



UNIVERSITÀ
DEGLI STUDI DI TRIESTE

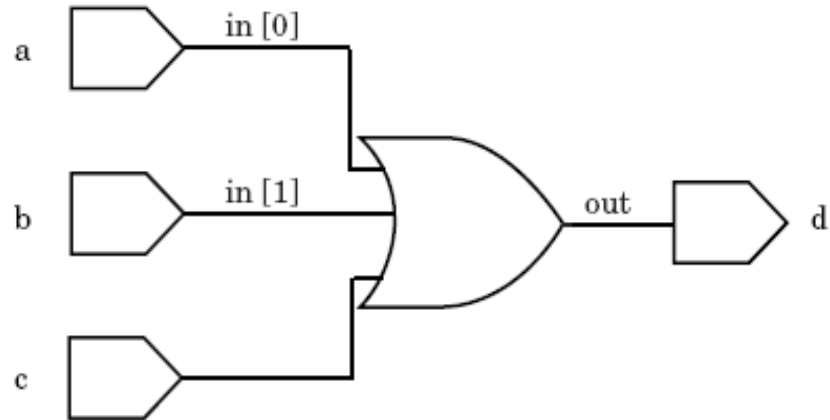


14 - VHDL Synthesis

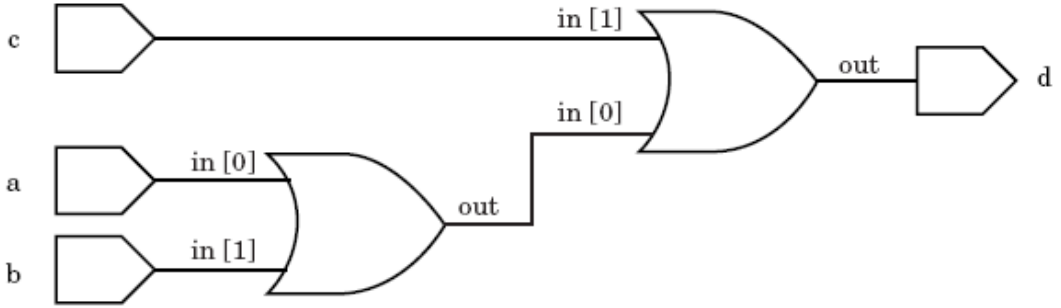
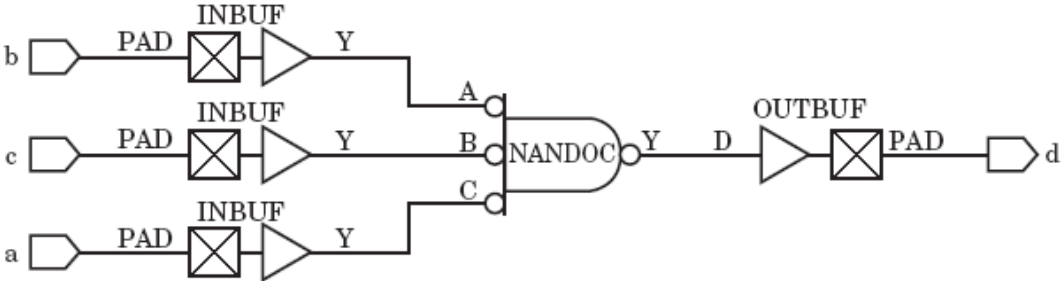
A.Carini – Progettazione di sistemi elettronici

Concurrent assignment of a gate

```
LIBRARY IEEE;  
USE IEEE.std_logic_1164.ALL;  
ENTITY or3 IS  
    PORT (a, b, c : IN std_logic;  
          d : OUT std_logic);  
END or3;  
  
ARCHITECTURE synth OF or3 IS  
BEGIN  
    d <= a OR b OR c;  
END synth;
```



Concurrent assignment of a gate



IF control flow statement

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY alarm_cntrl IS
    PORT( smoke, front_door, back_door, side_door,
          alarm_disable, main_disable,
          water_detect : IN std_logic;
          fire_alarm, burg_alarm,
          water_alarm : OUT std_logic);
END alarm_cntrl;

