HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY – VNU HCMC
OFFICE FOR INTERNATIONAL STUDY PROGRAM
FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING

*

DIGITAL SYSTEMS (LAB)
EXPERIMENTAL REPORT (Lab 2)

Lecturer : **Mr. Nguyễn Tuấn Hùng**
Subject : **Digital Systems**
Class : **TT06**
Name : **Lương Triển Thắng**
Student ID : **2051194**

Ho Chi Minh City, 7th June, 2022

Contents

II Laboratory 2

Adder and flip-flop

1. Known how to program 4-bit adder

a. Code

FA.vhd

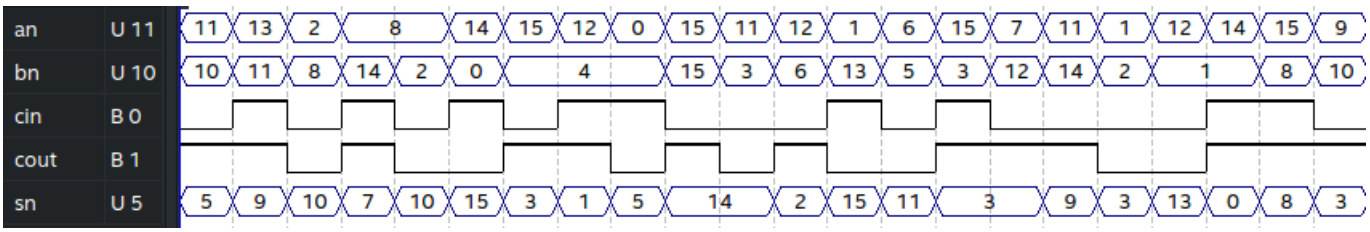
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Exc1 IS
        PORT (
                a : IN STD_LOGIC;
                b : IN STD_LOGIC;
                ci : IN STD_LOGIC;
                s : OUT STD_LOGIC;
                co : OUT STD_LOGIC
        );
END Exc1;

ARCHITECTURE arch OF Exc1 IS
BEGIN
        s <= a XOR b XOR ci;
        co <= (a AND b) OR (ci AND (a XOR b));
END ARCHITECTURE;
```

Exc1.vhd

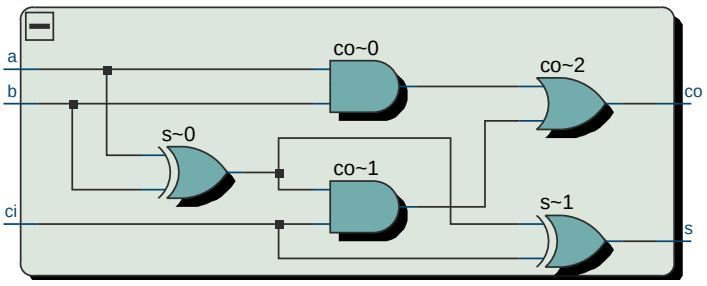
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc1 IS
        PORT (
                an : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                bn : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                cin : IN STD_LOGIC;
                sn : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
                cout : OUT STD_LOGIC
        );
END Exc1;

ARCHITECTURE arch OF Exc1 IS
        SIGNAL cn : STD_LOGIC_VECTOR(4 DOWNTO 1);
        COMPONENT FA IS
                PORT (
                        a : IN STD_LOGIC;
                        b : IN STD_LOGIC;
                        ci : IN STD_LOGIC;
                        s : OUT STD_LOGIC;
                        co : OUT STD_LOGIC
                );
        END COMPONENT;
BEGIN
        FA0 : FA PORT MAP(
                a => an(0),
```

```
                b => bn(0),
                ci => cin,
                s => sn(0),
                co => cn(1)
        );
        gen : FOR i IN 1 TO 3 GENERATE
                FAn : FA PORT MAP(
                        a => an(i),
                        b => bn(i),
                        ci => cn(i),
                        s => sn(i),
                        co => cn(i + 1)
                );
        END GENERATE;
        cout <= cn(4);
END ARCHITECTURE;
```

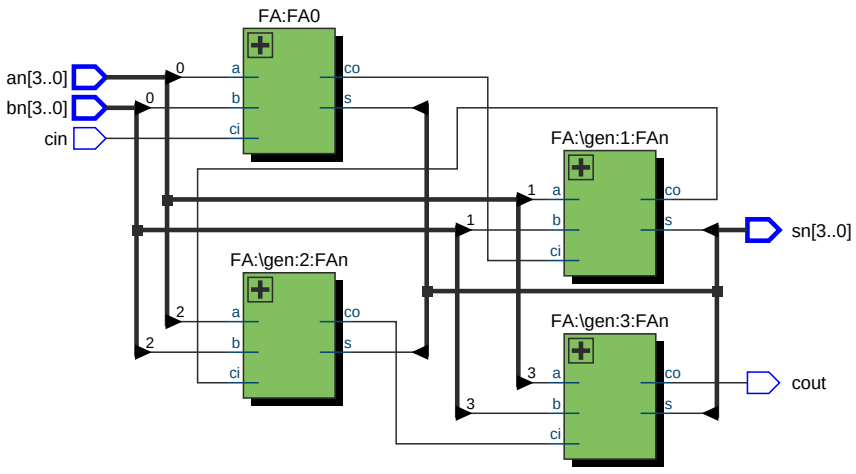
b. Waveform



c. Result of RTL viewer



Full adder



Top level

2. Known how to program BCD adder

Comparator > 9

v_4	v_3	v_2	v_1	v_0	z
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

$$\Rightarrow z = v_3(v_2 + v_1)$$

Circuit A

v_4	v_3	v_2	v_1	v_0	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	×	×	×	×	×
0	0	0	0	1	×	×	×	×	×
0	0	0	1	0	×	×	×	×	×
0	0	0	1	1	×	×	×	×	×
0	0	1	0	0	×	×	×	×	×
0	0	1	0	1	×	×	×	×	×
0	0	1	1	0	×	×	×	×	×
0	0	1	1	1	×	×	×	×	×
0	1	0	0	0	×	×	×	×	×
0	1	0	0	1	×	×	×	×	×
0	1	0	1	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	1
0	1	1	0	0	0	0	0	1	0
0	1	1	0	1	0	0	0	1	1
0	1	1	1	0	0	0	1	0	0
0	1	1	1	1	0	0	1	0	1
1	0	0	0	0	0	0	1	1	0
1	0	0	0	1	0	0	1	1	1
1	0	0	1	0	0	1	0	0	0
1	0	0	1	1	0	1	0	0	1
1	0	1	0	0	×	×	×	×	×
1	0	1	0	1	×	×	×	×	×
1	0	1	1	0	×	×	×	×	×
1	0	1	1	1	×	×	×	×	×
1	1	0	0	0	×	×	×	×	×
1	1	0	0	1	×	×	×	×	×
1	1	0	1	0	×	×	×	×	×
1	1	0	1	1	×	×	×	×	×
1	1	1	0	0	×	×	×	×	×
1	1	1	0	1	×	×	×	×	×
1	1	1	1	0	×	×	×	×	×
1	1	1	1	1	×	×	×	×	×

