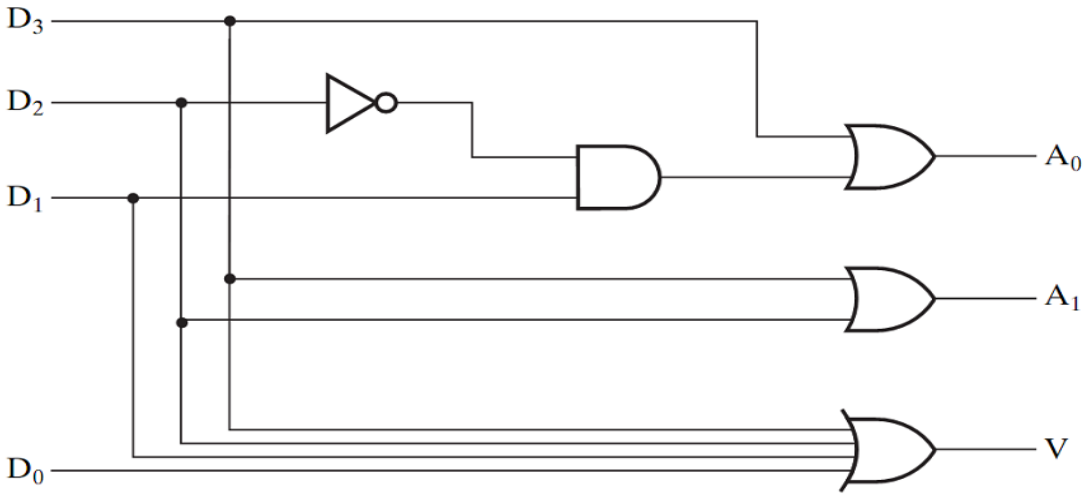
Tutorial 5 – Sample Solutions
VHDL

CO 2206 Computer Organization

Task 1

- For the logic circuit given below,
  - Determine the Boolean function for A0, A1 and V.
  - Write a high-level behavioral VHDL description to describe it.
  - Write a structural VHDL description to describe it.



Task 1: Ans.

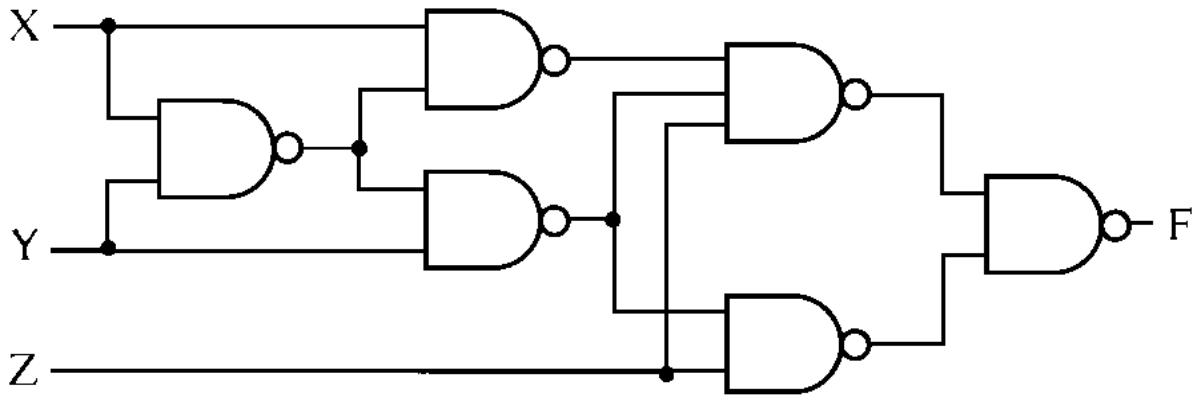
- A0 = B3 + D2'D1; A1 = D3+D2; V=D3+D2+D1+D0

```
-- Task 1
-- Behavioral VHDL Description

library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;

entity question3_cct is
      port(D3,D2,D1,D0:in std_logic;
            A0,A1,V:out std_logic);
end question3_cct;

architecture dataflow_1 of question3_cct is

begin
      A0 <= D3 or ( (not D2) and D1);
      A1 <= D3 or D2;
      V <= D3 or D2 or D1 or D0;
end structural_1;
```

```
-- Task 1
-- Structural VHDL Description

library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;

entity question3_cct is
      port(D3,D2,D1,D0:in std_logic;
            A0,A1,V:out std_logic);
end question3_cct;

architecture structural_1 of question3_cct is
-- complete this section with gates declaration

signal D2_n,N1: std_logic;

begin
      g0: NOT1 port map(D2,D2_n);
      g1: AND2 port map(D2_n,D1,N1);
      g2: OR2 port map(D3,N1,A0);
      g3: OR2 port map(D3,D2,A1);
      g4: OR4 port map(D3,D2,D1,D0,V);
end structural_1;
```

