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 3)

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

III Laboratory 3

Adders, subtractors and multipliers

1. Known how to program a system to add the value of an input A to itself repeatedly

a. Code

DFFn.vhd

```vhdl
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

```vhdl
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;
```

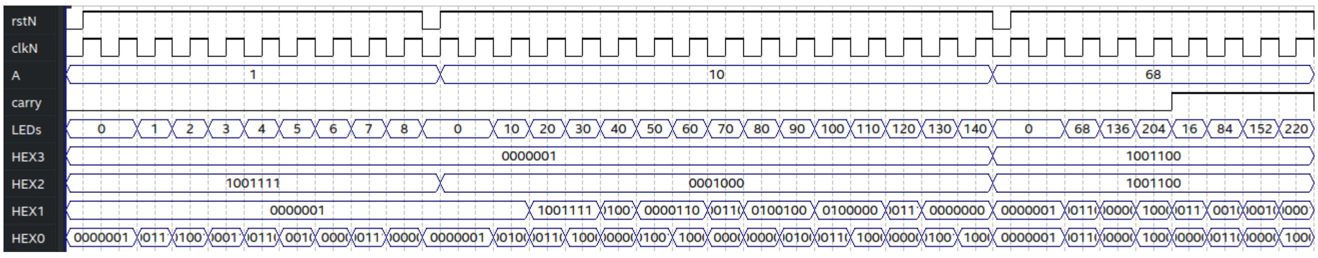
HEXDisplay.vhd

```
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
```

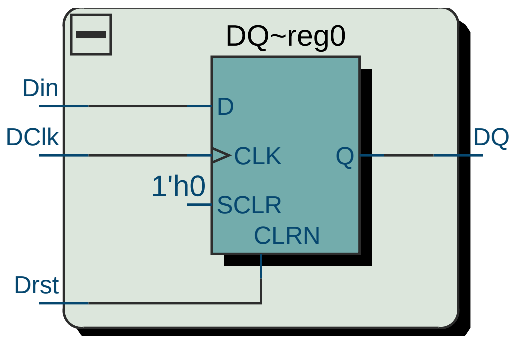
Exc1.vhd

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

ENTITY Exc1 IS
        PORT (
                A : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                LEDs : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                clkN, rstN : IN STD_LOGIC;
                carry : OUT STD_LOGIC
        );
END Exc1;

ARCHITECTURE arch OF Exc1 IS
        SIGNAL rst, clk, ofl, crr : STD_LOGIC;
        SIGNAL Qa, Qb, Qc : STD_LOGIC_VECTOR(8 DOWNTO 0);
        ATTRIBUTE KEEP : BOOLEAN;
        ATTRIBUTE KEEP OF Qc : SIGNAL IS TRUE;

        COMPONENT DFFn IS
                PORT (
                        Din, DClk, Drst : IN STD_LOGIC;
                        DQ : OUT STD_LOGIC);
        END COMPONENT;
        COMPONENT EightBitReg IS
                PORT (
                        EBRClk, EBRrst : IN STD_LOGIC;
                        EBRD : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                        EBRQ : OUT STD_LOGIC_VECTOR(8 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);

        Qb <= STD_LOGIC_VECTOR(unsigned(Qc) + unsigned(Qa));
        EBR0 : EightBitReg PORT MAP(EBRrst => rst, EBRClk => clk, EBRD => '0' & A, EBRQ => Qa);
        EBR1 : EightBitReg PORT MAP(EBRrst => rst, EBRClk => clk, EBRD => Qb, EBRQ => Qc);
        DFF0 : DFFn PORT MAP(Drst => rst, DClk => clk, Din => crr, DQ => carry);

        LEDs <= Qc(7 DOWNTO 0);
        crr <= Qb(8);
        HEX03 : HEXDisplay PORT MAP(c => A(7 DOWNTO 4), HEXn => HEX3);
        HEX02 : HEXDisplay PORT MAP(c => A(3 DOWNTO 0), HEXn => HEX2);
        HEX01 : HEXDisplay PORT MAP(c => Qc(7 DOWNTO 4), HEXn => HEX1);
        HEX00 : HEXDisplay PORT MAP(c => Qc(3 DOWNTO 0), HEXn => HEX0);
END ARCHITECTURE;
```

