7.1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux2Reg\_71 is

port ( A, B: in std\_logic\_vector (7 downto 0);

LDA, SEL, CLK: in std\_logic;

F: out std\_logic\_vector (7 downto 0));

end Mux2Reg\_71;

architecture Mux2Reg of Mux2Reg\_71 is

signal C : std\_logic\_vector(7 downto 0);

begin

process(CLK)

begin

if(CLK' event AND CLK = '1') then

if(LDA = '1') then

F <= C;

end if;

end if;

end process;

with SEL select

C <= B when '0',

A when '1',

B when others;

end Mux2Reg;

7.2

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux2Reg2Reg\_Decoder is

port ( X, Y, Z: in std\_logic\_vector (7 downto 0);

MS: in std\_logic\_vector (1 downto 0);

DS, CLK: in std\_logic;

RB, RA: out std\_logic\_vector (7 downto 0));

end Mux2Reg2Reg\_Decoder;

architecture Mux2Reg2Reg of Mux2Reg2Reg\_Decoder is

signal LD\_A, LD\_B: std\_logic;

signal INT1, INT2, FB: std\_logic\_vector (7 downto 0);

begin

--define the Decoder:

process (DS) -- CASE MUST be in a PROCESS

begin

case (DS) is

when '0' =>

LD\_A <= '1';

LD\_B <= '0';

when '1' =>

LD\_A <= '0';

LD\_B <= '1';

when others =>

LD\_A <='0';

LD\_B <= '0';

end case;

end process;

--define the MUX:、ｚ

with MS select

INT1 <= X when "11",

Y when "10",

Z when "01",

FB when"00",

FB when others;

--define the two FFs:

process(CLK)

begin

if(CLK' event AND CLK = '1') then

if(LD\_A = '1') then

INT2 <= INT1;

--RA <= INT1;

end if;

if(LD\_B = '1') then

RB <= INT2;

end if;

end if;

end process;

RA <= INT2;

end Mux2Reg2Reg;

7.3

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux2Reg2Mux2Reg is

port( X, Y: in std\_logic\_vector(7 downto 1);

LDB, LDA, S1, S0, CLK: in std\_logic;

RB: out std\_logic\_vector(7 downto 1));

end Mux2Reg2Mux2Reg;

architecture Mux2Reg2Mux2Reg of Mux2Reg2Mux2Reg is

signal INT1, INT2, INT3, FB: std\_logic\_vector(7 downto 1); --not just for Flip Flops, so added outside process

begin

with S1 select

INT1 <= X when '1',

FB when '0';

with S0 select

INT3 <= INT2 when '1',

Y when '0';

process(CLK)

begin

if(CLK' event AND CLK = '1') then

if(LDA = '1') then

INT2 <= INT1;

end if;

if(LDB = '1') then

FB <= INT3;

end if;

end if;

end process;

RB <= FB;

end Mux2Reg2Mux2Reg;

7.4

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity MUS2ANDReg\_Mux2ANDReg is

port(X, Y: in std\_logic\_vector(7 downto 0);

LDA, LDB, RD, S0, S1, CLK: in std\_logic;

RA, RB: out std\_logic\_vector(7 downto 0));

end MUS2ANDReg\_Mux2ANDReg;

architecture MUX\_REG of MUS2ANDReg\_Mux2ANDReg is

signal FW1\_1, FW3\_1, FB1\_3: std\_logic\_vector(7 downto 0);

begin

with S1 select

FW1\_1 <= X when '1',

Y when '0',

Y when others;

with S0 select

FW3\_1 <= FB1\_3 when '1',

Y when '0',

Y when others;

process(CLK)

begin

if(CLK' event AND CLK = '1') then

if(LDB = '1' AND (NOT RD = '1')) then

FB1\_3 <= FW1\_1;

end if;

if(LDA = '1' AND RD = '1') then

RA <= FW3\_1;

end if;

end if;

end process;

RB <= FB1\_3;

end MUX\_REG;

7.5

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Dec2Reg\_Mux2Reg is

port( A, B, C: in std\_logic\_vector(7 downto 0);

SL1, SL2, CLK: in std\_logic;

RAX, RBX: out std\_logic\_vector(7 downto 0));

end Dec2Reg\_Mux2Reg;

architecture Dec\_Reg of Dec2Reg\_Mux2Reg is

signal Dec1, Dec2: std\_logic; --use vector type here?

signal FW: std\_logic\_vector(7 downto 0);

begin

--define the Mux behavior

with SL2 select

FW <= B when '1',

C when '0';

process(CLK, SL1)

begin

--define the Decoder behavior

case(SL1) is

when '1' =>

Dec1 <= '1'; Dec2 <= '0';

when '0' =>

Dec1 <= '0'; Dec2 <= '1';

end case;

--define the FFs

if(CLK' event AND CLK = '1') then

if(Dec1 = '1') then

RAX <= A;

end if;

if(Dec2 = '1') then

RBX <= FW;

end if;

end if;

end process;

end Dec\_Reg;

7.6

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux2Reg\_Dec2Reg is

port( A, B, C: in std\_logic\_vector(7 downto 0);

SEL1, SEL2, CLK: in std\_logic;

RAP, RBP: out std\_logic\_vector(7 downto 0));

end Mux2Reg\_Dec2Reg;

architecture Mux2Reg\_Dec2Reg of Mux2Reg\_Dec2Reg is

signal Dec1, Dec2: std\_logic; --use vector type here?

signal FW: std\_logic\_vector(7 downto 0);

begin

--define the Mux behavior

with SEL1 select

FW <= A when '1',

B when '0';

process(CLK, SEL2)

begin

--define the Decoder behavior

case(SEL2) is

when '1' =>

Dec1 <= '1'; Dec2 <= '0';

when '0' =>

Dec1 <= '0'; Dec2 <= '1';

when others =>

Dec1 <= '0'; Dec2 <= '0';

end case;