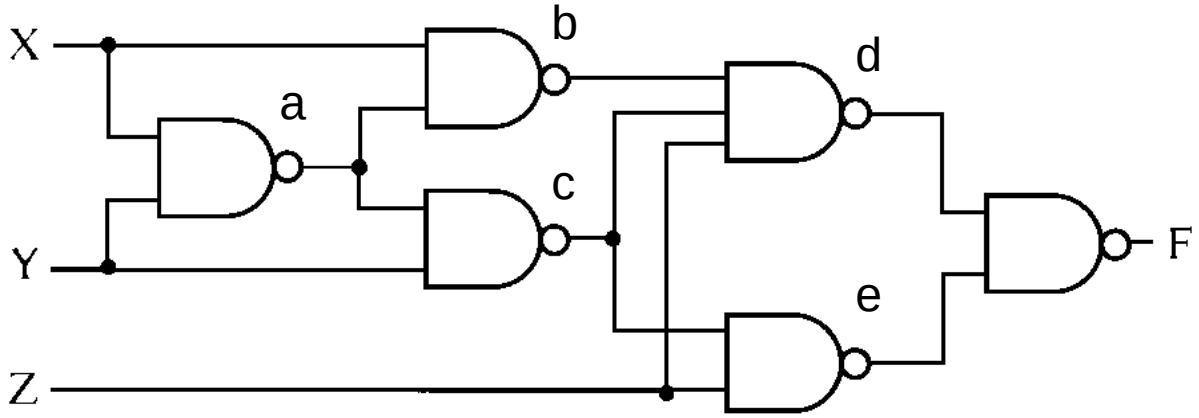
Task 2

- Write a dataflow description for the circuit in the figure below by using the Boolean equation for the output F.
- Write a test bench to simulate the operation of the circuit for all input combinations.



Task 2. Ans (1)

- Label internal signals



Task 2. Ans (2) Dataflow

```
-- data flow for Tutorial 5, Task 2
library ieee;
use ieee.std_logic_1164.all;
entity comb_ckt_2 is
    port (X,Y,Z: in std_logic;
        F: out std_logic);
end comb_ckt_2;

architecture dataflow_1 of comb_ckt_2 is

signal a,b,c,d,e: std_logic;

begin
    a <= X nand Y; b<= X nand a; c <= Y nand a;
    d <= b nand (c nand Z); e <= c nand Z;
    F <= d nand e;
end dataflow_1;
```

Task 2. Ans (2) Testbench

```
-- Test bench for Tutorial 5, Task 2
library ieee;
use ieee.std_logic_1164.all;
entity comb_ckt_2_tb is
end comb_ckt_2_tb;

architecture testbench_1 of comb_ckt_2 is

component comb_ckt_2 is
    port (X,Y,Z: in std_logic;
        F: out std_logic);
end component;

signal Xt,Yt,Zt,Ft: std_logic;

begin
    DUT: comb_ckt_2 port map (Xt,Yt,Zt,Ft);
    STIMULUS: process
        begin
        Xt <= '0'; Yt <= '0'; Zt <= '0';
        wait for 10 ns;
        Xt <= '0'; Yt <= '0'; Zt <= '1';
        wait for 10 ns;
        Xt <= '0'; Yt <= '1'; Zt <= '0';
        wait for 10 ns;
        Xt <= '0'; Yt <= '1'; Zt <= '1';
        wait for 10 ns;
        Xt <= '1'; Yt <= '0'; Zt <= '0';
        wait for 10 ns;
        Xt <= '1'; Yt <= '0'; Zt <= '1';
        wait for 10 ns;
        Xt <= '1'; Yt <= '1'; Zt <= '0';
        wait for 10 ns;
        Xt <= '1'; Yt <= '1'; Zt <= '1';
        wait for 10 ns;
    end process STIMULUS;
end testbench_1;
```