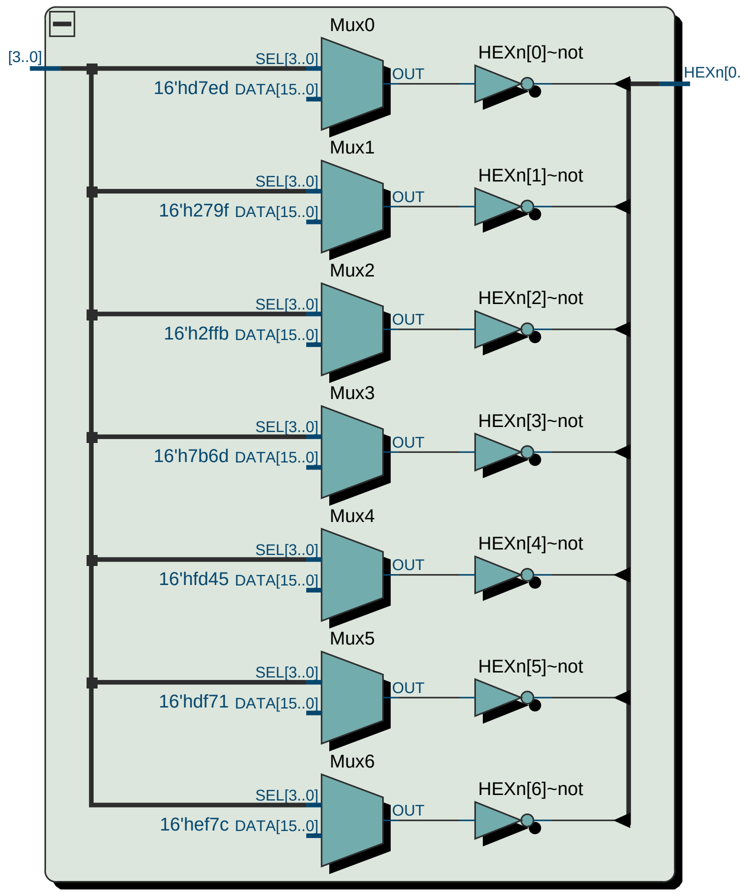
b. Waveform



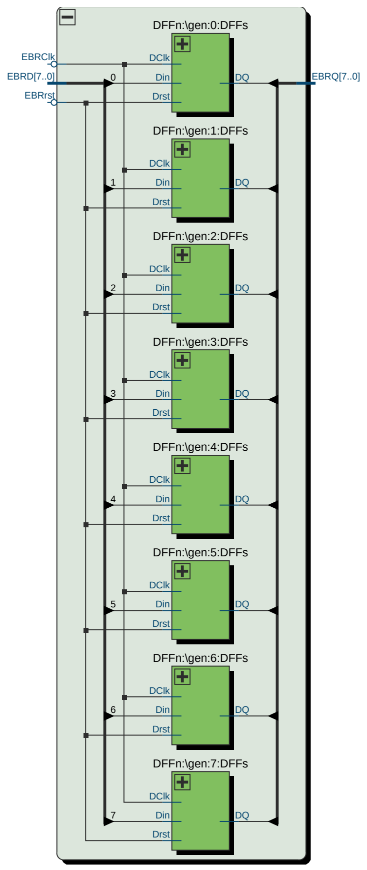
c. Result of RTL viewer



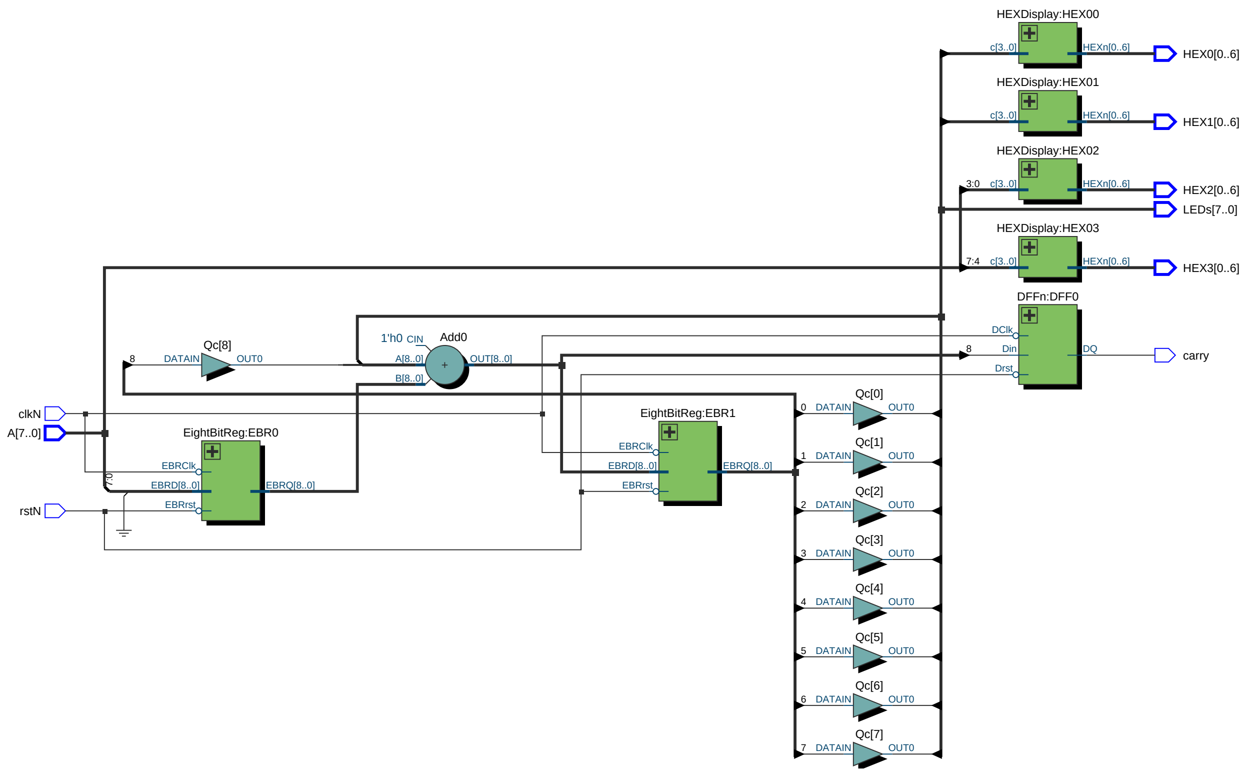
D flip-flop



HEX display decoder



8-bit register



Top level

2. Known how to program a system to add or subtract the value of an input A to itself repeatedly

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)
        );
```

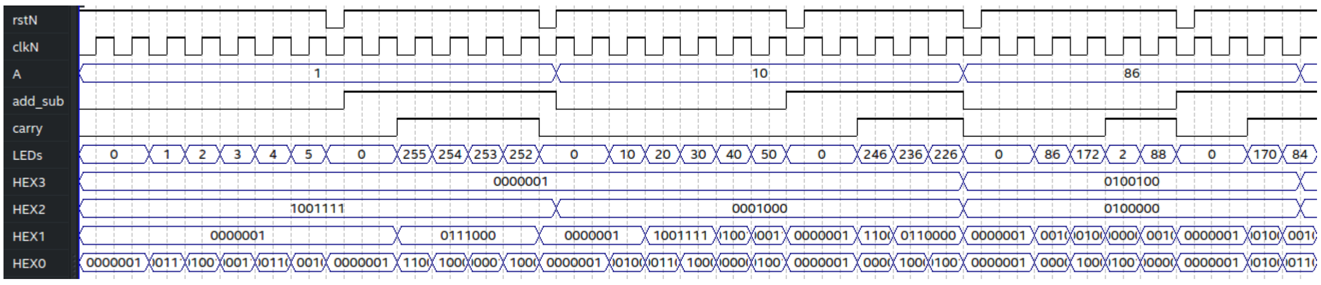
```
END HEXDisplay;