$$\begin{aligned} A_4 &= 0 \\ A_3 &= \overline{v}_3 v_1 \\ \Rightarrow A_2 &= \overline{v_2 \oplus v_1} \\ A_1 &= \overline{v}_1 \\ A_0 &= v_0 \end{aligned}$$

a. Code

MUX.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY MUX IS
        PORT (
                muxIn1 : IN STD_LOGIC;
                muxSel : IN STD_LOGIC;
                muxIn2 : IN STD_LOGIC;
                muxOut : OUT STD_LOGIC
        );
END ENTITY;
ARCHITECTURE arch OF MUX IS
BEGIN
        muxOut <= (NOT(muxSel) AND muxIn1) OR (muxSel AND muxIn2);
END arch;
```

FourBitMUX.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY FourBitMUX IS
	PORT (
		fourBitMuxIn1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		fourBitMuxIn2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		fourBitMuxSel : IN STD_LOGIC;
		fourBitMuxOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
	);
END ENTITY;

ARCHITECTURE arch OF FourBitMUX IS
	COMPONENT MUX
		PORT (
			muxIn1 : IN STD_LOGIC;
			muxSel : IN STD_LOGIC;
			muxIn2 : IN STD_LOGIC;
			muxOut : OUT STD_LOGIC
		);
	END COMPONENT;
BEGIN
	gen : FOR i IN 4 DOWNTO 0 GENERATE
		MUX2 : MUX PORT MAP(
			muxSel => fourBitMuxSel,
			muxIn1 => fourBitMuxIn1(i),
			muxIn2 => fourBitMuxIn2(i),
			muxOut => fourBitMuxOut(i)
		);
	END GENERATE;
END arch;
```

Comparator.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Comparator IS
	PORT (
		compIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		compOut : OUT STD_LOGIC
	);
END ENTITY;

ARCHITECTURE behave OF Comparator IS
BEGIN
	-- Z = A+B(C+D)
	compOut <= compIn(4) OR (compIn(3) AND (compIn(2) OR compIn(1)));

END ARCHITECTURE;
```

CircuitA.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY CircuitA IS
	PORT (
		dIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		dOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
	);
END ENTITY;
```

```
ARCHITECTURE behave OF CircuitA IS
BEGIN
        -- V = 0
        dOut(4) <= '0';

        -- W = B'*D
        dOut(3) <= NOT(dIn(3)) AND dIn(1);

        -- Y = C'D' + CD
        dOut(2) <= dIn(2) XNOR dIn(1);

        -- X = D'
        dOut(1) <= NOT(dIn(1));

        -- Z = E
        dOut(0) <= dIn(0);

END ARCHITECTURE;
```

BCD.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY BCD IS
        PORT (
                c : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
        );
END BCD;

ARCHITECTURE behavior OF BCD IS
        SIGNAL HEX : STD_LOGIC_VECTOR(0 TO 6);
BEGIN
        HEXn <= NOT(HEX);
        WITH c SELECT
                HEX <= "1111110" WHEN "00000",
                "0110000" WHEN "00001",
                "1101101" WHEN "00010",
                "1111001" WHEN "00011",
                "0110011" WHEN "00100",
                "1011011" WHEN "00101",
                "1011111" WHEN "00110",
                "1110000" WHEN "00111",
                "1111111" WHEN "01000",
                "1111011" WHEN "01001",
                "0000000" WHEN OTHERS;
END behavior;
```

BCDDisplay.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY BCDDisplay IS
        PORT (
                V : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX1 : OUT STD_LOGIC_VECTOR(0 TO 6)
        );
END ENTITY;
```

```
ARCHITECTURE behave OF BCDDisplay IS

        SIGNAL z : STD_LOGIC;
        SIGNAL A, m : STD_LOGIC_VECTOR(4 DOWNTO 0);

        COMPONENT FourBitMUX IS
                PORT (
                        fourBitMuxIn1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        fourBitMuxIn2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        fourBitMuxSel : IN STD_LOGIC;
                        fourBitMuxOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
                );
        END COMPONENT;

        COMPONENT Comparator IS
                PORT (
                        compIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        compOut : OUT STD_LOGIC
                );
        END COMPONENT;

        COMPONENT CircuitA IS
                PORT (
                        dIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        dOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
                );
        END COMPONENT;

        COMPONENT BCD IS
                PORT (
                        c : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
        END COMPONENT;
BEGIN
        comp : Comparator PORT MAP(
                compIn => V,
                compOut => z
        );

        cirA : CircuitA PORT MAP(
                dIn => V,
                dOut => A
        );

        mux : FourBitMUX PORT MAP(
                fourBitMuxIn1 => V,
                fourBitMuxIn2 => A,
                fourBitMuxSel => z,
                fourBitMuxOut => m
        );

        HEX01 : BCD PORT MAP(c => "0000" & z, HEXn => HEX1);
        HEX00 : BCD PORT MAP(c => m, HEXn => HEX0);

END ARCHITECTURE;
```

FA.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Exc1 IS
```

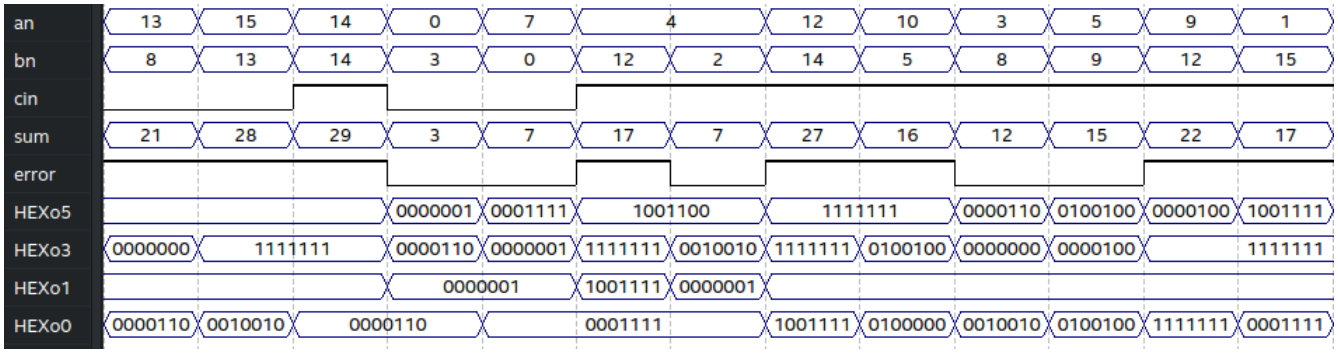
```
        PORT (
                a : IN STD_LOGIC;
                b : IN STD_LOGIC;
                ci : IN STD_LOGIC;
                s : OUT STD_LOGIC;
                co : OUT STD_LOGIC
        );
END Exc1;

