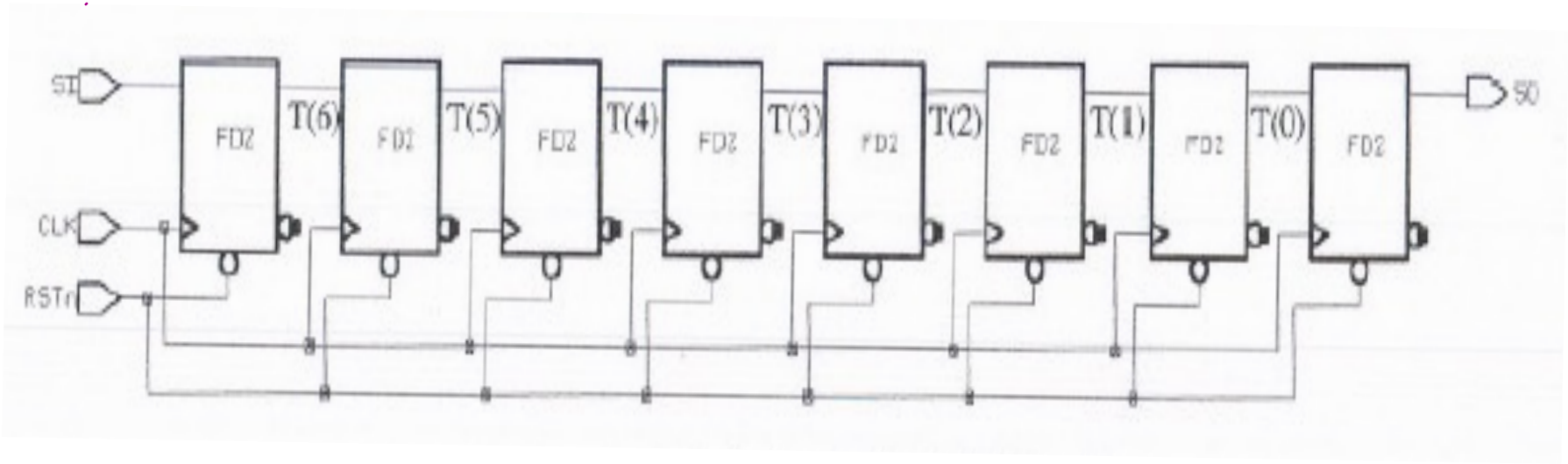
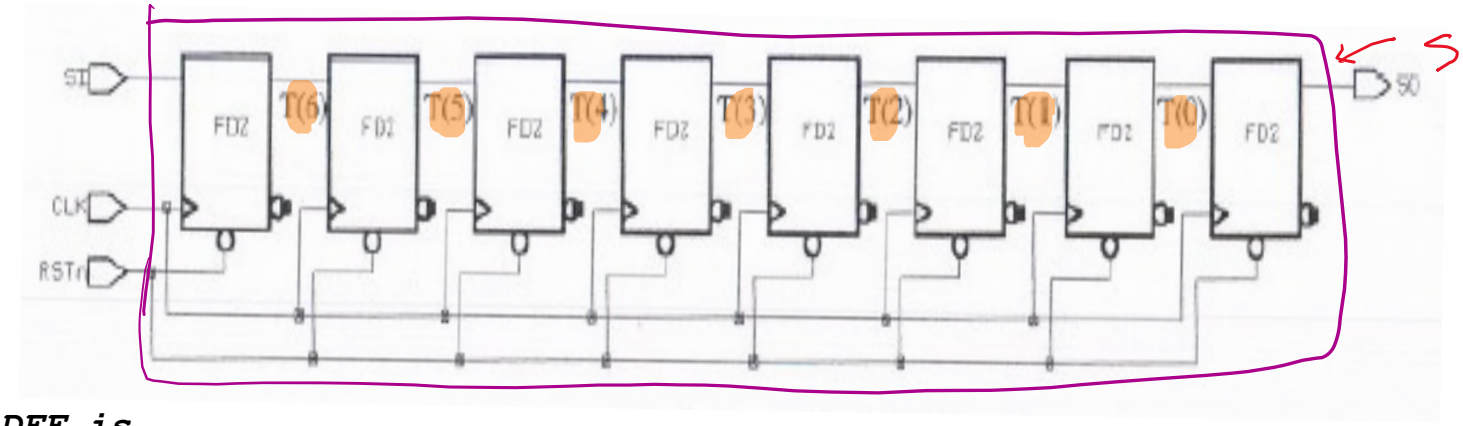
Component Instantiation using GENERATE