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
```

Exc2.vhd

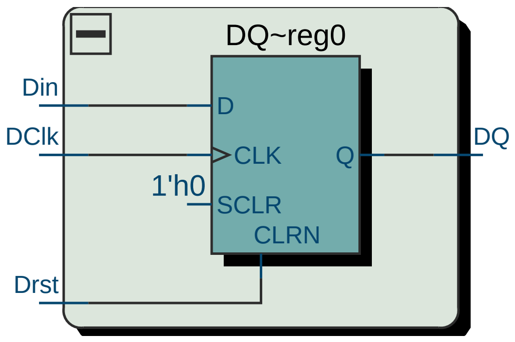
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Exc2 IS
        PORT (
                A : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                LEDs : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                clkN, rstN, add_sub : IN STD_LOGIC;
                carry : OUT STD_LOGIC
        );
END Exc2;

ARCHITECTURE arch OF Exc2 IS
        SIGNAL rst, clk, ofl, crr : STD_LOGIC;
        SIGNAL Qa, Qb, Qc, Qaa : STD_LOGIC_VECTOR(8 DOWNTO 0);
        ATTRIBUTE KEEP : BOOLEAN;
        ATTRIBUTE KEEP OF Qc : SIGNAL IS TRUE;
        COMPONENT DFFn IS
                PORT (
                        Din, DClk, Drst : IN STD_LOGIC;
                        DQ : OUT STD_LOGIC);
        END COMPONENT;

        COMPONENT EightBitReg IS
                PORT (
                        EBRClk, EBRrst : IN STD_LOGIC;
                        EBRD : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                        EBRQ : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
                );
        END COMPONENT;

        COMPONENT BCDDisplay IS
```

```
                PORT (
                        V : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        HEX02 : OUT STD_LOGIC_VECTOR(0 TO 6);
                        HEX01 : OUT STD_LOGIC_VECTOR(0 TO 6);
                        HEX00 : OUT STD_LOGIC_VECTOR(0 TO 6)
                );
        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);

        Qaa <= Qa WHEN add_sub = '0' ELSE STD_LOGIC_VECTOR(unsigned(NOT(Qa)) + 1);
        Qb <= STD_LOGIC_VECTOR(unsigned(Qc) + unsigned(Qaa));
        EBR0 : EightBitReg PORT MAP(EBRrst => rst, EBRClk => clk, EBRD => '0' & A, EBRQ => Qa);
        EBR1 : EightBitReg PORT MAP(EBRrst => rst, EBRClk => clk, EBRD => Qb, EBRQ => Qc);
        DFF0 : DFFn PORT MAP(Drst => rst, DClk => clk, Din => crr, DQ => carry);

        LEDs <= Qc(7 DOWNTO 0);
        crr <= Qb(8);
        HEX03 : HEXDisplay PORT MAP(c => A(7 DOWNTO 4), HEXn => HEX3);
        HEX02 : HEXDisplay PORT MAP(c => A(3 DOWNTO 0), HEXn => HEX2);
        HEX01 : HEXDisplay PORT MAP(c => Qc(7 DOWNTO 4), HEXn => HEX1);
        HEX00 : HEXDisplay PORT MAP(c => Qc(3 DOWNTO 0), HEXn => HEX0);
END ARCHITECTURE;
```