ARCHITECTURE arch OF Exc1 IS
BEGIN
        s <= a XOR b XOR ci;
        co <= (a AND b) OR (ci AND (a XOR b));
END ARCHITECTURE;
```

Exc2.vhd

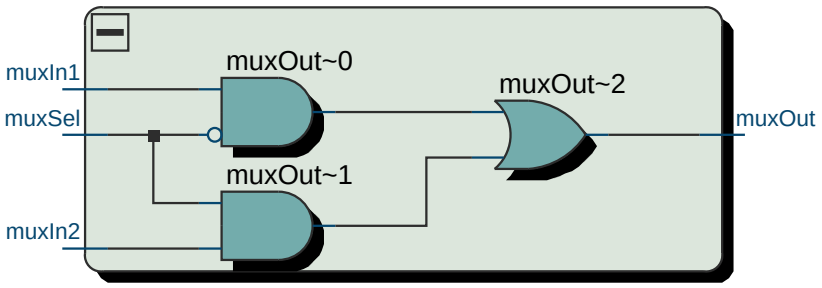
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc2 IS
        PORT (
                an : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                bn : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                cin : IN STD_LOGIC;
                HEXo0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo1 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo3 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo5 : OUT STD_LOGIC_VECTOR(0 TO 6);
                error : OUT STD_LOGIC;
                sum : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
        );
END ENTITY;

ARCHITECTURE arch OF Exc2 IS
        SIGNAL cn : STD_LOGIC_VECTOR(4 DOWNTO 1);
        SIGNAL sn : STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL errorA, errorB : STD_LOGIC;
        COMPONENT FA IS
                PORT (
                        a : IN STD_LOGIC;
                        b : IN STD_LOGIC;
                        ci : IN STD_LOGIC;
                        s : OUT STD_LOGIC;
                        co : OUT STD_LOGIC
                );
        END COMPONENT;

        COMPONENT BCDDisplay IS
                PORT (
                        V : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                        HEX1 : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
        END COMPONENT;

        COMPONENT BCD IS
                PORT (
                        c : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
        END COMPONENT;
```

```
        COMPONENT Comparator IS
                PORT (
                        compIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        compOut : OUT STD_LOGIC
                );
        END COMPONENT;
BEGIN
        HEX5 : BCD PORT MAP(c => '0' & an, HEXn => HEXo5);
        HEX3 : BCD PORT MAP(c => '0' & bn, HEXn => HEXo3);

        FA0 : FA PORT MAP(
                a => an(0),
                b => bn(0),
                ci => cin,
                s => sn(0),
                co => cn(1)
        );
        gen : FOR i IN 1 TO 3 GENERATE
                FAn : FA PORT MAP(
                        a => an(i),
                        b => bn(i),
                        ci => cn(i),
                        s => sn(i),
                        co => cn(i + 1)
                );
        END GENERATE;

        sn(4) <= cn(4);
        sum <= sn;

        BCDHEX : BCDDisplay PORT MAP(V => sn, HEX0 => HEXo0, HEX1 => HEXo1);
        errorLED0 : Comparator PORT MAP(compIn => '0' & an, compOut => errorA);
        errorLED1 : Comparator PORT MAP(compIn => '0' & bn, compOut => errorB);

        error <= errorA OR errorB;
END ARCHITECTURE;
```