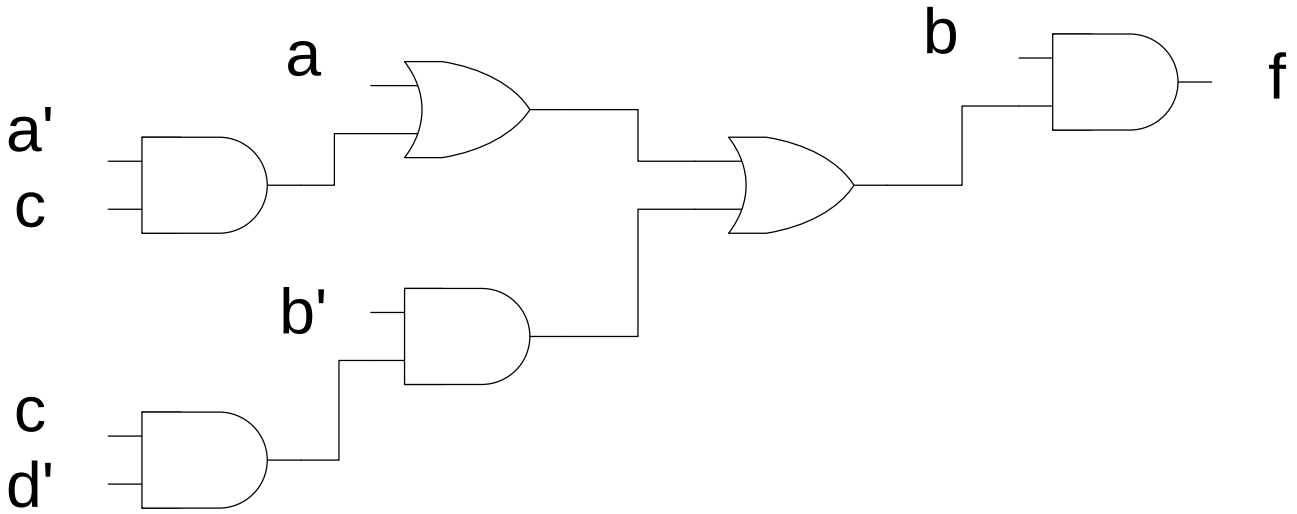
Task 3

- Find a logic diagram representing minimum two-level logic to implement the VHDL dataflow description below. Note that complemented inputs are available.

```
-- Tutorial 5, Task 3
library ieee;
use ieee.std_logic_1164.all;
entity comb_ckt_3 is
    port (a, b, c, d, a_n, b_n, c_n, d_n: in std_logic;
        f, g: out std_logic);
-- a_n, b_n, ... are complements of a, b, ..., respectively
end comb_ckt_3;

architecture dataflow_1 of comb_ckt_3 is
begin
    f <= b and ((a or (a_n and c)) or (b_n and (c and d_n)));
    g <= b and (c or (a_n and c_n) or (c_n and d_n));
end dataflow_1;
```