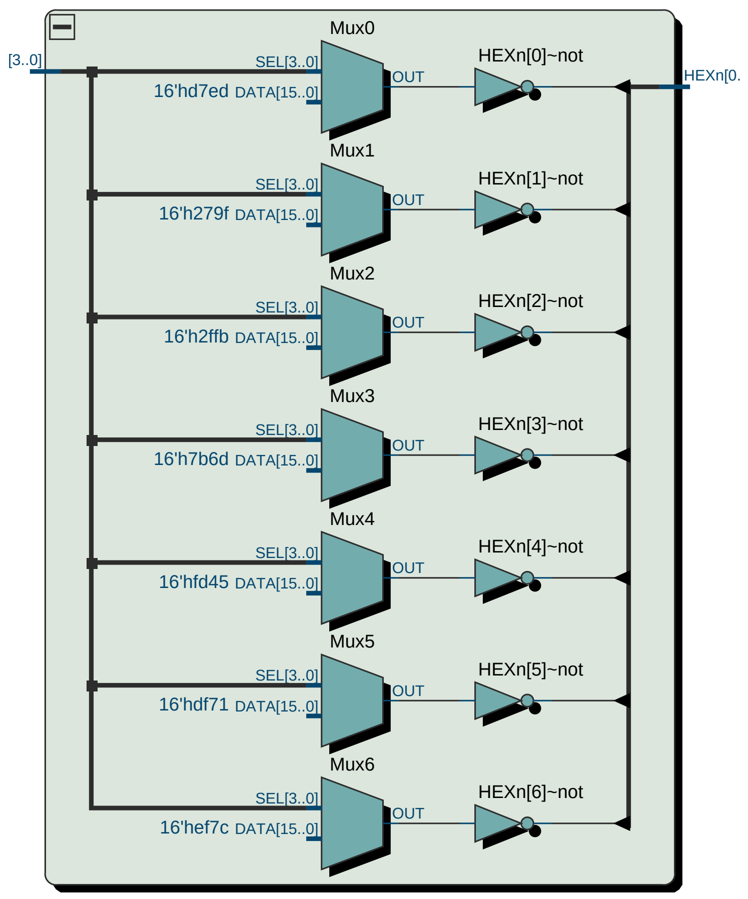
b. Waveform



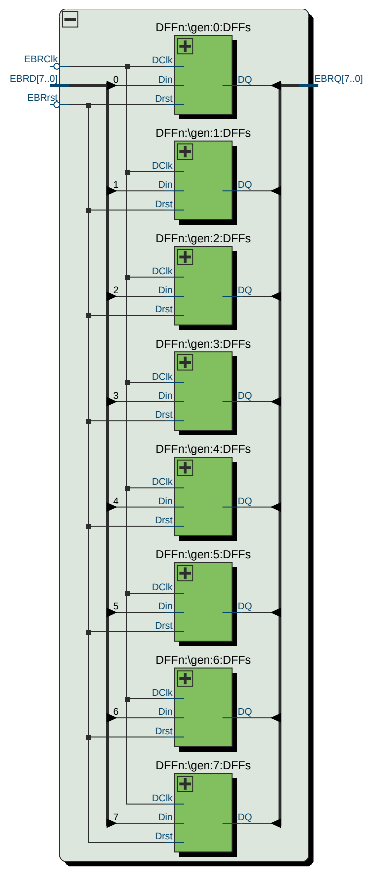
c. Result of RTL viewer



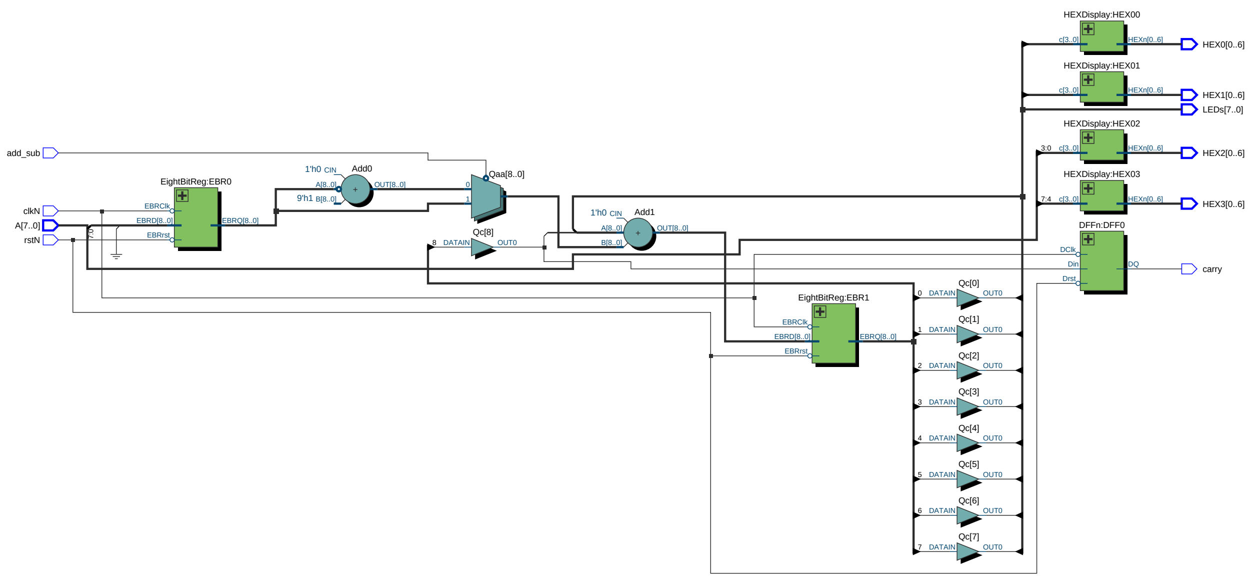
D flip-flop



HEX display decoder



8-bit register



Top level

3. Known how to program 8×8 multiplier circuit with registered inputs and outputs

In this exercise, I only use two registers to store A and B. I removed the last register so that the result can output directly without waiting for another clock.

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;
```

Mul.vhd

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

ENTITY Mul IS
        PORT (
                mulA, mulB : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                mulP : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
        );
END Mul;
```

```
ARCHITECTURE arch OF Mul IS
        SIGNAL a_padded, zero : STD_LOGIC_VECTOR(15 DOWNTO 0);
        TYPE ab_t IS ARRAY(0 TO 7) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
        SIGNAL ab : ab_t;
BEGIN

        zero <= "0000000000000000";
        a_padded <= "00000000" & mulA;

        ab(0) <= a_padded WHEN mulB(0) = '1' ELSE
        zero;
        sum : FOR i IN 1 TO 7 GENERATE
                ab(i) <= a_padded(15 - i DOWNTO 0) & zero(i - 1 DOWNTO 0) WHEN mulB(i) = '1' ELSE
                zero;
        END GENERATE;

        mulP <= ab(0) + ab(1) + ab(2) + ab(3) + ab(4) + ab(5) + ab(6) + ab(7);
END ARCHITECTURE;
```

HEXDisplay.vhd

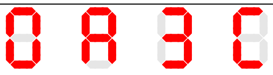
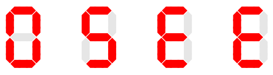
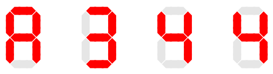
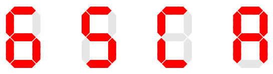
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

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
```

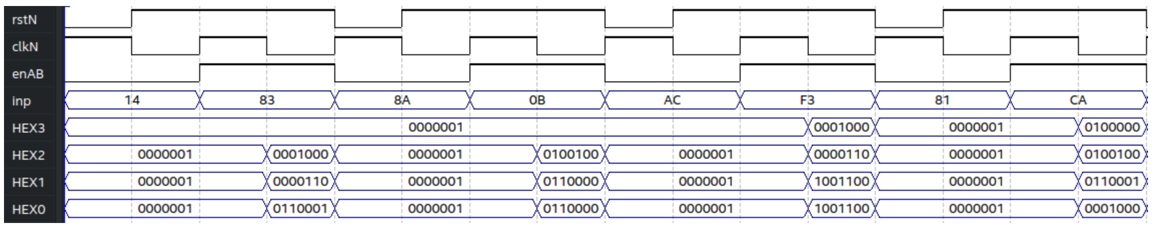
Exc3.vhd

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

ENTITY Exc3 IS
```