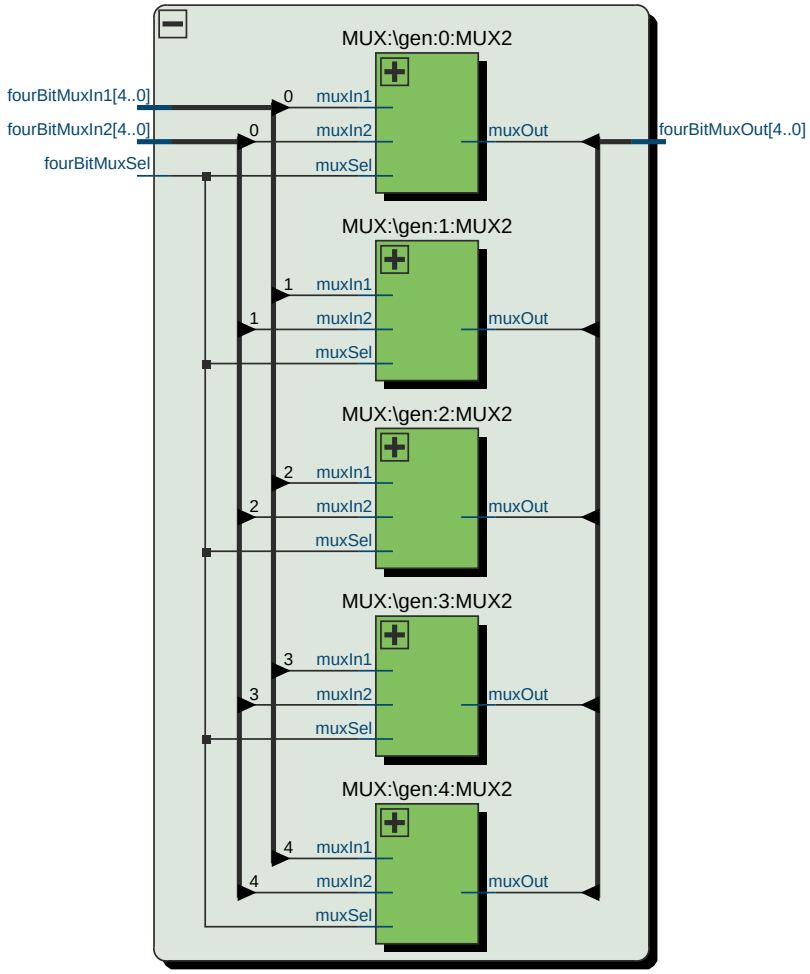
b. Waveform



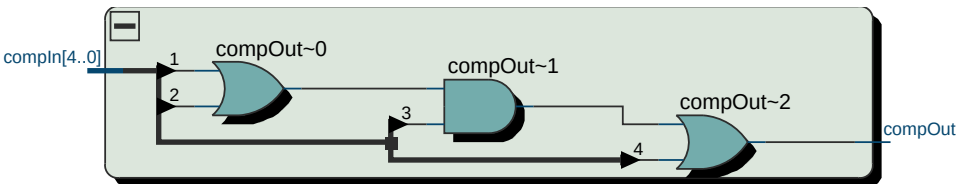
c. Result of RTL viewer



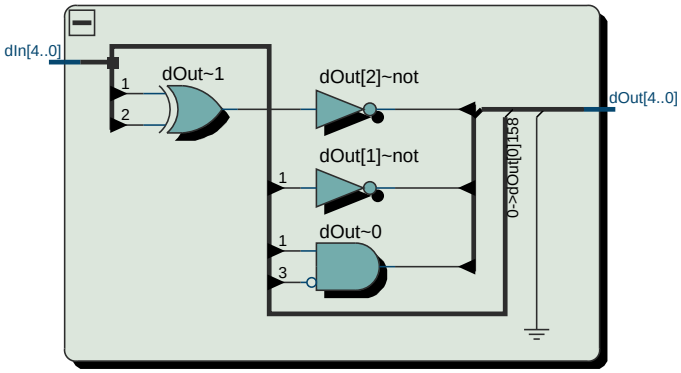
MUX



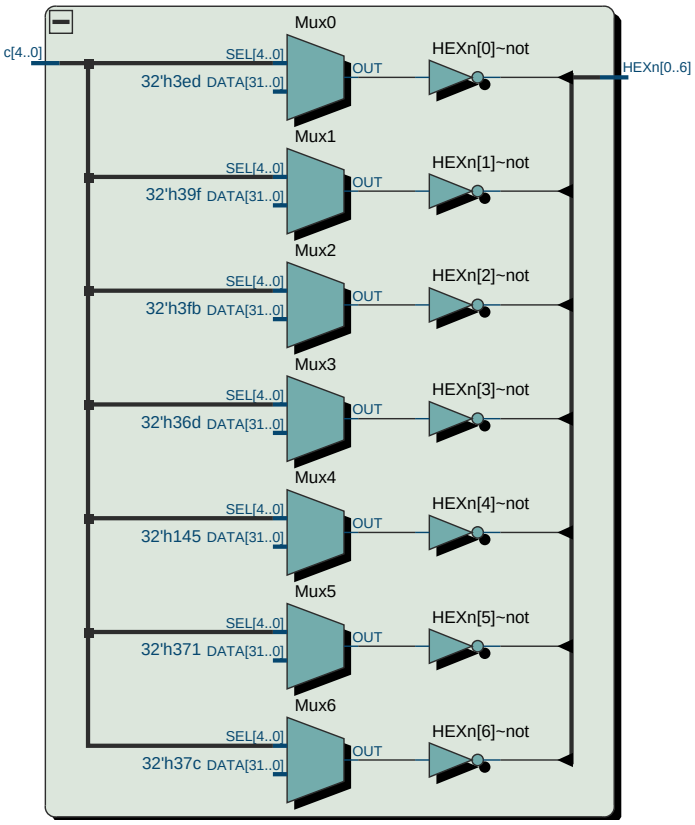
4-bit MUX



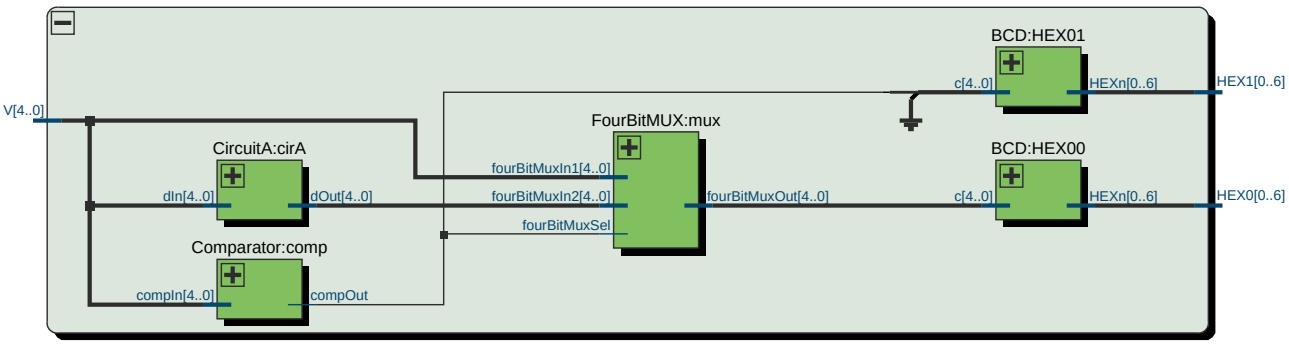
Comparator



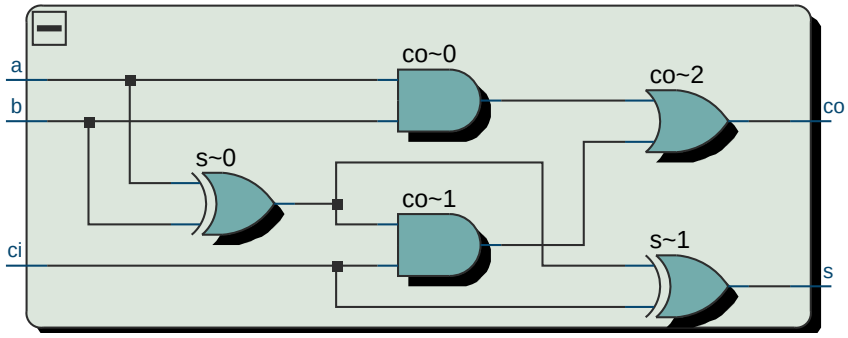
CircuitA



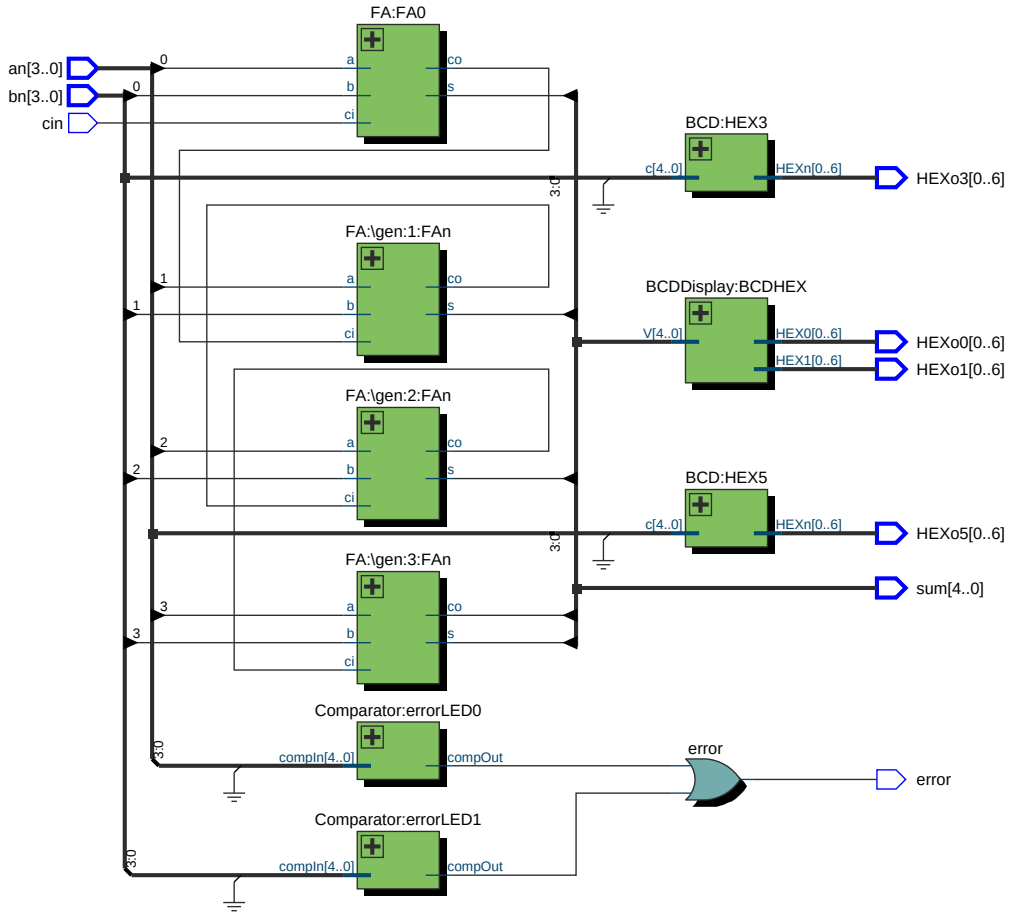
BCD MUX



BCD Decoder



Full adder



Top level

3. Known how to program BCD adder

a. Code

BCD.vhd

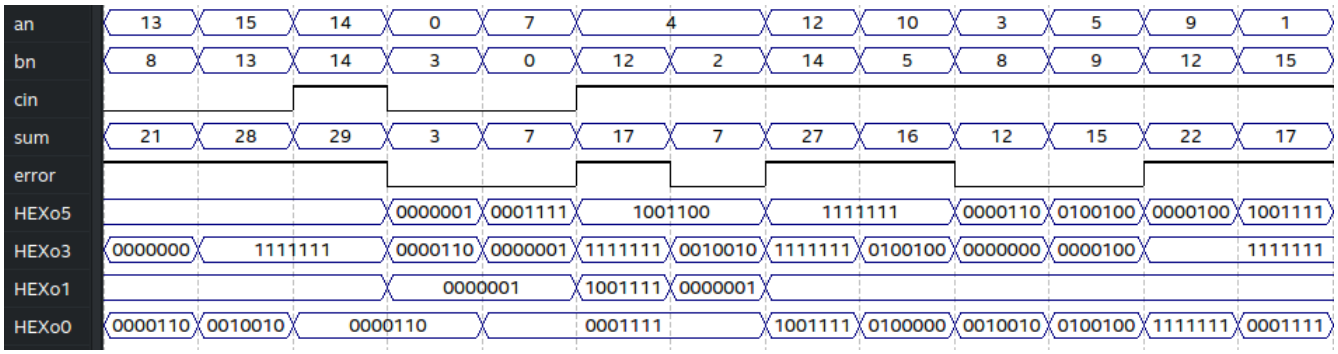
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY BCD IS
        PORT (
                c : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
        );
END BCD;

ARCHITECTURE behavior OF BCD IS
        SIGNAL HEX : STD_LOGIC_VECTOR(0 TO 6);
BEGIN
        HEXn <= NOT(HEX);
        WITH c SELECT
                HEX <= "1111110" WHEN "00000",
                "0110000" WHEN "00001",
                "1101101" WHEN "00010",
                "1111001" WHEN "00011",
                "0110011" WHEN "00100",
                "1011011" WHEN "00101",
                "1011111" WHEN "00110",
                "1110000" WHEN "00111",
                "1111111" WHEN "01000",
                "1111011" WHEN "01001",
                "0000000" WHEN OTHERS;
END behavior;
```

Exc3.vhd

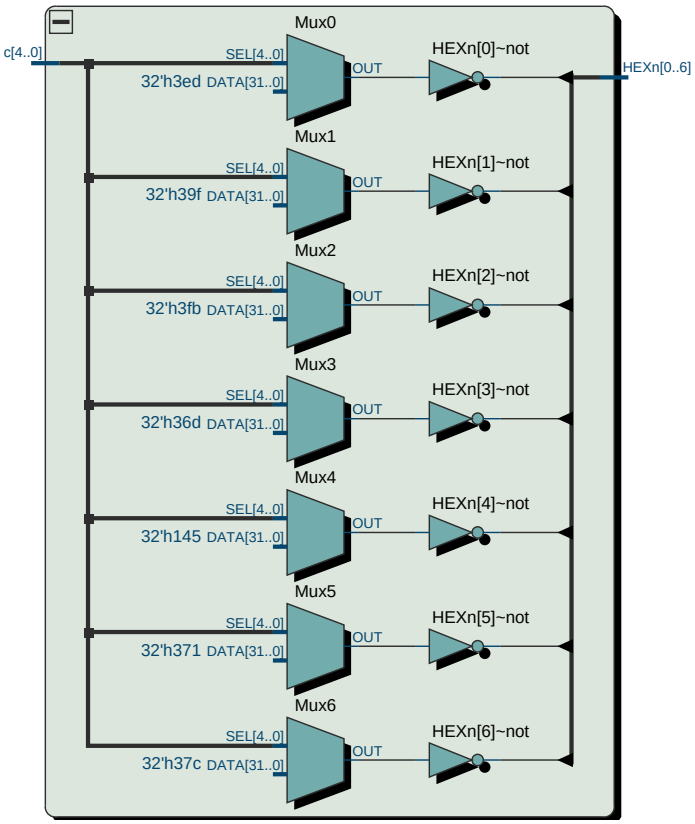
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc3 IS
        PORT (
