like power dissipation and switching speed. The pin layout and logic operations are the same.
  - The 7404, 74S04, 74LS04, and 74ALS04 are all hex (six to a chip) inverters.

4-9 Basic Characteristics of Digital ICs

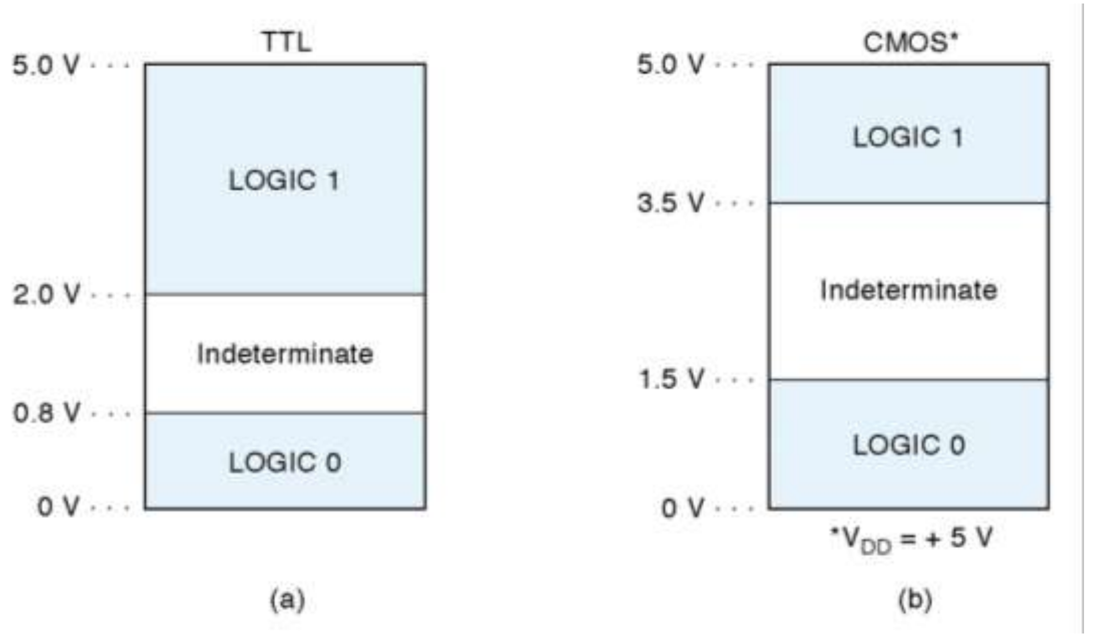
- The CMOS family consists of several series, some of which are shown in table 4-7.
- CMOS devices perform the same function as, but are not necessarily pin for pin compatible with TTL devices.
- Characteristics of TTL and CMOS devices will be explored in more detail in chapter 8.

4-9 Basic Characteristics of Digital ICs

- Power (referred to as V_{CC}) and ground connections are required for chip operation.
- V_{CC} for TTL devices is normally +5 V.
- V_{DD} for CMOS devices can be from +3 to +18 V.
- Logic levels for TTL and CMOS devices are shown in figure 4-31.

4-9 Basic Characteristics of Digital ICs

- Voltages that fall in the *indeterminate* range will provide unpredictable results and should be avoided.



4-9 Basic Characteristics of Digital ICs

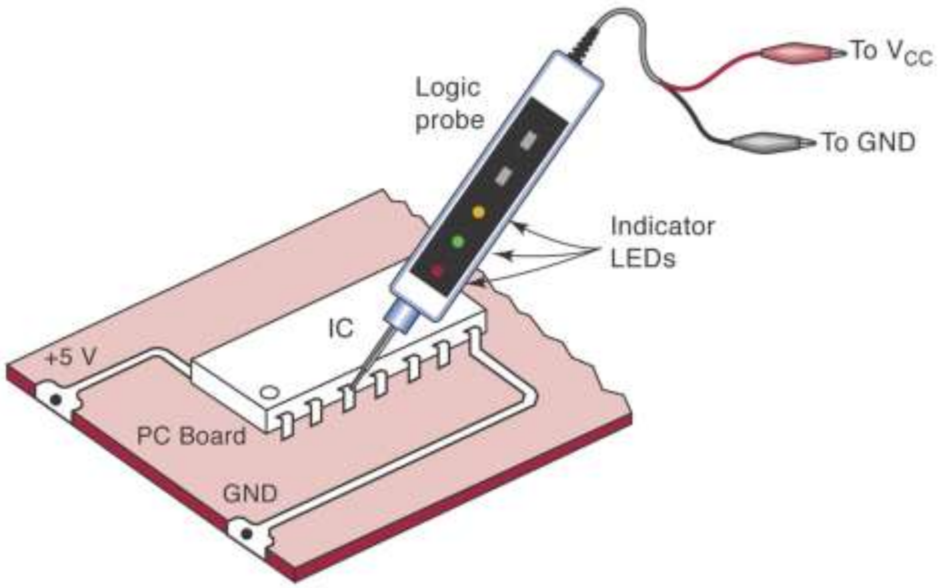
- Inputs that are not connected are said to be floating. The consequences of floating inputs differ for TTL and CMOS.
  - Floating TTL input acts like a logic 1. The voltage measurement may appear in the indeterminate range, but the device will behave as if there is a 1 on the floating input.
  - Floating CMOS inputs can cause overheating and damage to the device. Some ICs have protection circuits built in, but the best practice is to tie all unused inputs either high or low.