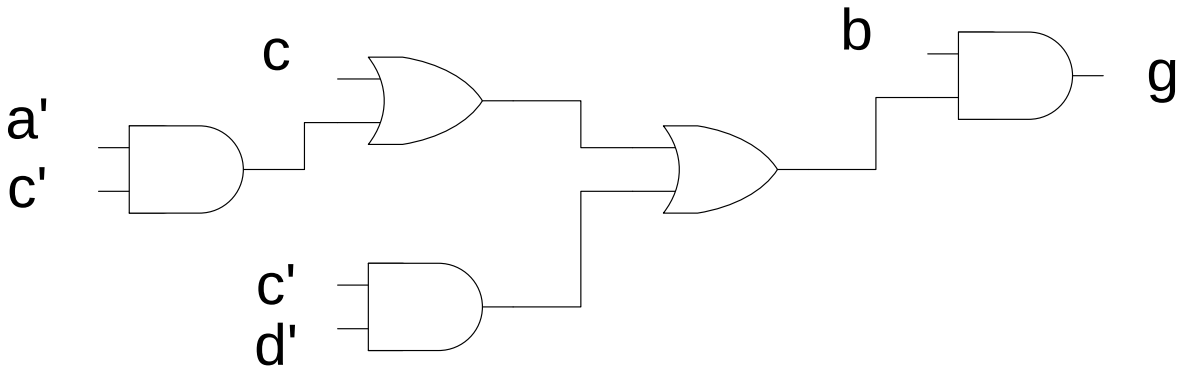
Task 3. Ans (1)

- `f <= b and ((a or (a_n and c)) or (b_n and (c and d_n)));`



Task 3. Ans (2)

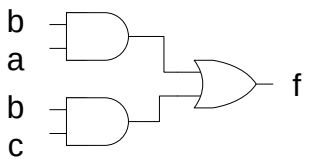
- `g <= b and (c or (a_n and c_n) or (c_n and d_n));`



Task 3. Ans (3)

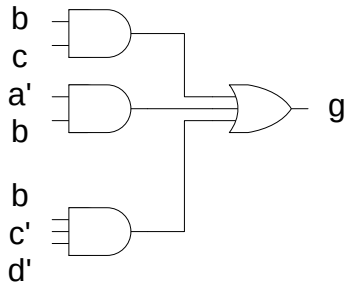
- However the previous implementation was not “two-level”. A two –level implementation can be either in SOP or POS. SOP or POS can be obtained by expanding the function or using K-map.

f = b . ((a + (a’ . c)) + (b’ . (c . d’)))
= b . ((a+a’).(a+c) + b’.c.d’)
= b . (a + c + b’.c.d’)
= b.a + b.c + b.b’.c.d’
= b.a + b.c



f = b.a + b.c

g = b . (c + (a’ . c’) + (c’ . d’))
= b . ((c+a’).(c+c’) + c’.d’)
= b . (c + a’ + c’d’)
= b.c + a’.b + bc’d’



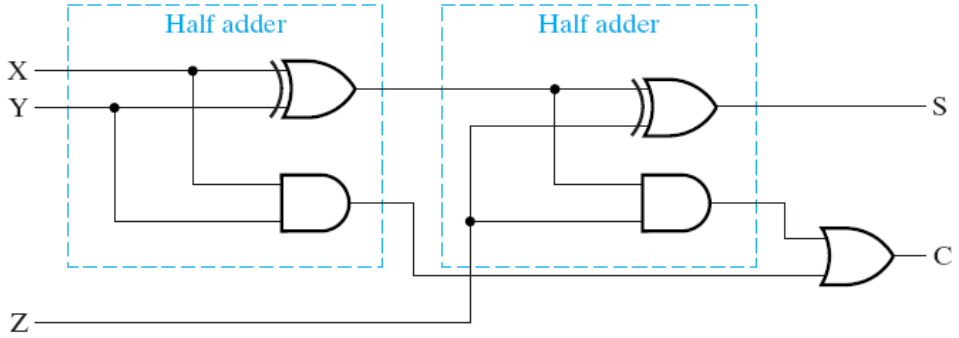
g = b.c + a’.b + bc’d’

Task 4

- Write a high-level behavioral VHDL description for a 1-bit full adder.
- Write a structural VHDL description for a 1-bit adder-subtractor. Make use of the 1-bit full adder above as a component.
- Write a high-level behavioral VHDL description for a 1-bit adder-subtractor.