                an : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                bn : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                cin : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
                HEXo0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo1 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo3 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEXo5 : OUT STD_LOGIC_VECTOR(0 TO 6);
                sum : OUT STD_LOGIC_VECTOR(4 DOWNTO 0);
                error : OUT STD_LOGIC
        );
END ENTITY;

ARCHITECTURE arch OF Exc3 IS
        SIGNAL sn : STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL digit1, digit0 : STD_LOGIC_VECTOR(4 DOWNTO 0);

        COMPONENT BCD IS
                PORT (
                        c : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
```

```
        END COMPONENT;
BEGIN
        HEX5 : BCD PORT MAP(c => '0' & an, HEXn => HEXo5);
        HEX3 : BCD PORT MAP(c => '0' & bn, HEXn => HEXo3);

        sn <= STD_LOGIC_VECTOR(unsigned('0' & an) + unsigned('0' & bn) + unsigned(cin));
        digit1 <= STD_LOGIC_VECTOR(unsigned(sn) / 10);
        digit0 <= STD_LOGIC_VECTOR(unsigned(sn) MOD 10);

        HEX01 : BCD PORT MAP(c => digit1, HEXn => HEXo1);
        HEX00 : BCD PORT MAP(c => digit0, HEXn => HEXo0);

        sum <= sn;

        error <= '1' WHEN (unsigned(an) > 9 OR unsigned(bn) > 9) ELSE '0';
END ARCHITECTURE;
```

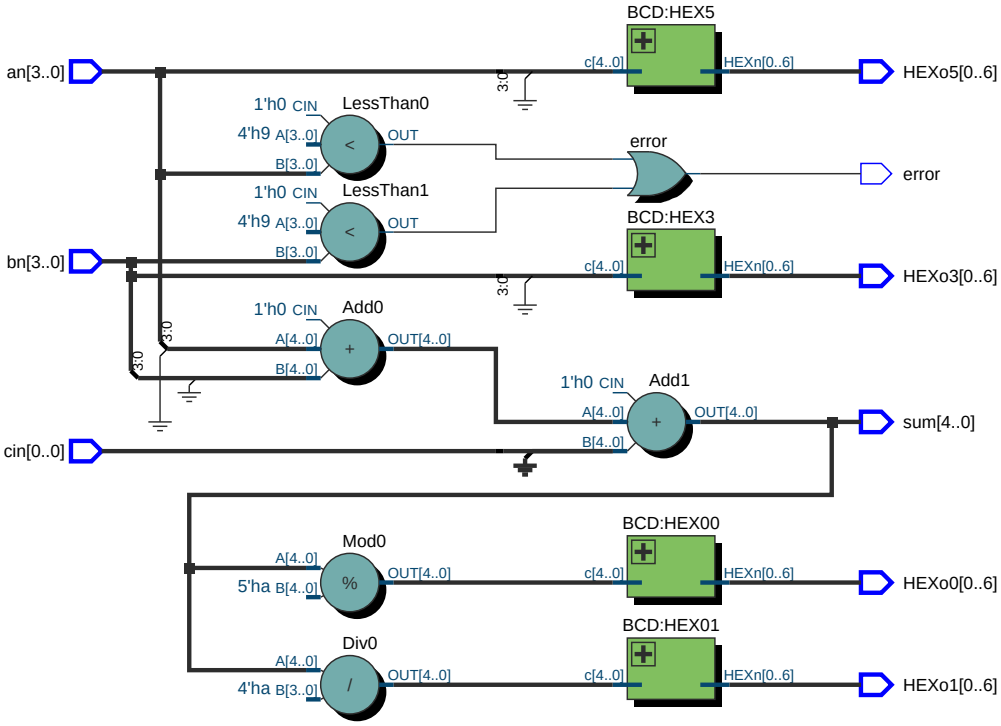
b. Waveform



c. Result of RTL viewer



BCD MUX



Top level

The circuit is simplier when relying more on the VHDL compiler.