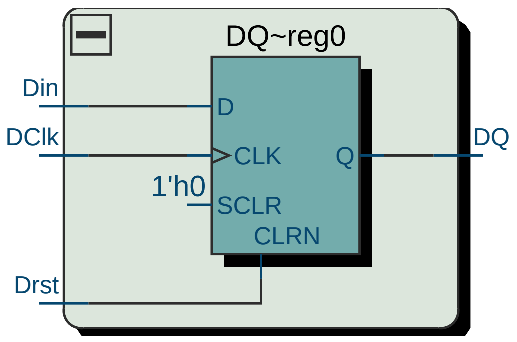
```
        PORT (
                inp : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                enAB, clkN, rstN : IN STD_LOGIC
        );
END Exc3;

ARCHITECTURE arch OF Exc3 IS
        SIGNAL a, b : STD_LOGIC_VECTOR(7 DOWNTO 0);
        SIGNAL p : STD_LOGIC_VECTOR(15 DOWNTO 0);
        SIGNAL clk, rst : STD_LOGIC;
        COMPONENT EightBitReg IS
                PORT (
                        EBRClk, EBRrst, EBRen : IN STD_LOGIC;
                        EBRD : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        EBRQ : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
                );
        END COMPONENT;

        COMPONENT Mul IS
                PORT (
                        mulA, mulB : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        mulP : OUT STD_LOGIC_VECTOR(15 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
        rst <= NOT(rstN);
        clk <= NOT(clkN);
        EBRA : EightBitReg PORT MAP(
                EBRClk => clk, EBRrst => rst,
                EBRen => enAB, EBRD => inp, EBRQ => a);
        EBRB : EightBitReg PORT MAP(
                EBRClk => clk, EBRrst => rst,
                EBRen => NOT(enAB), EBRD => inp, EBRQ => b);

        multiplier : Mul PORT MAP(mulA => a, mulB => b, mulP => p);

        HEX03 : HEXDisplay PORT MAP(c => p(15 DOWNTO 12), HEXn => HEX3);
        HEX02 : HEXDisplay PORT MAP(c => p(11 DOWNTO 8), HEXn => HEX2);
        HEX01 : HEXDisplay PORT MAP(c => p(7 DOWNTO 4), HEXn => HEX1);
        HEX00 : HEXDisplay PORT MAP(c => p(3 DOWNTO 0), HEXn => HEX0);
END ARCHITECTURE;
```

b. Waveform

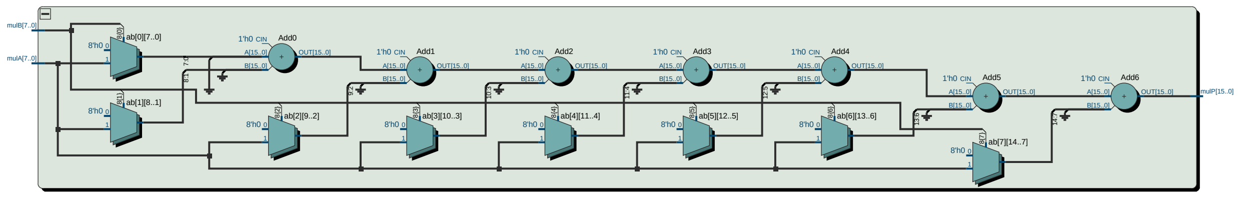
A	B	P	7-segment display	HEX output (active low) 3	2	1	0
14	83	0A3C		0000001	0001000	0000110	0110001
8A	0B	05EE		0000001	0100100	0110000	0110000
AC	F3	A344		0001000	0000110	1001100	1001100
81	CA	65CA		0100000	0100100	0110001	0001000