Component Instantiation using GENERATE

- ***8-b shifter register***
- ***24-b shifter register with 3 8-b shifter register***
- ***GENERATE statement***
- ***GENERATE examples***

Component Instantiation: 8-b shifter register



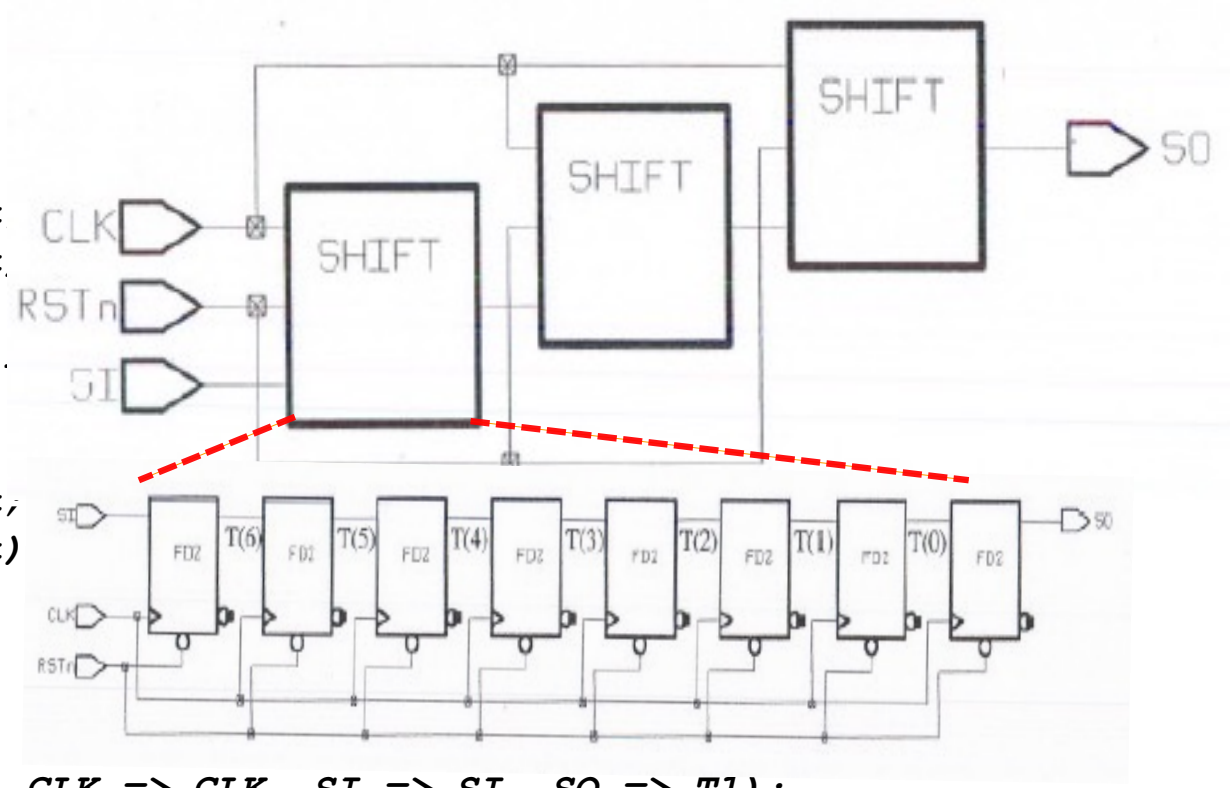



```
entity DFF is
   port (
      RSTn, CLK, D : in  bit;
      Q            : out bit);
end DFF;
architecture RTL of DFF is
begin
   process (RSTn, CLK)
   begin
      if (RSTn = '0') then
         Q <= '0';
      elsif (CLK'event and CLK = '1')
then
         Q <= D;
      end if;
   end process;
end RTL;
-------------------------------------
entity SHIFT is
   port (
      RSTn, CLK, SI : in  bit;
      SO            : out bit);
end SHIFT;
```

```
architecture RTL1 of SHIFT is
   component DFF
   port (
      RSTn, CLK, D : in  bit;
      Q            : out bit);
   end component;
   signal T : bit_vector(6 downto 0);
begin
   bit7 : DFF
      port map (RSTn => RSTn, CLK => CLK, D => SI,
Q => T(6));
   bit6 : DFF
      port map (RSTn, CLK, T(6), T(5));
   bit5 : DFF
      port map (RSTn, CLK, T(5), T(4));
   bit4 : DFF
      port map (CLK => CLK, RSTn => RSTn, D =>
T(4), Q => T(3));
   bit3 : DFF
      port map (RSTn, CLK, T(3), T(2));
   bit2 : DFF
      port map (RSTn, CLK, T(2), T(1));
   bit1 : DFF
      port map (RSTn, CLK, T(1), T(0));
   bit0 : DFF
      port map (RSTn, CLK, T(0), SO);
end RTL1;
```