4. Known how to program a master-slave D flip-flop

a. Code

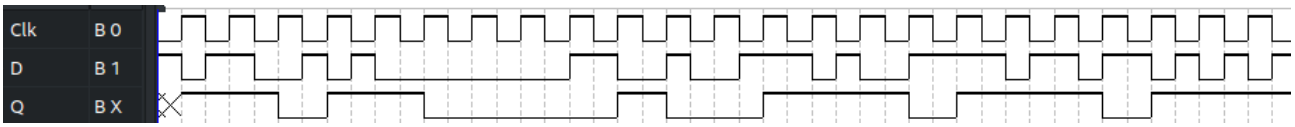
DFFn.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY DFFn IS
        PORT (
                DClk, Din : IN STD_LOGIC;
                DQ : OUT STD_LOGIC
        );
END DFFn;

ARCHITECTURE Structural OF DFFn IS
        SIGNAL R_g, S_g, Qa, Qb, S, R : STD_LOGIC;
        ATTRIBUTE KEEP : BOOLEAN;
        ATTRIBUTE KEEP OF R_g, S_g, Qa, Qb : SIGNAL IS TRUE;

BEGIN
        S <= Din;
        R <= NOT(Din);
        R_g <= R AND DClk;
        S_g <= S AND DClk;
        Qa <= R_g NOR Qb;
        Qb <= S_g NOR Qa;
        DQ <= Qa;
END Structural;
```

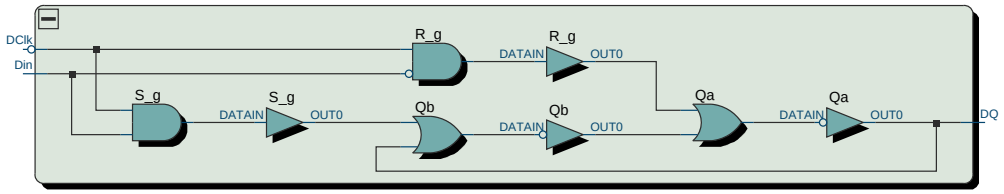
Exc4.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Exc4 IS
        PORT (
                D, Clk : IN STD_LOGIC;
                Q : OUT STD_LOGIC
        );
END Exc4;

ARCHITECTURE arch OF Exc4 IS
        SIGNAL Qm : STD_LOGIC;
        COMPONENT DFFn IS
                PORT (
                        DClk, Din : IN STD_LOGIC;
                        DQ : OUT STD_LOGIC
                );
        END COMPONENT;
BEGIN
        DFF1 : DFFn PORT MAP(DClk => NOT(Clk), Din => D, DQ => Qm);
        DFF0 : DFFn PORT MAP(DClk => Clk, Din => Qm, DQ => Q);
END ARCHITECTURE;
```

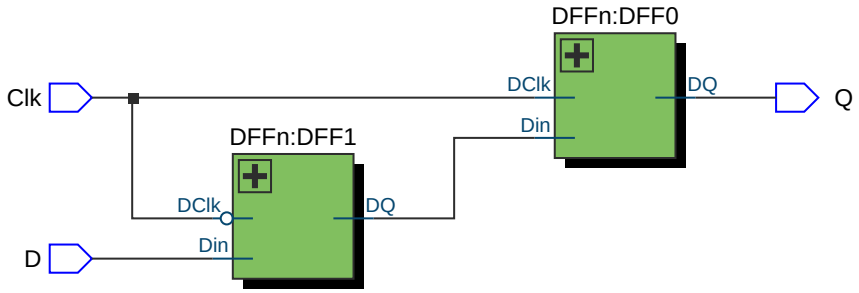
b. Waveform



c. Result of RTL viewer



D flip-flop



Top level

5. Known how to program a master-slave D flip-flopCompare the different behavior of the three storage elements: a gated D latch, a positive-edge triggered D flip-flop, and a negative-edge triggered D flip-flop.

a. Code

DL.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY DL IS
        PORT (
                DLin, DLClk : IN STD_LOGIC;
                DLQ : OUT STD_LOGIC);
END DL;
ARCHITECTURE Behavior OF DL IS
BEGIN
        PROCESS (DLin, DLClk)
        BEGIN
                IF DLClk = '1' THEN
                        DLQ <= DLin;
                END IF;
        END PROCESS;
END Behavior;
```

DFFn.vhd

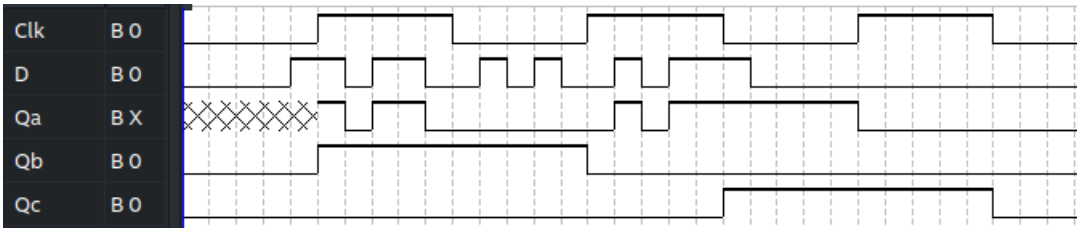
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY DFFn IS
        PORT (
                DClk, Din : IN STD_LOGIC;
                DQ : OUT STD_LOGIC
        );
END DFFn;

ARCHITECTURE Structural OF DFFn IS
BEGIN
        PROCESS (DClk)
        BEGIN
                IF rising_edge(DClk) THEN
                        DQ <= Din;
                END IF;
        END PROCESS;
END Structural;
```

Exc5.vhd

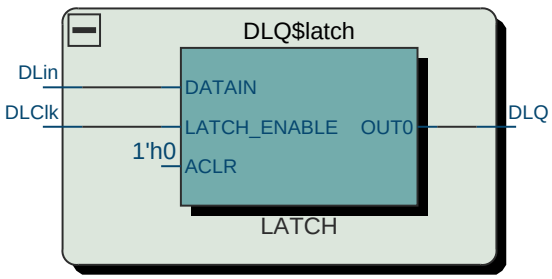
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc5 IS
        PORT (
                D, Clk : IN STD_LOGIC;
                Qa, Qb, Qc : OUT STD_LOGIC
        );
END Exc5;