ARCHITECTURE synth OF alarm_cntrl IS
BEGIN
    PROCESS(smoke, front_door, back_door, side_door,
            alarm_disable, main_disable,
            water_detect)

    BEGIN
```

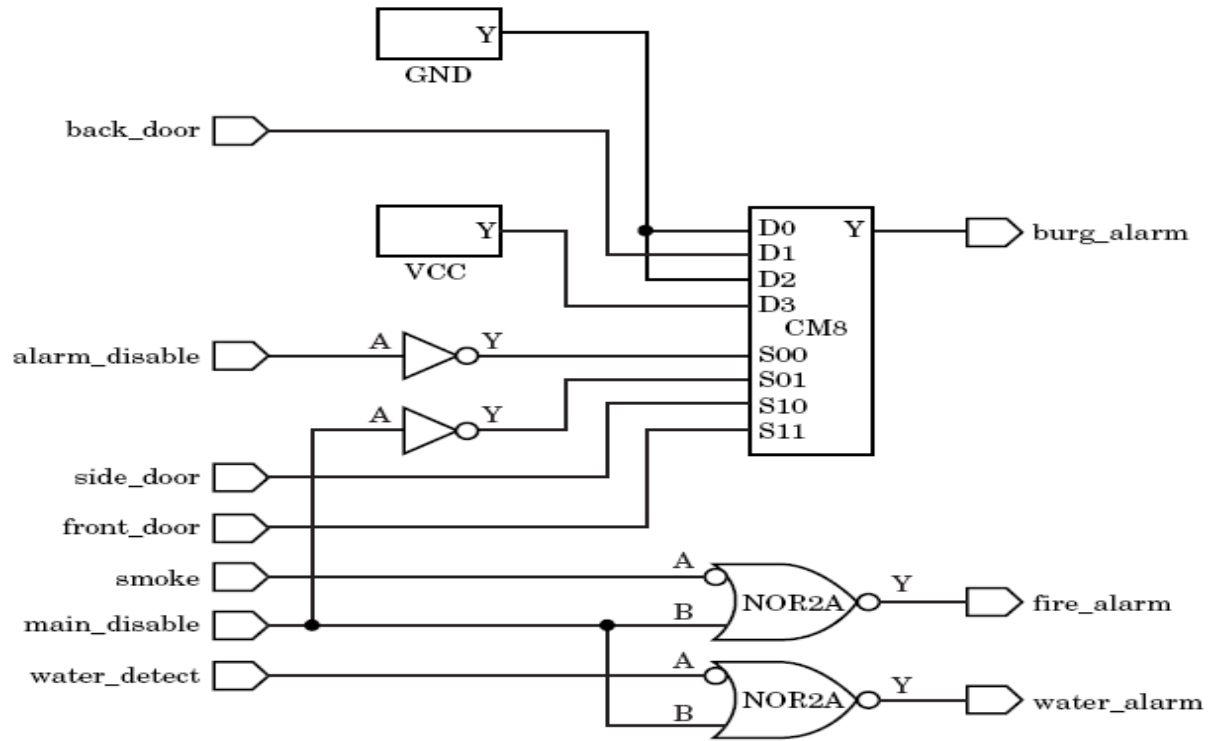
IF control flow statement

```
IF ((smoke = '1') AND (main_disable = '0')) THEN
    fire_alarm <= '1';
ELSE
    fire_alarm <= '0';
END IF;

IF (((front_door = '1') OR (back_door = '1') OR
    (side_door = '1')) AND
    ((alarm_disable = '0') AND (main_disable =
    '0')))) THEN
    burg_alarm <= '1';
ELSE
    burg_alarm <= '0';
END IF;

IF ((water_detect = '1') AND (main_disable = '0'))
    THEN
    water_alarm <= '1';
ELSE
    water_alarm <= '0';
END IF;
END PROCESS;
END synth;
```

IF control flow statement



CASE control flow

```
PACKAGE comp_pack IS
  TYPE bit8 is range 0 TO 255;
  TYPE t_comp IS (greater_than, less_than, equal,
                 not_equal, grt_equal, less_equal);
END comp_pack;