4-10 Troubleshooting Digital Systems

- 3 basic steps
  - Fault detection, determine operation to expected operation.
  - Fault isolation, test and measure to isolate the fault.
  - Fault correction, repair the fault.
- Good troubleshooting skills come through experience in actual hands-on troubleshooting.
- The basic troubleshooting tools used here will be: the logic probe, oscilloscope, and logic pulser.
- The most important tool is the technician’s brain.

4-10 Troubleshooting Digital Systems

- The logic probe will indicate the presence or absence of a signal when touched to a pin as indicated below.



LEDs			
Red	Green	Yellow	Logic Condition
OFF	ON	OFF	LOW
ON	OFF	OFF	HIGH
OFF	OFF	OFF	INDETERMINATE*
X	X	FLASHING	PULSING

* Includes open or floating condition

4-11 Internal Digital IC Faults

- Most common internal failures:
  - Malfunction in the internal circuitry.
  - Inputs or outputs shorted to ground or V_{CC}
  - Inputs or outputs open-circuited
  - Short between two pins (other than ground or V_{CC})

4-11 Internal Digital IC Faults

- Malfunction in internal circuitry
  - Outputs do not respond properly to inputs. Outputs are unpredictable.
- Input internally shorted to ground or supply
  - The input will be stuck in LOW or HIGH state.
- Output internally shorted to ground or supply
  - Output will be stuck in LOW or HIGH state.
- Open-circuited input or output
  - Floating input in a TTL device will result in a HIGH output. Floating input in a CMOS device will result in erratic or possibly destructive output.
  - An open output will result in a floating indication.
- Short between two pins
  - The signal at those pins will always be identical.

4-12 External Faults

- Open signal lines – signal is prevented from moving between points. Some causes:
  - Broken wire
  - Poor connections (solder or wire-wrap)
  - Cut or crack on PC board trace
  - Bent or broken IC pins.
  - Faulty IC socket
- Detect visually and verify with an ohmmeter.

4-12 External Faults

- Shorted signal lines – the same signal will appear on two or more pins. V_{CC} or ground may also be shorted. Some causes:
  - Sloppy wiring
  - Solder bridges
  - Incomplete etching
- Detect visually and verify with an ohmmeter.

4-12 External Faults

- Faulty power supply – ICs will not operate or will operate erratically.
  - May lose regulation due to an internal fault or because circuits are drawing too much current.
  - Always verify that power supplies are providing the specified range of voltages and are properly grounded.
  - Use an oscilloscope to verify that AC signals are not present.

4-12 External Faults

- Output loading – caused by connecting too many inputs to the output of an IC.
  - Causes output voltage to fall into the indeterminate range.
  - This is called *loading* the output.
  - Usually a result of poor design or bad connection.

4-13 Troubleshooting Case Study

- Example 4-28 illustrates the process involved in troubleshooting a fairly simple circuit.
- The reasoning process can be applied to more complex digital circuits.
- Take the time to carefully study the example and use the opportunities at the end of the chapter to practice and develop troubleshooting skills.

4-14 Programmable Logic Devices

- PLDs allow the design process to be automated.
- Designers identify inputs, outputs, and logical relationships.
- PLDs are electronically configured to form the defined logic circuits.

4-14 Programmable Logic Devices

- PLD ICs can be programmed out of system or in system.
  - For out of system programming the PLD is placed in a fixture called a **programmer** which is connected to a PC running software that translates and loads the information.
  - In system programming is done by connecting directly to four designated "portal" pins while the IC remains in the system. An interface cable connects the PLD to a PC running the software that loads the device.

4-14 Programmable Logic Devices

- Logic circuits can be described using schematic diagrams, logic equations, truth tables, and HDL.
- PLD development software can convert any of these descriptions into 1s and 0s and loaded into the PLD.
- Altera MAX+PLUS II is an example of development software that allows the user to describe circuits using graphic design files and timing diagrams.