ARCHITECTURE arch OF Exc5 IS
        COMPONENT DL IS
```

```
                PORT (
                        DLin, DLClk : IN STD_LOGIC;
                        DLQ : OUT STD_LOGIC);
        END COMPONENT;

        COMPONENT DFFn IS
                PORT (
                        Din, DClk : IN STD_LOGIC;
                        DQ : OUT STD_LOGIC);
        END COMPONENT;
BEGIN
        DFF2 : DL PORT MAP(DLin => D, DLClk => Clk, DLQ => Qa);
        DFF1 : DFFn PORT MAP(Din => D, DClk => Clk, DQ => Qb);
        DFF0 : DFFn PORT MAP(Din => D, DClk => NOT(Clk), DQ => Qc);
END ARCHITECTURE;
```

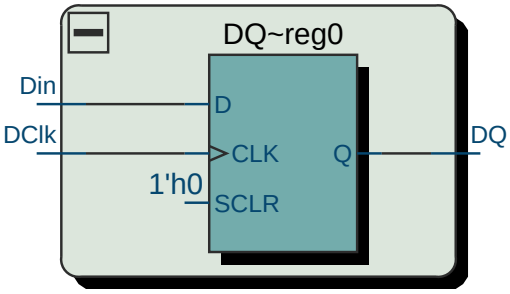
b. Waveform



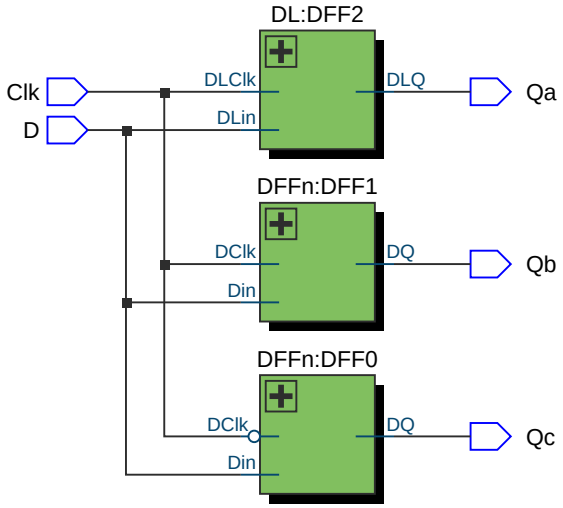
c. Result of RTL viewer



D latch



D flip-flop



Top level

6. Apply all the previous experiments

a. Code

DFFn.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY DFFn IS
        PORT (
                Din, DClk, Drst : IN STD_LOGIC;
                DQ : OUT STD_LOGIC);
END DFFn;
ARCHITECTURE Behavior OF DFFn IS
BEGIN
        PROCESS (Drst, DClk)
        BEGIN
                IF rising_edge(DClk) THEN
                        DQ <= Din;
                END IF;
                IF Drst = '1' THEN
                        DQ <= '0';
                END IF;
        END PROCESS;
END Behavior;
```

EightBitReg.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY EightBitReg IS
        PORT (
                EBRClk, EBRrst : IN STD_LOGIC;
                EBRD : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                EBRQ : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
END EightBitReg;

ARCHITECTURE arch OF EightBitReg IS
        COMPONENT DFFn IS
                PORT (
                        Din, DClk, Drst : IN STD_LOGIC;
                        DQ : OUT STD_LOGIC);
        END COMPONENT;
BEGIN
        gen : FOR i IN 7 DOWNTO 0 GENERATE
                DFFs : DFFn PORT MAP(Din => EBRD(i), DClk => EBRClk, Drst => EBRrst, DQ => EBRQ(i));
        END GENERATE;
END ARCHITECTURE;
```

HEXDisplay.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY HEXDisplay IS
        PORT (
                c : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
        );
END HEXDisplay;
```

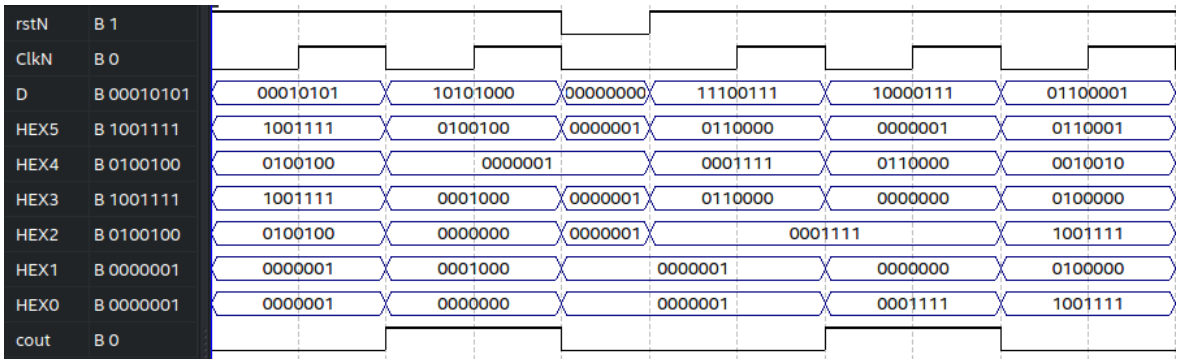
```
ARCHITECTURE behavior OF HEXDisplay IS
        SIGNAL HEX : STD_LOGIC_VECTOR(0 TO 6);
BEGIN
        HEXn <= NOT(HEX);
        WITH c SELECT
                HEX <= "1111110" WHEN "0000",
                "0110000" WHEN "0001",
                "1101101" WHEN "0010",
                "1111001" WHEN "0011",
                "0110011" WHEN "0100",
                "1011011" WHEN "0101",
                "1011111" WHEN "0110",
                "1110000" WHEN "0111",
                "1111111" WHEN "1000",
                "1111011" WHEN "1001",

                "1110111" WHEN "1010",
                "0011111" WHEN "1011",
                "1001110" WHEN "1100",
                "0111101" WHEN "1101",
                "1001111" WHEN "1110",
                "1000111" WHEN "1111",
                "0000000" WHEN OTHERS;
END behavior; -- behavior
```

Exc6.vhd

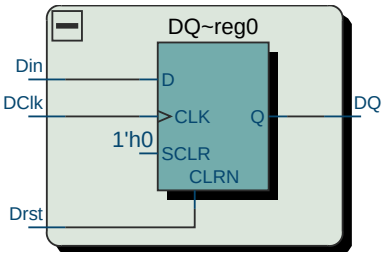
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc6 IS
        PORT (
                ClkN, rstN : IN STD_LOGIC;
                D : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                cout : OUT STD_LOGIC;
                HEX5 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX4 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX3 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX2 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX1 : OUT STD_LOGIC_VECTOR(0 TO 6);
                HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6)
        );
END Exc6;

ARCHITECTURE arch OF Exc6 IS
        SIGNAL sum : STD_LOGIC_VECTOR(8 DOWNTO 0);
        SIGNAL Q : STD_LOGIC_VECTOR(7 DOWNTO 0);
        SIGNAL Clk, rst : STD_LOGIC;
        COMPONENT EightBitReg IS
                PORT (
                        EBRClk, EBRrst : IN STD_LOGIC;
                        EBRD : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        EBRQ : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
                );
        END COMPONENT;