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE WORK.comp_pack.ALL;
ENTITY compare IS
  PORT( a, b : IN bit8;
        ctrl : IN t_comp;
        gt, lt, eq, neq, gte, lte : OUT std_logic);
END compare;

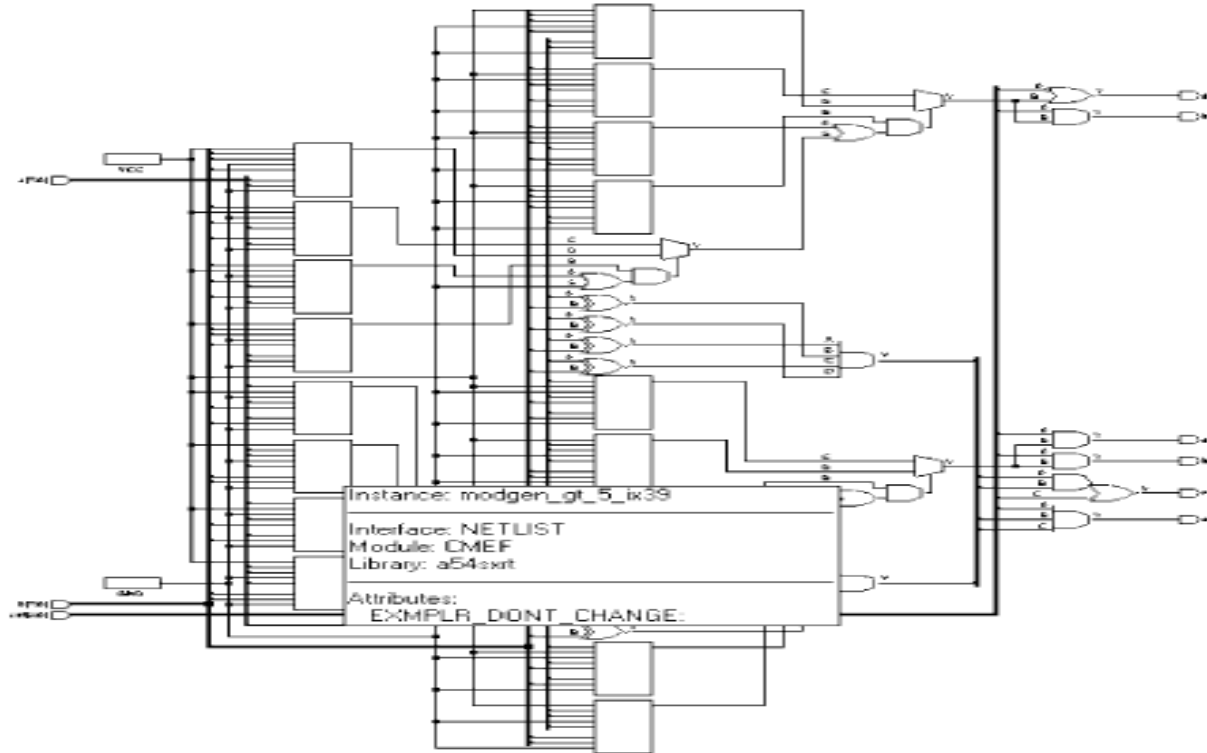
ARCHITECTURE synth OF compare IS
BEGIN

  PROCESS(a, b, ctrl)
  BEGIN
    gt <= '0'; lt <= '0'; eq <= '0'; neq <= '0'; gte <=
      '0'; lte <= '0';
```

CASE control flow