Component Instantiation: 24-b shifter Register

```
entity SHIFT24 is
   port (
      RSTn, CLK, SI : in  bit
      SO            : out bit
end SHIFT24;
architecture RTL5 of SHIFT24 .
   component SHIFT
   port (
      RSTn, CLK, SI : in  bit;
      SO            : out bit)
   end component;
   signal T1, T2 : bit;
begin
   stage2 : SHIFT
      port map (RSTn => RSTn, CLK => CLK, SI => SI, SO => T1);
   stage1 : SHIFT
      port map (RSTn => RSTn, CLK => CLK, SI => T1, SO => T2);
   stage0 : SHIFT
      port map (RSTn => RSTn, CLK => CLK, SI => T2, SO => SO);
end RTL5;
```



Generate Statement

generate statement::=

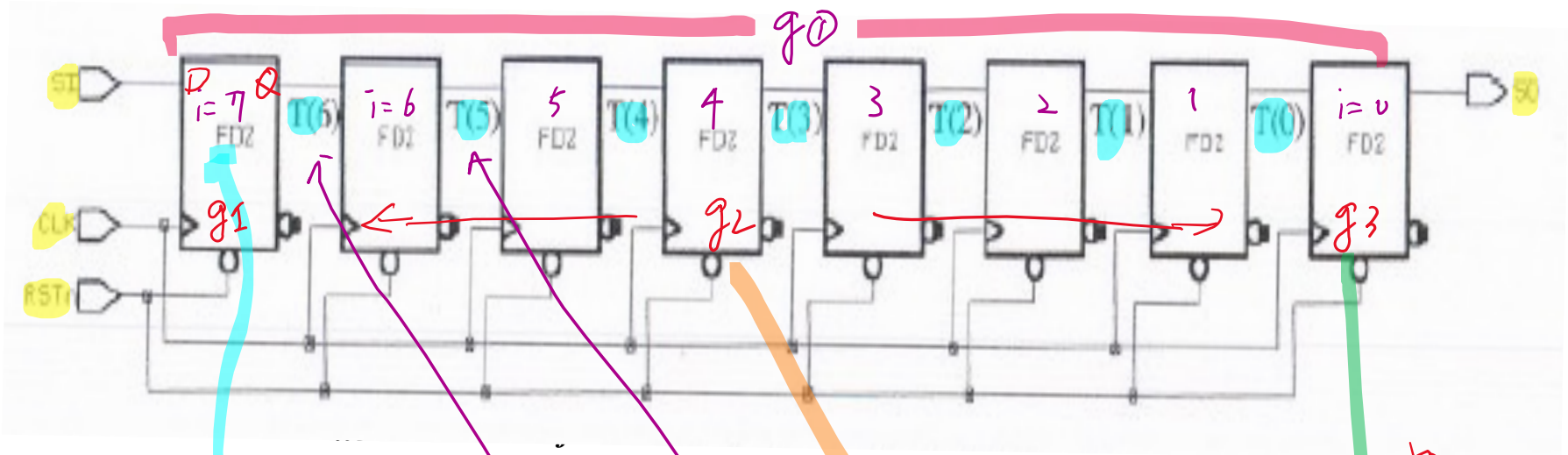
generate_label: for generate_parameter_spec generate | if condition generate