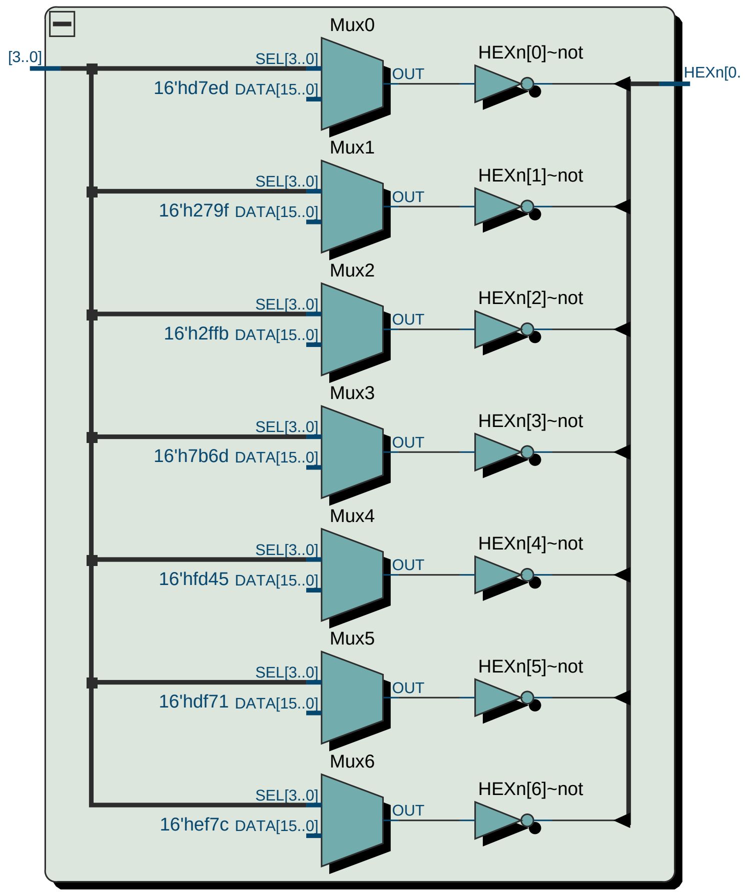
c. Result of RTL viewer



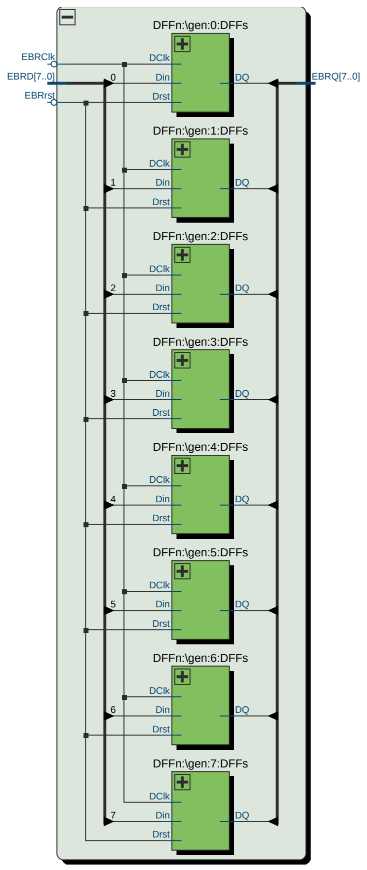
D flip-flop



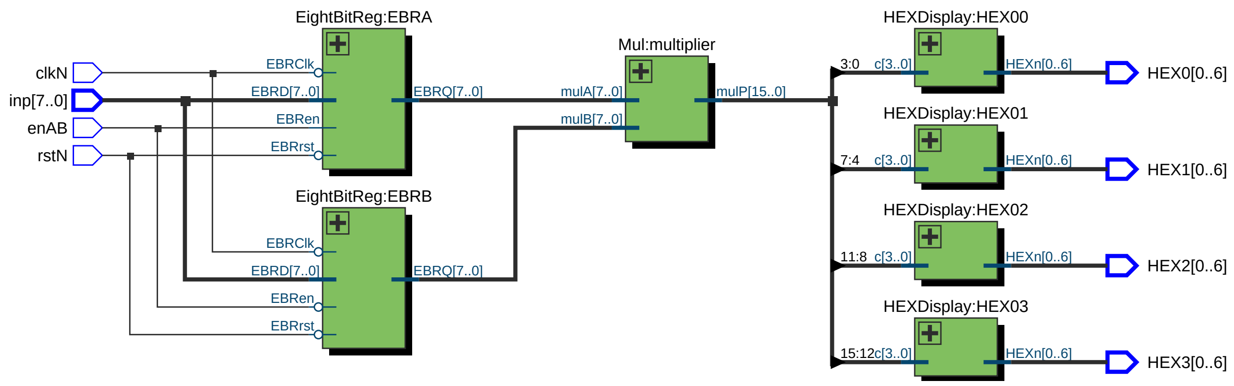
Multiplier



HEX display decoder



8-bit register



Top level

4. Known how to program ALU circuit with registered inputs and outputs.

I changed the circuit a bit in order to work on a limited-input devices such as DE10.

First, change SW[7:0] to the desired number A. Push KEY1 (clk) to store the value into A register.

Then, change SW[7:0] to the desired number B. Push KEY1 (clk) while holding KEY2 to store the value into B register.

Or, you can input B first, then A later.

Lastly, change SW[9:8] to 00, 01, 10 or 11 for $A+B$, $A-B$, $B-A$ or $A\times B$, respectively. The result will show up on HEX[3:0].

An example is shown in the waveform section.

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;
```

Mul.vhd

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

ENTITY Mul IS
        PORT (
                mulA, mulB : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                mulP : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
        );
END Mul;
```

```
ARCHITECTURE arch OF Mul IS
        SIGNAL a_padded, zero : STD_LOGIC_VECTOR(15 DOWNTO 0);
        TYPE ab_t IS ARRAY(0 TO 7) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
        SIGNAL ab : ab_t;
BEGIN

        zero <= "0000000000000000";
        a_padded <= "00000000" & mulA;

        ab(0) <= a_padded WHEN mulB(0) = '1' ELSE
        zero;
        sum : FOR i IN 1 TO 7 GENERATE
                ab(i) <= a_padded(15 - i DOWNTO 0) & zero(i - 1 DOWNTO 0) WHEN mulB(i) = '1' ELSE
                zero;
        END GENERATE;

        mulP <= ab(0) + ab(1) + ab(2) + ab(3) + ab(4) + ab(5) + ab(6) + ab(7);
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
```

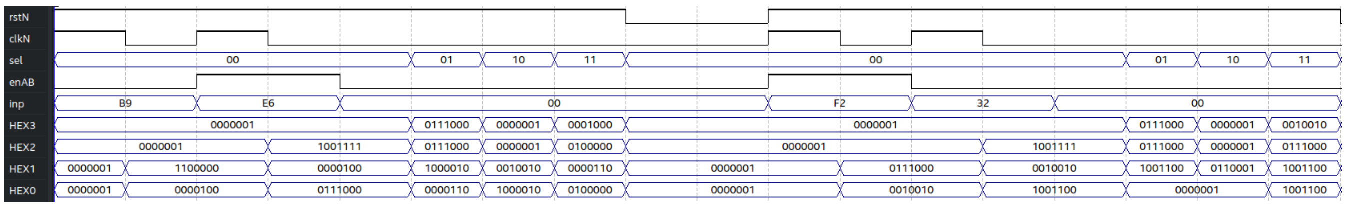
Exc3.vhd

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

ENTITY Exc4 IS
```

```
        PORT (
                inp : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
                enAB, clkN, rstN : IN STD_LOGIC;
                sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0)
        );
END Exc4;

ARCHITECTURE arch OF Exc4 IS
        SIGNAL a, b : STD_LOGIC_VECTOR(7 DOWNTO 0);
        SIGNAL sum, subAB, subBA : STD_LOGIC_VECTOR(15 DOWNTO 0);
        SIGNAL p, result : STD_LOGIC_VECTOR(15 DOWNTO 0);
        SIGNAL clk, rst : STD_LOGIC;

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

        COMPONENT Mul IS
                PORT (
                        mulA, mulB : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        mulP : OUT STD_LOGIC_VECTOR(15 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
        rst <= NOT(rstN);
        clk <= NOT(clkN);
        EBRA : EightBitReg PORT MAP(
                EBRClk => clk, EBRrst => rst,
                EBRen => NOT(enAB), EBRD => inp, EBRQ => a);
        EBRB : EightBitReg PORT MAP(
                EBRClk => clk, EBRrst => rst,
                EBRen => enAB, EBRD => inp, EBRQ => b);

        multiplier : Mul PORT MAP(mulA => a, mulB => b, mulP => p);
        sum <= STD_LOGIC_VECTOR(unsigned("00000000" & a) + unsigned("00000000" & b));
        subAB <= STD_LOGIC_VECTOR(unsigned("00000000" & a) - unsigned("00000000" & b));
        subBA <= STD_LOGIC_VECTOR(unsigned("00000000" & b) - unsigned("00000000" & a));