```
CASE ctrl IS
  WHEN greater_than =>
    IF (a > b) THEN
      gt <= '1';
    END IF;
  WHEN less_than =>
    IF (a < b) THEN
      lt <= '1';
    END IF;
  WHEN equal =>
    IF (a = b) THEN
      eq <= '1';
    END IF;
  WHEN not_equal =>
    IF (a /= b) THEN
      neq <= '1';
    END IF;
  WHEN grt_equal =>
    IF (a >= b) THEN
      gte <= '1';
    END IF;
  WHEN less_equal =>
    IF (a > b) THEN
      lte <= '1';
    END IF;
END CASE;
END PROCESS;
END synth;
```


CASE control flow



D flip-flop

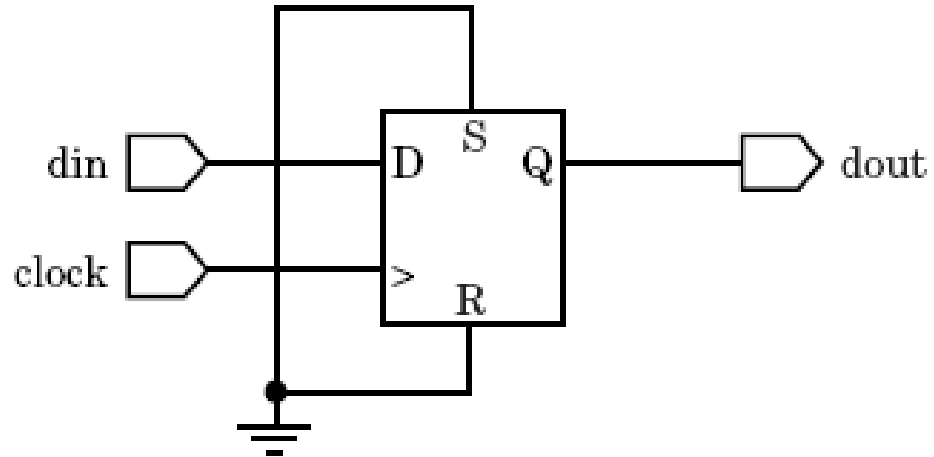
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY dff IS
    PORT( clock, din : IN std_logic;
          dout : OUT std_logic);
END dff;

ARCHITECTURE synth OF dff IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL ((clock'EVENT) AND (clock = '1'));

        dout <= din;

    END PROCESS;
END synth;
```

D flip-flop

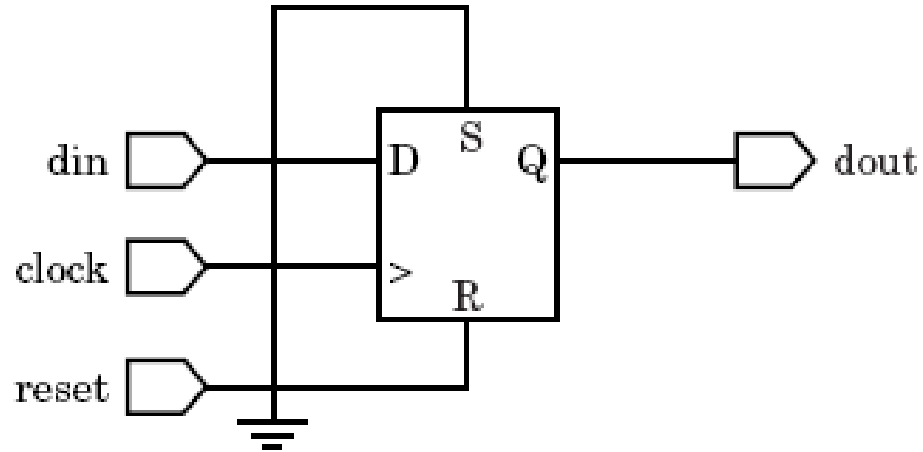


Asynchronous reset

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY dff_asynch IS
    PORT( clock, reset, din : IN std_logic;
          dout : OUT std_logic);
END dff_asynch;