concurrent statements

end generate [generate_label];

optional

- *Generate statement is a concurrent statement.*
- *If statement and for loop statements are sequential statements.*
- *The if condition generate statement does not have else, elsif clause.*
- *A label is required for a generate statement.*




```
component DFF
port (     RSTn, CLK, D : in  bit;
                Q       : out bit);
end component;
signal T : bit_vector(6 downto 0);
begin
g0 : for i in 7 downto 0 generate
  g1 : if (i = 7) generate
    bit7 : DFF
      port map (RSTn => RSTn, CLK => CLK, D => SI, Q => T(6));
  end generate;
  g2 : if (i > 0) and (i < 7) generate
    bitm : DFF
      port map (RSTn, CLK, T(i), T(i-1));
  end generate;
  g3 : if (i = 0) generate
    bit0 : DFF
      port map (RSTn, CLK, T(0), SO);
  end generate;
end generate;
end RTL2;
```

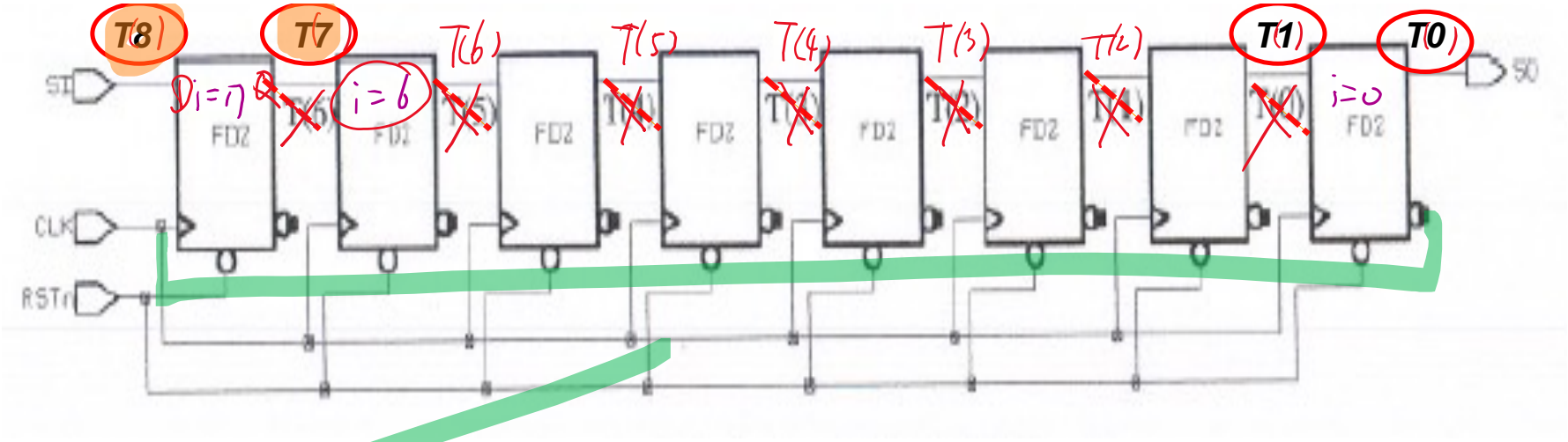
Top Generate label

use for-loop to instantiate DFF 8 times

3 sub-generate

i=6

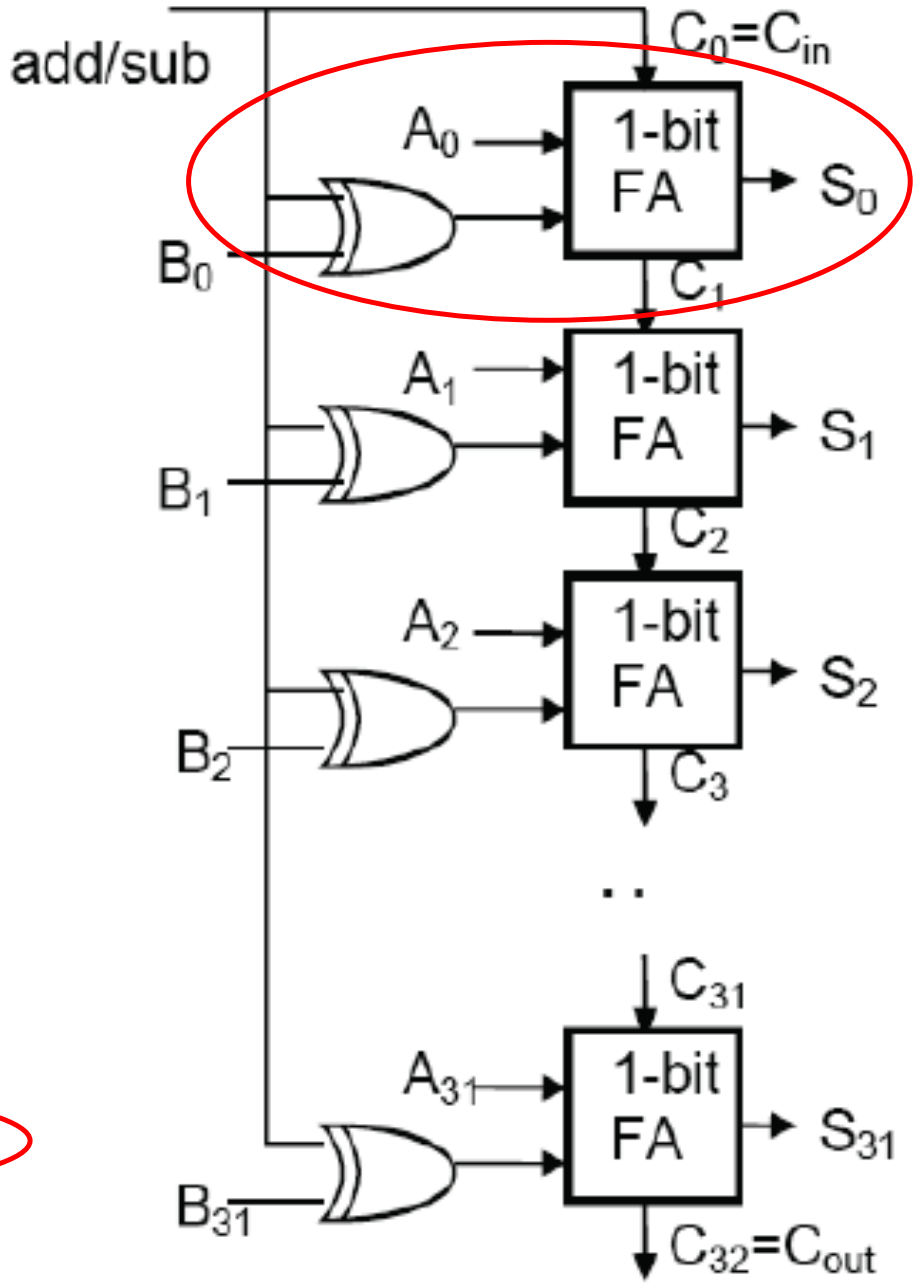
T(6) T(5)



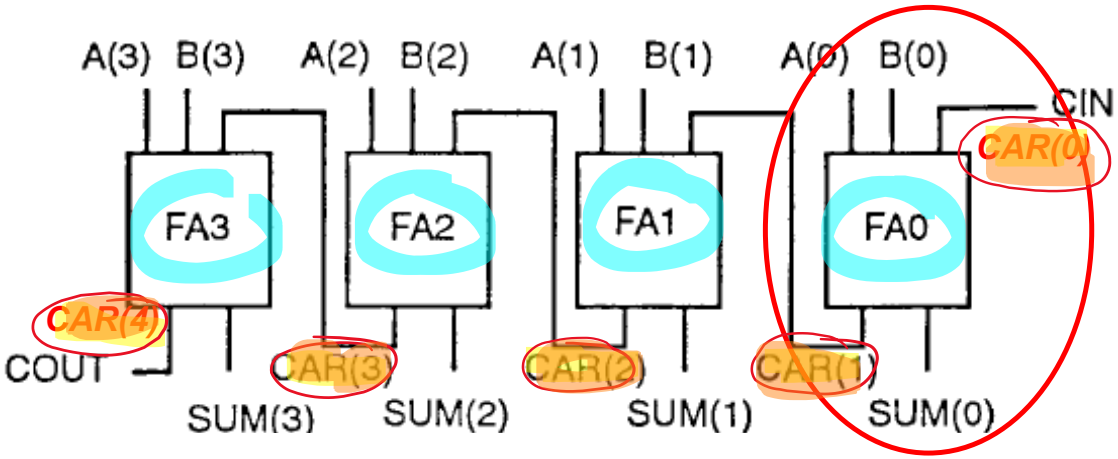

```
architecture RTL3 of SHIFT is
  component DFF
  port (     RSTn, CLK, D : in  bit;
             Q            : out bit);
  end component;

  signal T : bit_vector(8 downto 0);

begin
  T(8) <= SI;
  SO   <= T(0);
  g0 : for i in 7 downto 0 generate
    allbit : DFF
      port map (RSTn => RSTn, CLK => CLK, D => T(i+1), Q => T(i));
  end generate;
end RTL3;
```