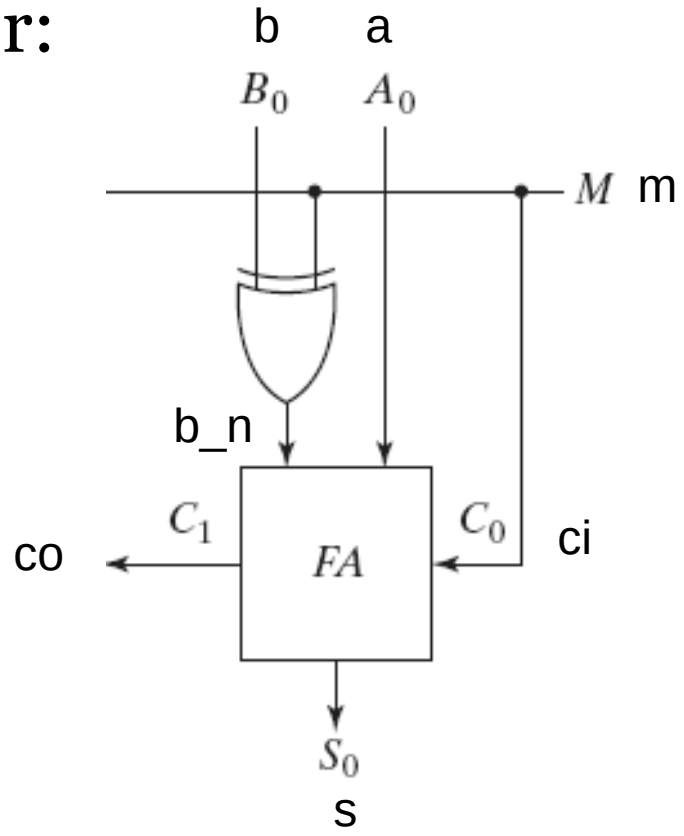
Task 4. Ans (1)

```
-- 1-bit Full Adder: Dataflow
library ieee;
use ieee.std_logic_1164.all;

entity full_adder is
   port (x, y, z : in std_logic;
   s, cout : out std_logic);
end full_adder;

architecture dataflow_1 of full_adder is
begin
   s <= (x xor y) xor z;
   cout <= (z and (x xor y)) or (x and y);
end dataflow_1;
```



Task 4. Ans (2)

- 1-bit Adder-Subtractor:
  - M = 1 → Subtractor
  - M = 0 → Adder



Task 4. Ans (3)

```
-- 1-bit Adder-Subtractor: Structural
library ieee;
use ieee.std_logic_1164.all;

entity adder_subtractor is
   port (a, b, ci, m : in std_logic;
   s, co : out std_logic);
end adder_subtractor;

architecture struc_1 of adder_subtractor is

signal b_n: std_logic;

component full_adder
    port(x,y,z: in std_logic;
        s,cout: out std_logic);
end component;
…
```

Task 4. Ans (4)

```
…
component XOR2
   port(in1,in2: in std_logic;
             out1: out std_logic);
end component;

begin
   g0: XOR2 port map (m,b,b_n);
   FA: full_adder port map (a,b,ci,s,co);
  -- for bit 0, z or ci of FA will be connected to m,
   for higher order bits, z of FA will be
   -- connected to co of previous bit
end struc_1;
```

Task 4. Ans (5)

```
-- 1-bit Adder-Subtractor: Behavioral
library ieee;
use ieee.std_logic_1164.all;

entity adder_subtractor2 is
   port (a, b, ci, m : in std_logic;
         s, co : out std_logic);
end adder_subtractor2;

architecture behavioral_1 of adder_subtractor2 is
begin
   s <= ((a xor b) xor ci) when m = '1' else
        ((a xor (not b)) xor ci);
   co <= ((ci and (a xor b)) or (a and b)) when m = '1' else
         ((ci and (a xor (not b))) or (a and (not b)));
end behavioral_1;
```