ARCHITECTURE synth OF dff_asynch IS
BEGIN
    PROCESS(reset, clock)
    BEGIN
        IF (reset = '1') THEN
            dout <= '0';
        ELSEIF (clock'EVENT) AND (clock = '1') THEN
            dout <= din;
        END IF;
    END PROCESS;
END synth;
```

Asynchronous reset



Asynchronous Preset and Clear

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY dff_pc IS
    PORT( preset, clear, clock, din : IN std_logic;
          dout : OUT std_logic);
END dff_pc;

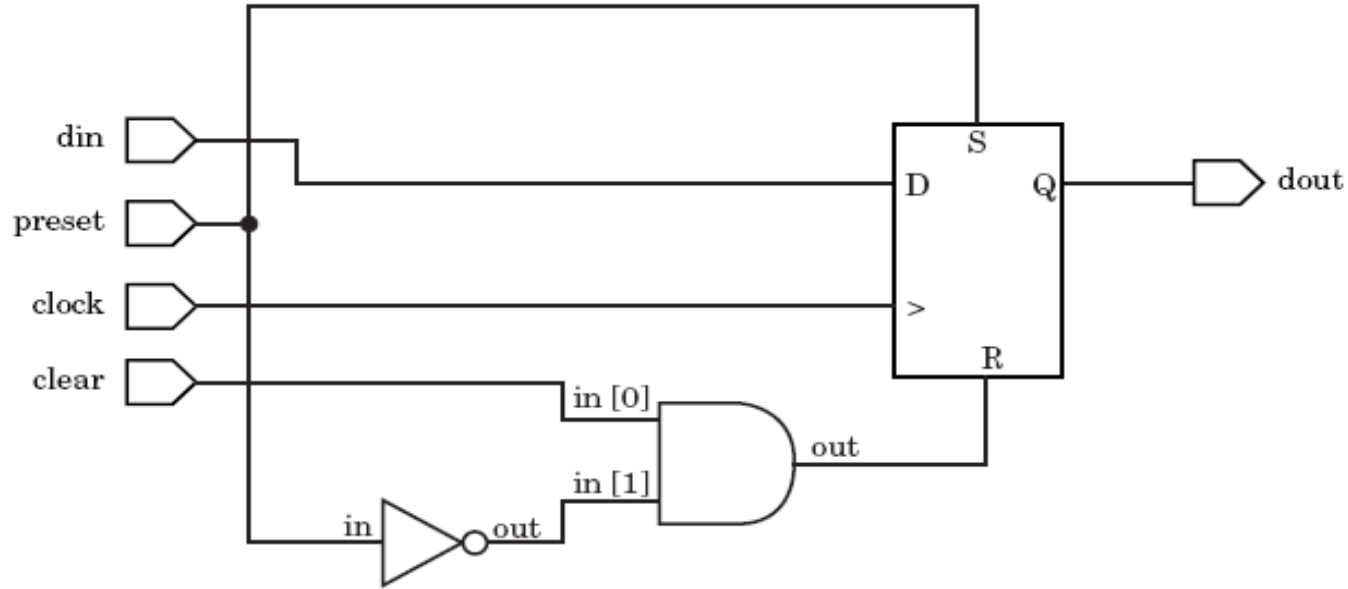
ARCHITECTURE synth OF dff_pc IS
BEGIN
PROCESS(preset, clear, clock)
BEGIN
    IF (preset = '1') THEN
        dout <= '1';

    ELSEIF (clear = '1') THEN
        dout <= '0';

    ELSEIF (clock'EVENT) AND (clock = '1') THEN
        dout <= din;

    END IF;
END PROCESS;
END synth;
```

Asynchronous Preset and Clear



4 bit counter

```
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
PACKAGE count_types IS
    SUBTYPE bit4 IS std_logic_vector(3 DOWNTO 0);
END count_types;

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
USE WORK.count_types.ALL;
ENTITY count IS
    PORT(clock, load, clear : IN std_logic;
          din : IN bit4;
          dout : INOUT bit4);
END count;
```


4 bit counter

```
ARCHITECTURE synth OF count IS
  SIGNAL count_val : bit4;
BEGIN
  PROCESS(load, clear, din, dout)
  BEGIN
    IF (load = '1') THEN
      count_val <= din;
    ELSEIF (clear = '1') THEN
      count_val <= "0000";
    ELSE
      count_val <= dout + "0001";
    END IF;
  END PROCESS;

  PROCESS
  BEGIN
    WAIT UNTIL clock'EVENT and clock = '1';

    dout <= count_val;
  END PROCESS;
END synth;
```

4 bit shifter

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
PACKAGE shift_types IS
    SUBTYPE bit4 IS std_logic_vector(3 downto 0);
END shift_types;
```