New signal declaration
=> One generate statement

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
ENTITY byte_adder IS
PORT(  A,B : IN STD_LOGIC(31 DOWNTO 0);
       CIN : IN STD_LOGIC;
       S : OUT STD_LOGIC(31 DOWNTO 0);
       CO : OUT STD_LOGIC );
END ENTITY;

ARCHITECTURE arch_byt OF byte_adder IS

COMPONENT fa IS
PORT(  a,b,cin : IN STD_LOGIC;
       sum,cout : OUT STD_LOGIC);
END COMPONENT;

COMPONENT xor2 IS
PORT(  a,b : IN STD_LOGIC;
       c : OUT STD_LOGIC);
END COMPONENT;

SIGNAL SIG,CAR : STD_LOGIC(31 DOWNTO 0);

BEGIN
       CAR(0) <= CIN;
       GK: FOR k IN 31 DOWNTO 0 GENERATE
       X1 : xor2 PORT MAP ( CIN, B(k), SIG(k));
       F1 : fa PORT MAP(A(k), SIG(k), CAR(k), S(k), CAR(k+1))
       END GENERATE GK;
       COUT <= CAR(32);
END ARCHITECTURE arch_byt;
```



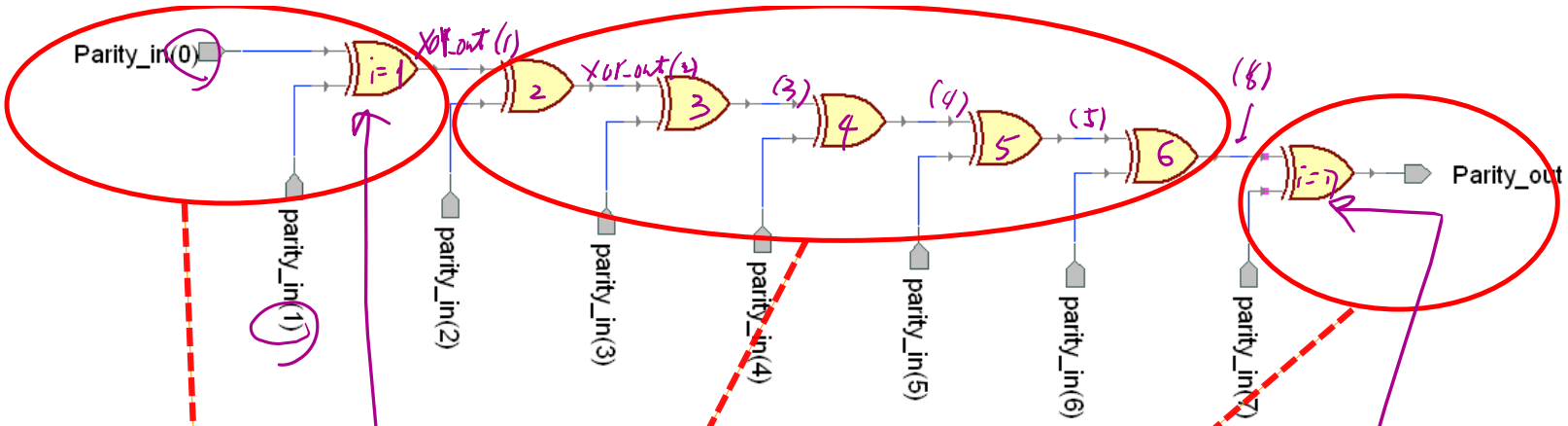
Example:




```
entity FULL_ADD4 is
port (A, B: in BIT_VECTOR(3 downto 0); CIN: in BIT;
SUM: out BIT_VECTOR(3 downto 0); COUT: out BIT);
end FULL_ADD4:

architecture FOR_GENERATE of FULL_ADD4 is
component FULL_ADDER
port (A, B, C: in BIT; COUT, SUM: out BIT);
end component;

signal CAR: BIT_VECTOR(4 downto 0);
begin
CAR(0) <= CIN;
GK: for K in 3 downto 0 generate
FA: FULL_ADDER port map (CAR(K), A(K), B(K), CAR(K+1),SUM(K));
end generate GK;
COUT <= CAR(4);
end FOR_GENERATE;
```

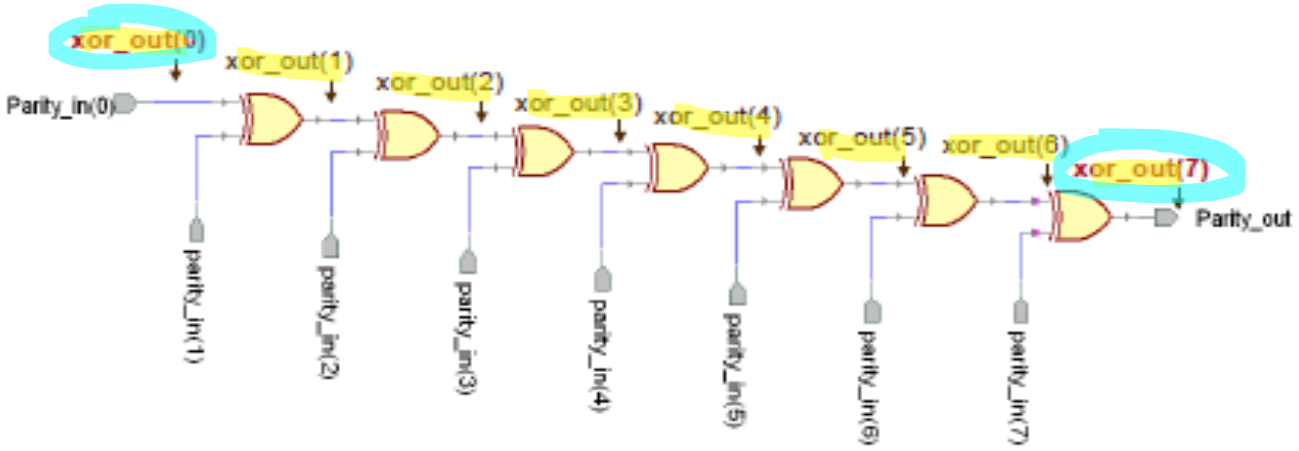
loop 4 times

Example:




```
ARCHITECTURE parity_dataflow OF parity IS

SIGNAL xor_out: STD_LOGIC_VECTOR (6 DOWNTO 1);

BEGIN
    G2: FOR i IN 1 TO 7 GENERATE
        left_xor:  IF i=1 GENERATE
            xor_out(i) <= parity_in(i-1) XOR parity_in(i);
        END GENERATE;
        middle_xor: IF (i>1) AND (i<7) GENERATE
            xor_out(i) <= xor_out(i-1) XOR parity_in(i);
        END GENERATE;
        right_xor: IF i=7 GENERATE
            parity_out <= xor_out(i-1) XOR parity_in(i);
        END GENERATE;
    END GENERATE;
END parity_dataflow;
```

Example:




```
ARCHITECTURE parity_dataflow OF parity IS

SIGNAL xor_out: STD_LOGIC_VECTOR (7 DOWNTO 0);

BEGIN

    xor_out(0) <= parity_in(0);

    G2: FOR i IN 1 TO 7 GENERATE
            xor_out(i) <= xor_out(i-1) XOR parity_in(i);
    END GENERATE G2;

    parity_out <= xor_out(7);

END parity_dataflow;
```

New signal declaration
=> One generate statement

one generate