4-14 Programmable Logic Devices

- Hierarchical design – small logic circuits are defined and combined with other circuits to form a large section of a project. Large sections can be combined and connected for form a system.
- Top-down design requires the definition of sub sections that will make up the system, and definition of the individual circuits that will make up each sub section.
- Each level of the hierarchy can be designed and tested individually.

4-14 Programmable Logic Devices

- A system is built from the bottom up.
  - Each block is described by a design file.
  - The designed block is tested
  - After testing it is compiled using development software.
  - The compiled block is tested using a simulator for verify correct operation.
  - A PLD is programmed to verify correct operation.
- The flowchart in figure 4-48 summarizes the design process for each block of the system.

4-15 Representing Data in HDL

- Binary, hexadecimal, and decimal values are represented in HDL as shown in table 4-8.
- In order to describe a port with more than one data bit we assign a name and the number of bits. This is called a bit array or bit vector.

4-15 Representing Data in HDL

- AHDL syntax – a name for the bit vector is followed by the range of index designations enclosed in square brackets. This would appear in the SUBDESIGN section.
  - To declare an 8 bit input port called p1:

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

  - 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 be connected by specifying the bits inside the braces.

4-15 Representing Data in HDL

- VHDL syntax – a name for the bit vector is followed by the mode, the type, and the range enclosed in parenthesis. This would appear in the ENTITY section.
  - To declare an 8 bit called p1:

    ```
    PORT (p1          :IN BIT_VECTOR (7 DOWNTO 0);
    ```
  - To assign the 8 bit port p1 to a signal named temp:

    ```
    SIGNAL  temp      :BIT_VECTOR (7 DOWNTO 0);
    BEGIN
              temp <=p1;
    END;
    ```
  - Since no elements are specified, all the bits will be connected. Individual bits could be connected by placing bit number inside parentheses.

4-15 Representing Data in HDL

- VHDL offers some standardized data types in libraries.
- Libraries contain collections of VHDL code that can be used to avoid reinventing the wheel.
- Many convenient functions such as standard TTL device descriptions are contained in macrofunctions.

4-16 Truth Tables Using HDL

- Circuits can be designed directly from truth tables in HDL.
- Figures 4-50 and 4-51 describe the syntax for AHDL and VHDL truth tables respectively.

4-17 Decision Control Structures in HDL

- 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.

4-17 Decision Control Structures in HDL

- 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;
```

4-17 Decision Control Structures in HDL

- The IF/THEN/ELSE in VHDL:

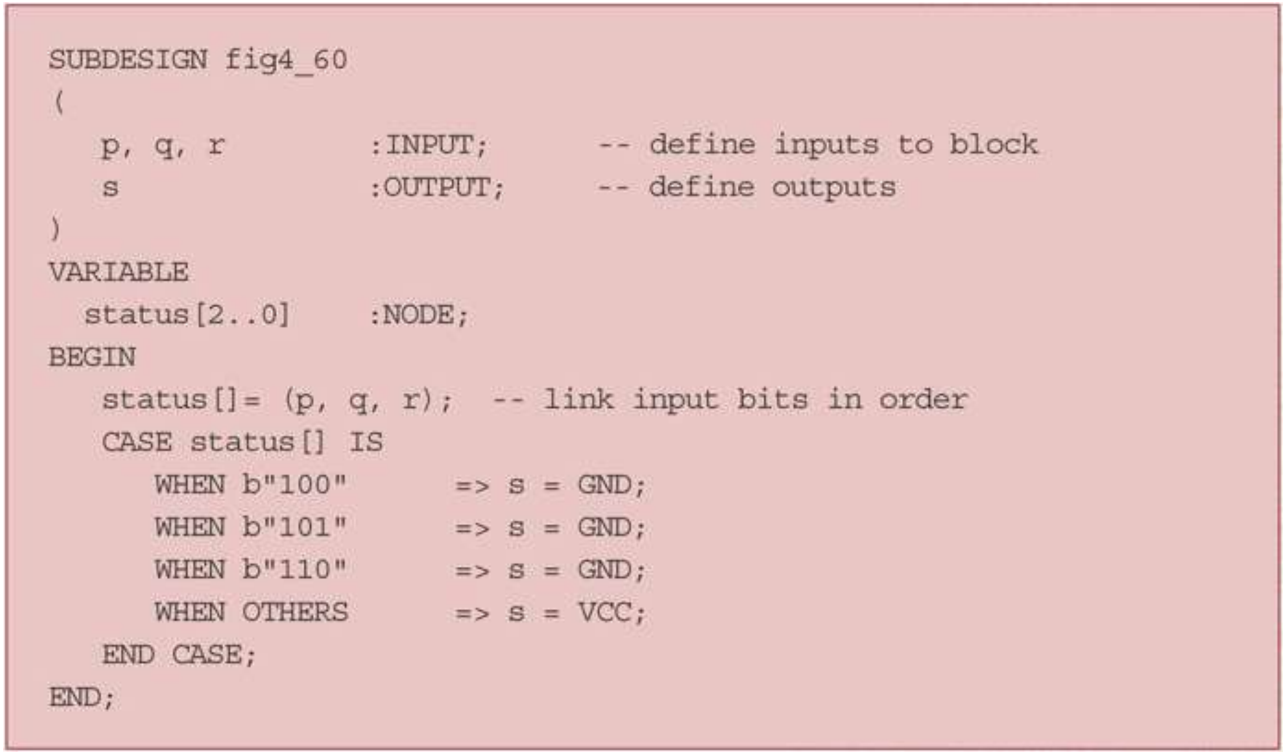
```
ENTITY fig4_55 IS
PORT( digital_value :IN INTEGER RANGE 0 TO 15; -- 4-bit input
      z             :OUT BIT);
END fig4_55;