```
USE WORK.shift_types.ALL;
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY shifter IS
    PORT( din : IN bit4;
          clk, load, left_right : IN std_logic;
          dout : INOUT bit4);
END shifter;
```

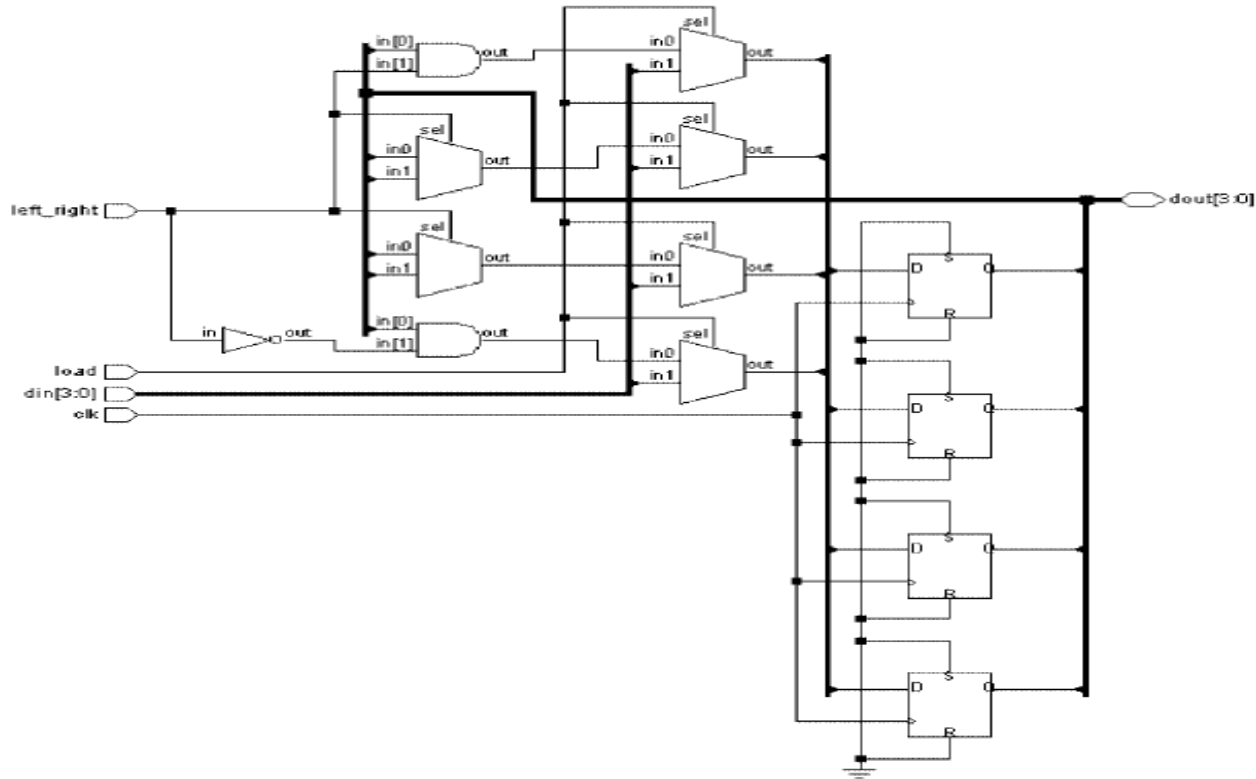
4 bit shifter

```
ARCHITECTURE synth OF shifter IS
  SIGNAL shift_val : bit4;
BEGIN
  nxt: PROCESS(load, left_right, din, dout)
  BEGIN
    IF (load = '1') THEN
      shift_val <= din;
    ELSEIF (left_right = '0') THEN
      shift_val(2 downto 0) <= dout(3 downto 1);
      shift_val(3) <= '0';
    ELSE
      shift_val(3 downto 1) <= dout(2 downto 0);
      shift_val(0) <= '0';
    END IF;
  END PROCESS;