        COMPONENT HEXDisplay IS
                PORT (
                        c : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                        HEXn : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
```

```
        END COMPONENT;
BEGIN
        Clk <= NOT(ClkN);
        rst <= NOT(rstN);
        sum <= STD_LOGIC_VECTOR(unsigned('0' & Q) + unsigned('0' & D));
        EBR : EightBitReg PORT MAP(EBRClk => Clk, EBRrst => rst, EBRD => D, EBRQ => Q);
        HEX05 : HEXDisplay PORT MAP(c => sum(7 DOWNTO 4), HEXn => HEX5);
        HEX04 : HEXDisplay PORT MAP(c => sum(3 DOWNTO 0), HEXn => HEX4);
        HEX03 : HEXDisplay PORT MAP(c => D(7 DOWNTO 4), HEXn => HEX3);
        HEX02 : HEXDisplay PORT MAP(c => D(3 DOWNTO 0), HEXn => HEX2);
        HEX01 : HEXDisplay PORT MAP(c => Q(7 DOWNTO 4), HEXn => HEX1);
        HEX00 : HEXDisplay PORT MAP(c => Q(3 DOWNTO 0), HEXn => HEX0);
        cout <= sum(8);
END ARCHITECTURE;
```

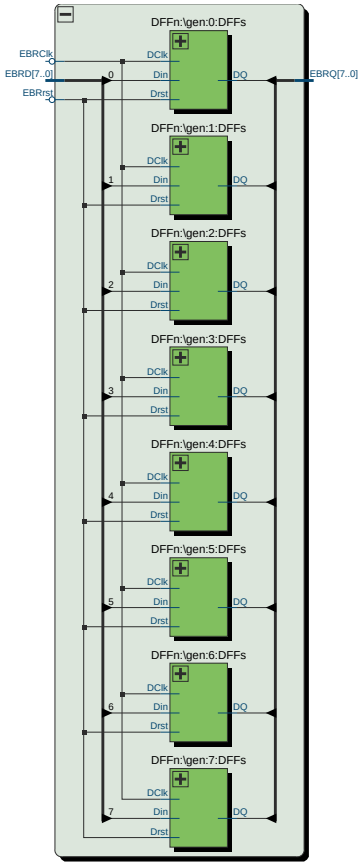
b. Waveform



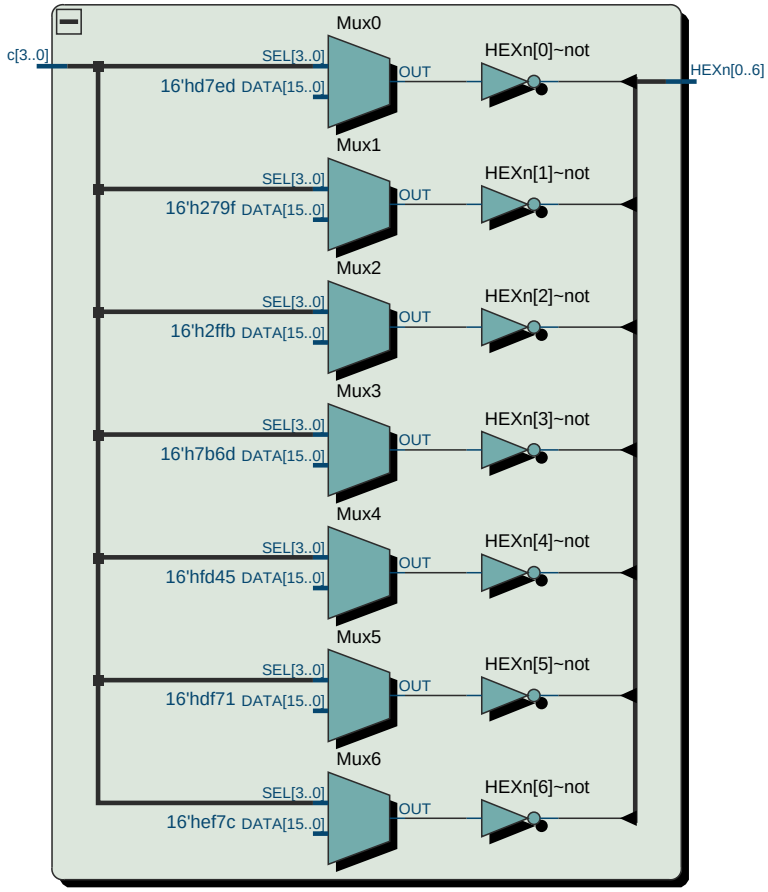
c. Result of RTL viewer



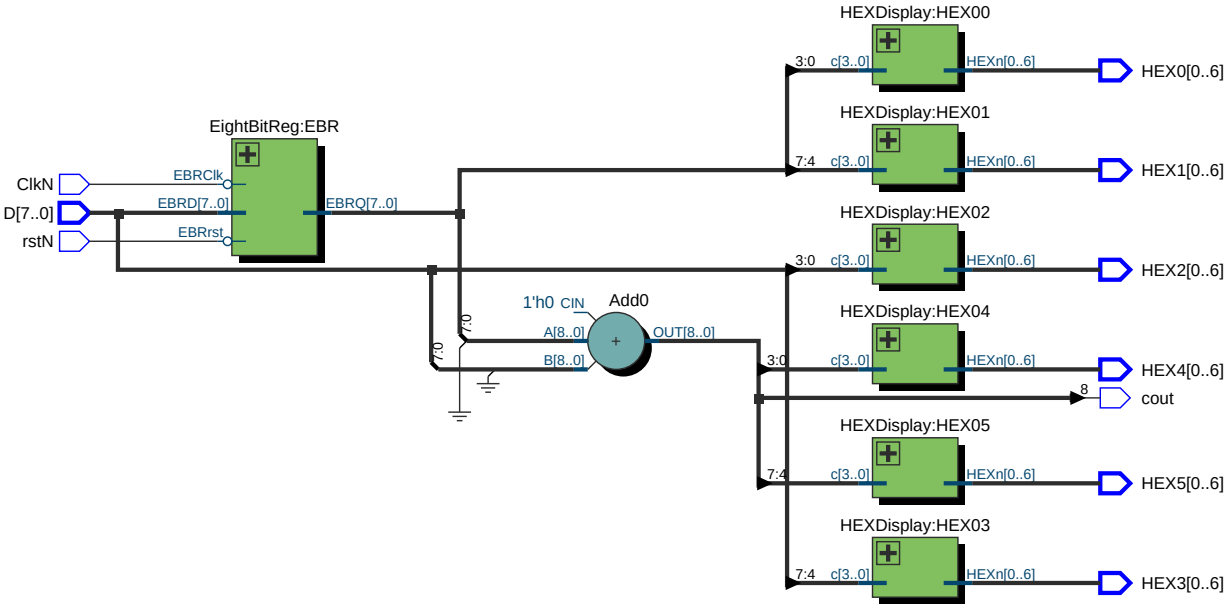
D flip-flop



8-bit register



HEX display decoder



Top level