ARCHITECTURE truth OF fig4_55 IS

BEGIN
   PROCESS (digital_value)
      BEGIN
         IF (digital_value > 6) THEN
            z <= '1';
         ELSE
            z <= '0';
      END IF;
END PROCESS ;
END truth;
```

4-17 Decision Control Structures in HDL

- The CASE construct determines the value of an expression or object and then goes through a list of values (cases) to determine what action to take.
- This is different than the IF/ELSEIF because there is only one action or match for a case statement.

4-17 Decision Control Structures in HDL

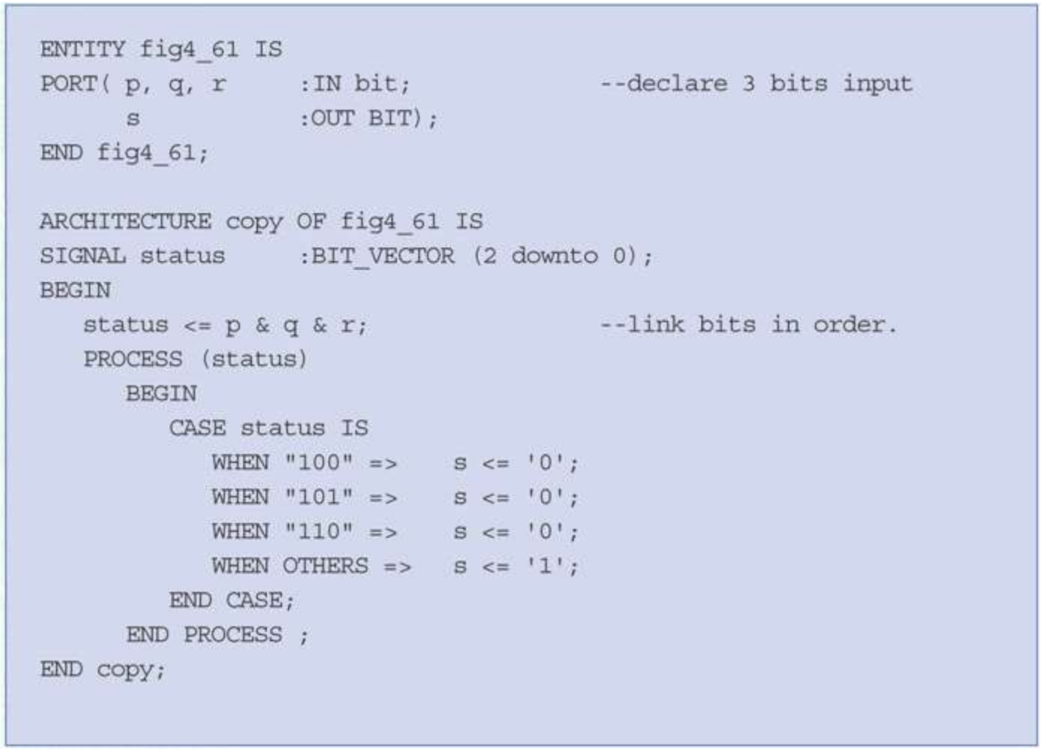
- The case construct in AHDL:



```
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;
```

4-17 Decision Control Structures in HDL

- The case construct in VHDL:



```
ENTITY fig4_61 IS
PORT( p, q, r       :IN bit;                --declare 3 bits input
      s             :OUT BIT);
END fig4_61;

ARCHITECTURE copy OF fig4_61 IS
SIGNAL status       :BIT_VECTOR (2 downto 0);
BEGIN
   status <= p & q & r;                     --link bits in order.
   PROCESS (status)
      BEGIN
         CASE status IS
            WHEN "100" =>    s <= '0';
            WHEN "101" =>    s <= '0';
            WHEN "110" =>    s <= '0';
            WHEN OTHERS =>   s <= '1';
         END CASE;
      END PROCESS ;
END copy;
```