        WITH sel SELECT
                result <= p WHEN "11",
                sum WHEN "00",
                subAB WHEN "01",
                subBA WHEN "10";
        HEX03 : HEXDisplay PORT MAP(c => result(15 DOWNTO 12), HEXn => HEX3);
        HEX02 : HEXDisplay PORT MAP(c => result(11 DOWNTO 8), HEXn => HEX2);
        HEX01 : HEXDisplay PORT MAP(c => result(7 DOWNTO 4), HEXn => HEX1);
        HEX00 : HEXDisplay PORT MAP(c => result(3 DOWNTO 0), HEXn => HEX0);
END ARCHITECTURE;
```

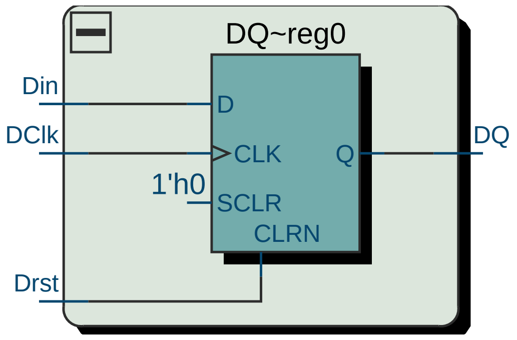
b. Waveform

A	B	Select	Operation	P	7-segment display	HEX output (active low) 3	2	1	0
B9	E6	00	$A + B$	019F	0 1 9 F	0000001	1001111	0000100	0111000
		01	$A - B$	FFD3	F F d 3	0111000	0111000	1000010	0000110
		10	$B - A$	002D	0 0 2 d	0000001	0000001	0010010	1000010
		11	$A \times B$	A636	A 6 3 6	0001000	0100000	0000110	0100000
32	F2	00	$A + B$	0124	0 1 2 4	0000001	1001111	0010010	1001100
		01	$A - B$	FF40	F F 4 0	0111000	0111000	1001100	0000001
		10	$B - A$	00C0	0 0 C 0	0000001	0000001	0110001	0000001
		11	$A \times B$	2F44	2 F 4 4	0010010	0111000	1001100	1001100