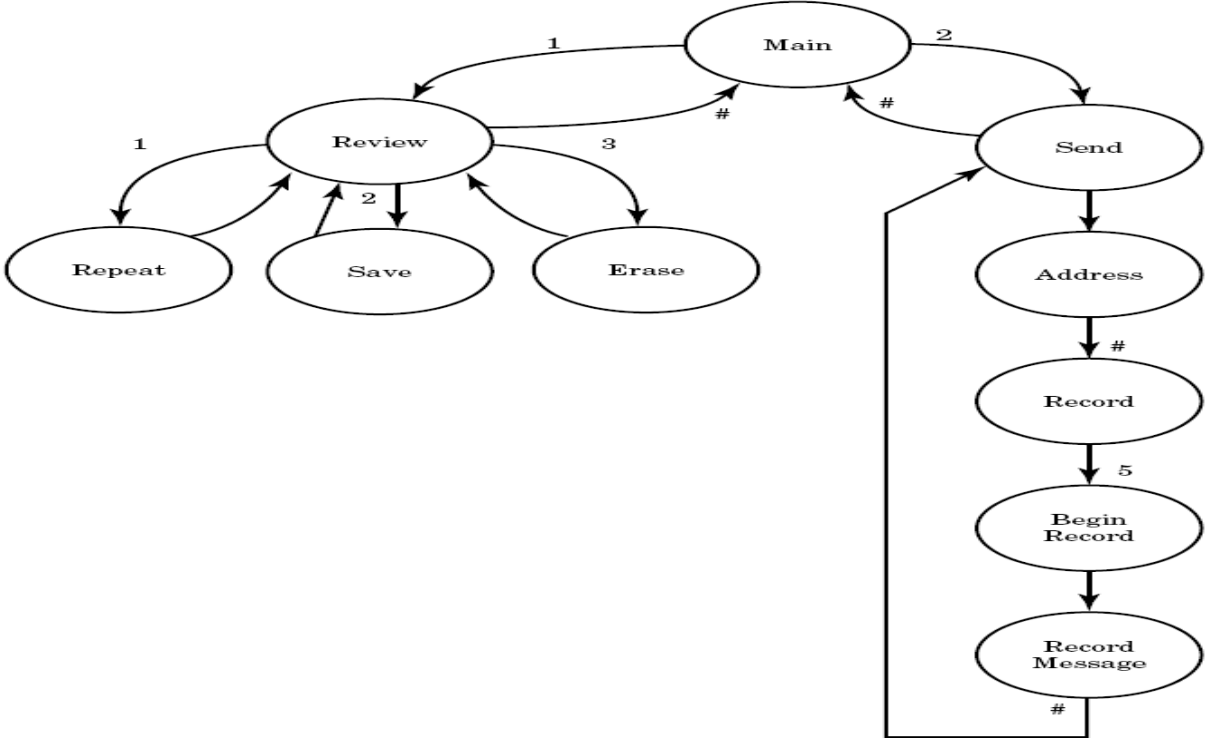
  current: PROCESS
  BEGIN
    WAIT UNTIL clk'EVENT AND clk = '1';

    dout <= shift_val;
  END PROCESS;
END synth;
```

4 bit shifter



State-machine example



State-machine example

```
PACKAGE vm_pack IS
  TYPE t_vm_state IS (main_st, review_st, repeat_st,
                      save_st,
                      erase_st, send_st,
                      address_st, record_st,
                      begin_rec_st, message_st);
  TYPE t_key IS ('0','1','2','3','4','5','6','7','8','9',
                '*',' ');
END vm_pack;

USE WORK.vm_pack.ALL;
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY control IS
  PORT( clk : in std_logic;
        key : in t_key;
        play, recrd, erase, save, address
        : out std_logic);
END control;

ARCHITECTURE synth OF control IS
  SIGNAL next_state, current_state :
    t_vm_state;
BEGIN
```

State-machine example

```
PROCESS(current_state, key)
BEGIN
  play <= '0';
  save <= '0';
  erase <= '0';
  recrd <= '0';
  address <= '0';

CASE current_state IS
  WHEN main_st =>
    IF (key = '1') THEN
      next_state <= review_st;
    ELSEIF (key = '2') THEN
      next_state <= send_st;
    ELSE
      next_state <= main_st;
    END IF;

  WHEN review_st =>
    IF (key = '1') THEN
      next_state <= repeat_st;
    ELSEIF (key = '2') THEN
      next_state <= save_st;
    ELSEIF (key = '3') THEN
      next_state <= erase_st;
    ELSEIF (key = '#') THEN
      next_state <= main_st;
    ELSE
      next_state <= review_st;
    END IF;
```

State-machine example