--define the FFs

if(CLK' event AND CLK = '1') then

if(Dec1 = '1') then

RAP <= FW;

end if;

if(Dec2 = '1') then

RBP <= C;

end if;

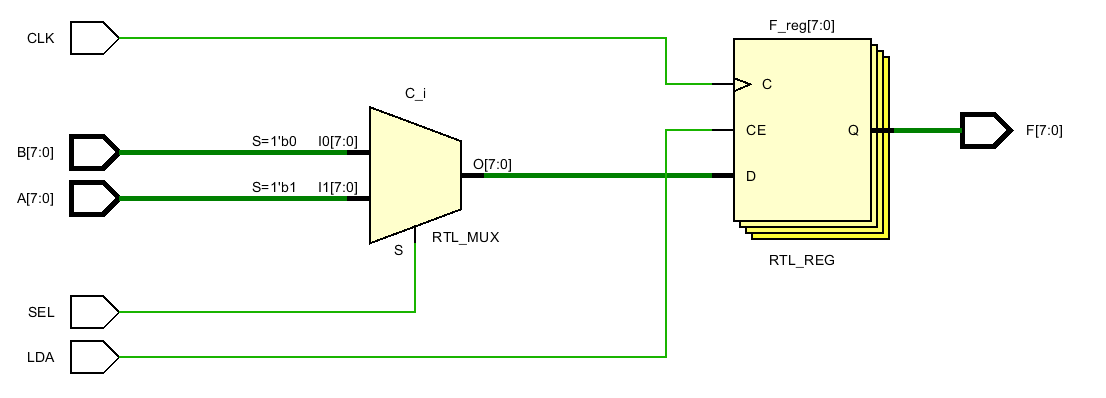
end if;

end process;

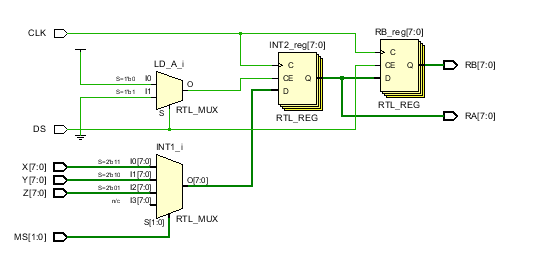
end Mux2Reg\_Dec2Reg;

## Schematics:

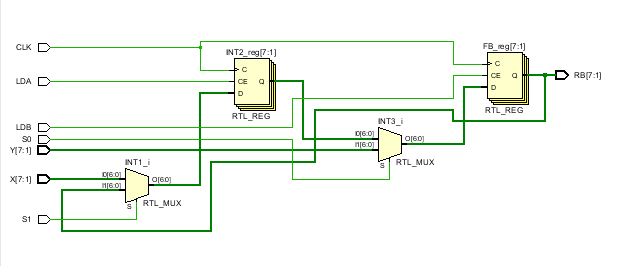
7.1



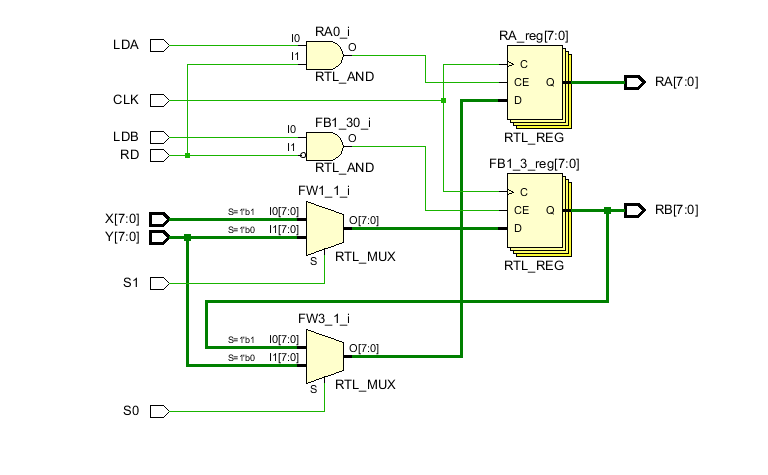
7.2



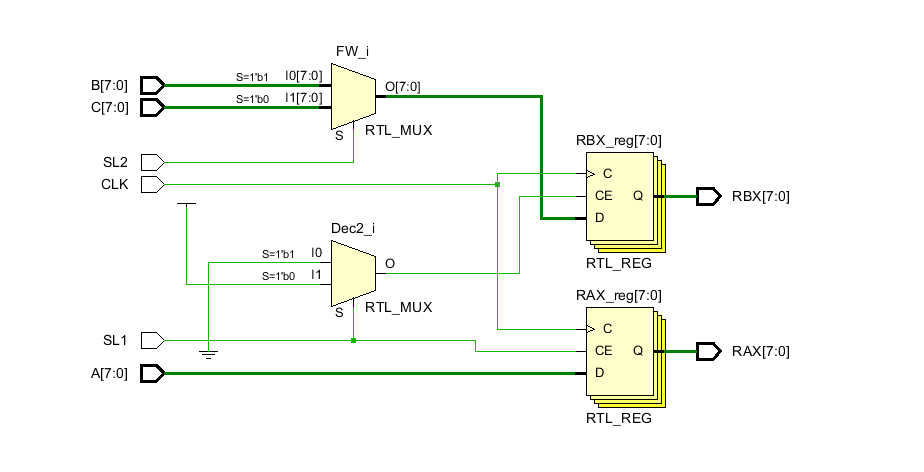
7.3



7.4



7.5



7.6