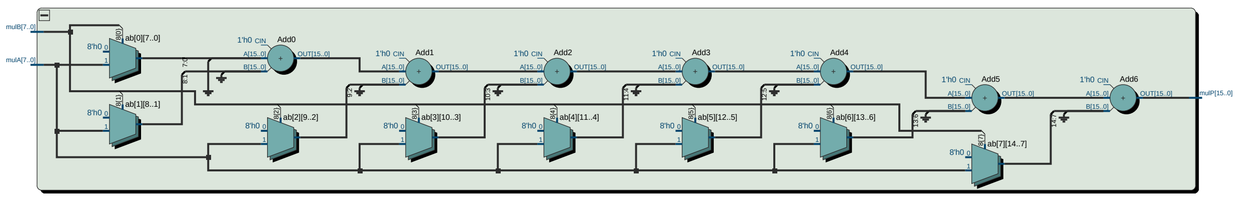


rstN
clkN
sel: 00, 01, 10, 11, 00, 01, 10, 11
enAB
inp: B9, E6, 00, F2, 32, 00
HEX3: 0000001, 0111000, 0000001, 0001000, 0000001, 0111000, 0000001, 0010010
HEX2: 0000001, 1001111, 0111000, 0000001, 0100000, 0000001, 1001111, 0111000, 0000001, 0111000
HEX1: 0000001, 1100000, 0000100, 1000010, 0010010, 0000110, 0000001, 0111000, 0010010, 1001100, 0110001, 1001100
HEX0: 0000001, 0000100, 0111000, 0000110, 1000010, 0100000, 0000001, 0010010, 1001100, 0000001, 1001100

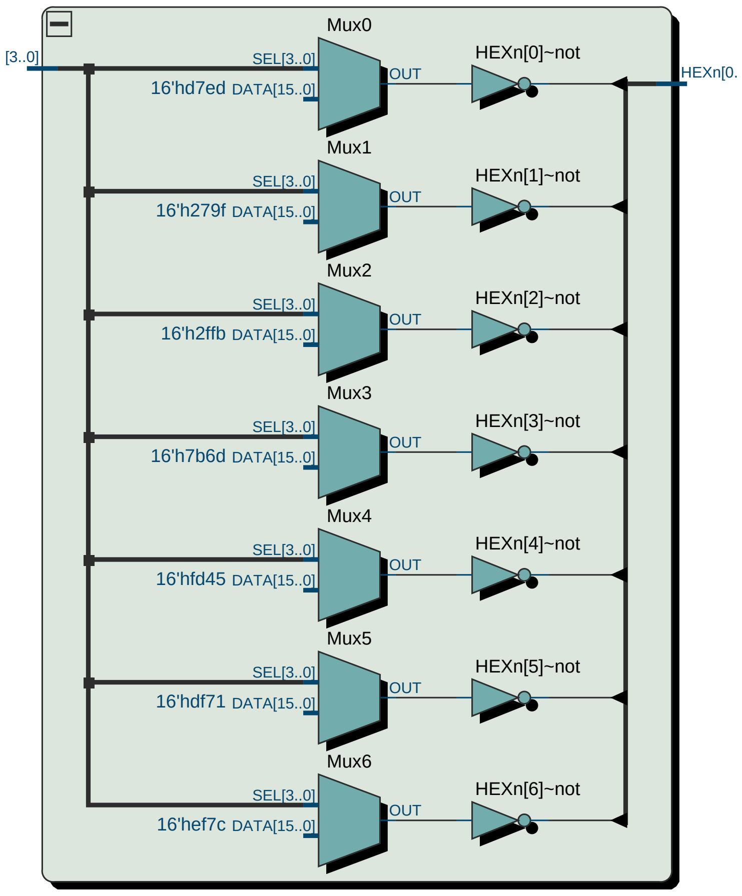
c. Result of RTL viewer



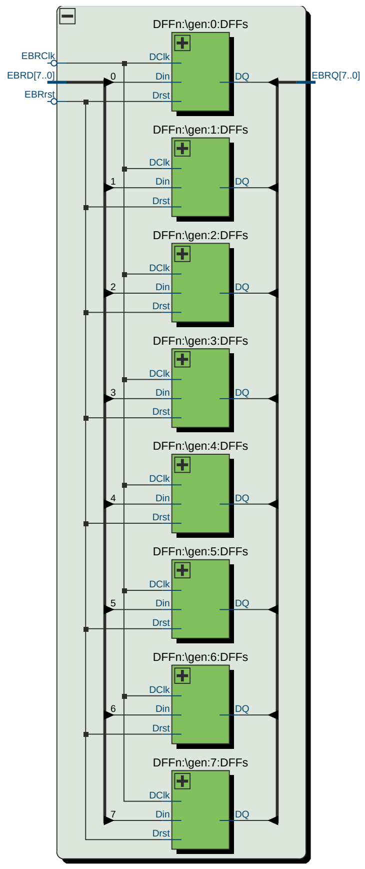
D flip-flop



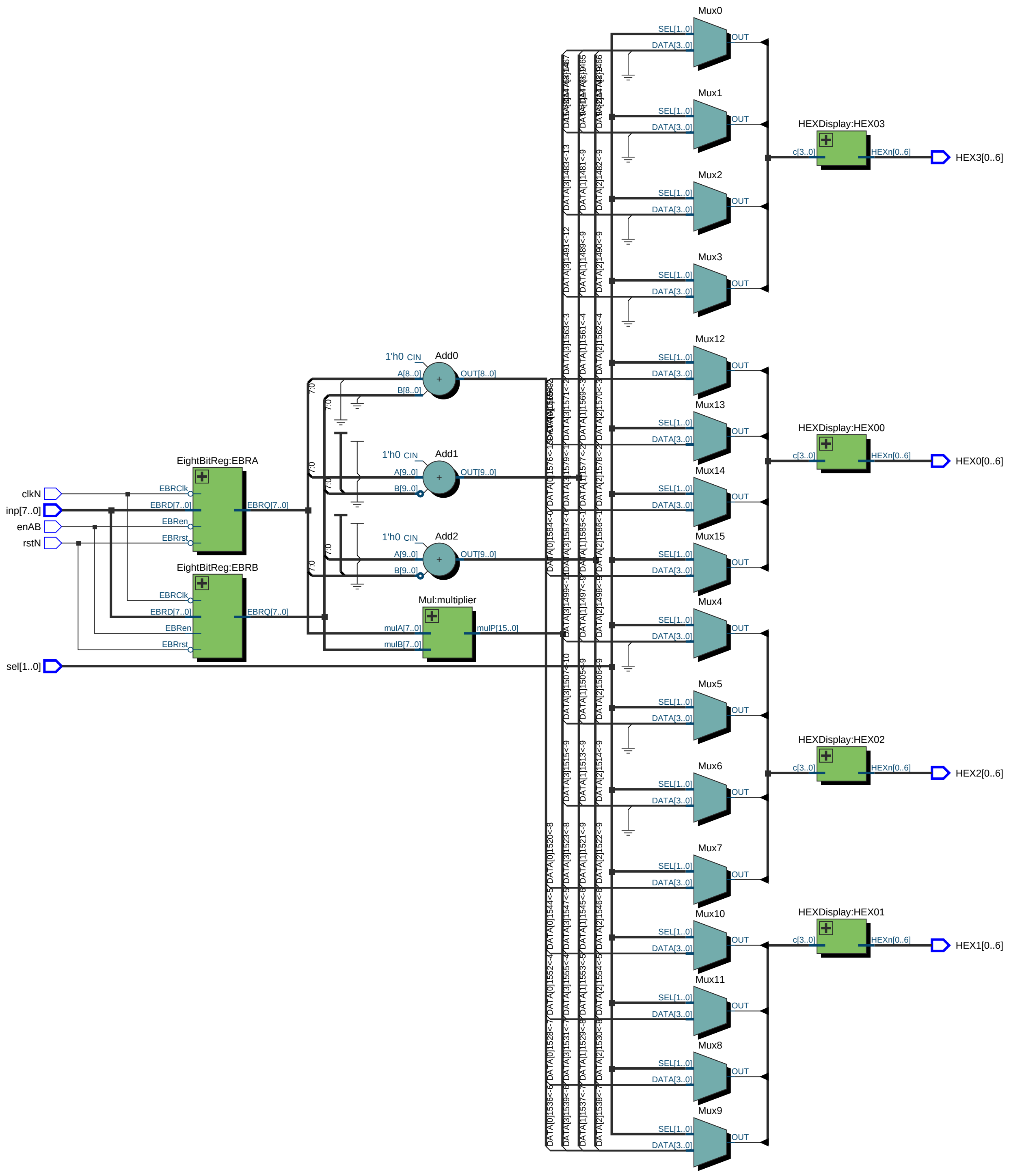
Multiplier



HEX display decoder



8-bit register


Mux0
Mux1
Mux2
Mux3
Mux12
Mux13
Mux14
Mux15
Mux4
Mux5
Mux6
Mux7
Mux10
Mux11
Mux8
Mux9
HEXDisplay:HEX03
HEXDisplay:HEX00
HEXDisplay:HEX02
HEXDisplay:HEX01
HEX3[0..6]
HEX0[0..6]
HEX2[0..6]
HEX1[0..6]
EightBitReg:EBRA
EightBitReg:EBRB
Mul:multiplier
Add0
Add1
Add2
clkN
inp[7..0]
enAB
rstN
sel[1..0]

Top level