```
WHEN repeat_st =>
  play <= '1';
  next_state <= review_st;

WHEN save_st =>
  save <= '1';
  next_state <= review_st;

WHEN erase_st =>
  erase <= '1';
  next_state <= review_st;

WHEN send_st =>
  next_state <= address_st;

WHEN address_st =>
  address <= '1';
  IF (key = '#') THEN
    next_state <= record_st;
  ELSE
    next_state <= address_st;
  END IF;

WHEN record_st =>
  IF (key = '5') THEN
    next_state <= begin_rec_st;
  ELSE
    next_state <= record_st;
  END IF;
```


State-machine example

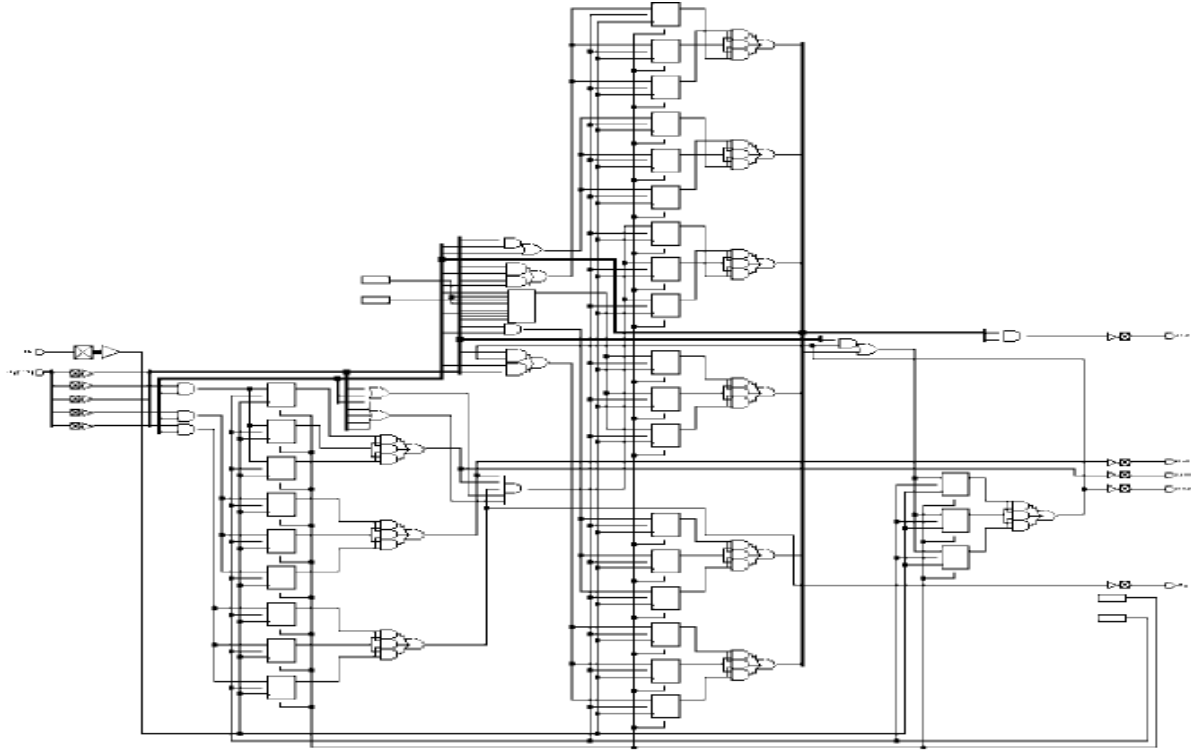
```
WHEN begin_rec_st =>
  recrd <= '1';
  next_state <= message_st;

WHEN message_st =>
  recrd <= '1';
  IF (key = '#') THEN
    next_state <= send_st;
  ELSE
    next_state <= message_st;
  END IF;
END CASE;
END PROCESS;

PROCESS
BEGIN
  WAIT UNTIL clk = '1' AND clk'EVENT;

  current_state <= next_state;
END PROCESS;
END synth;
```

State-machine example



See:

- Douglas L. Perry, «VHDL programming by example» McGraw Hill,
 - Chapter 10