

HI4 ハードウェア実験

第 3 回実験

実験者：HI4 45 号 山口惺司
共同実験者：HI4 46 号 吉田旺雅
提出日 2024/08/05

1. 目的

第1回、第2回実験では組合せ回路を作成したが、今回は process 文を用いて順序回路を作成、動作確認をすることで学習する。

2. 実験内容

2.1. 課題1

2.1.1. 課題内容

以下のプログラムを実行、動作確認し、どのような回路か考察しなさい。

2.1.2. プログラム

```
library IEEE;
use IEEE.std_logic_1164.all;

entity kadai3 is
    port(
        D, CLK : in std_logic;
        Q, NQ : out std_logic
    );
end kadai3;

architecture rtl of kadai3 is
begin
    process(CLK)
    begin
        if(CLK'event and CLK = '1') then
            Q <= D;
            NQ <= not D;
        end if;
    end process;
end rtl;
```

2.1.3. ピン設定情報

ピン設定情報を図1に示す。





		Node Name	Direction	Location
1		CLK	Input	PIN_D13
2		D	Input	PIN_G26
3		NQ	Output	PIN_AE23
4		Q	Output	PIN_AF23

図1 課題1 ピン設定情報

2.1.4. 結果

CLK のボタンが押されたときに D が 0 だった時, Q が 0, D が 1 だった時, Q が 1 となった.
NQ は Q と反対の出力がされた.

2.1.5. 考察

process を使って順序回路を実装している.

実行結果からこの回路は D-フリップフロップと推測される.

if 文の中には CLK に変化があり, かつ, CLK が 1 の時という条件が書かれている.

条件に当てはまった時の D の値を Q に入れている.

2.2. 課題 2

2.2.1. 課題内容

課題 1 を参考に JK フリップフロップを構成し, 動作確認しなさい.

2.2.2. プログラム

```
library IEEE;
use IEEE.std_logic_1164.all;

entity kadai42 is
    port (
        J, K, CLK : in std_logic;
        Q : out std_logic
    );
end kadai42;

architecture rtl of kadai42 is
    signal A : std_logic := '0';

begin
    process(CLK)
    begin
        if(CLK'event and CLK = '1') then
            if (j = '1' and k = '0') then
                A <= '1';
            elsif (j = '0' and k = '1') then
                A <= '0';
            elsif (j = '1' and k = '1') then
                A <= not A;
            end if;
        end if;
    end process;
end;
```

```

end if;
Q <= A;
end process;
end rtl;

```

2.2.3. ピン設定情報

ピン設定情報を図 2 に示す.





	Node Name	Direction	Location
	CLK	Input	PIN_N25
	J	Input	PIN_N26
	K	Input	PIN_P25
	Q	Output	PIN_AE23

図 2 課題 2 ピン設定情報

2.2.4. 結果

実行結果を表 1 に示す.

表 1 課題 2-JK-ff 実行結果

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	\bar{Q}

2.2.5. 考察

実行結果から

J=0, K=0 の時, Q を保持(保持状態)

J=0, K=1 の時, Q を 0 にする(リセット状態)

J=1, K=0 の時, Q を 1 にする(セット状態)

J=1, K=1 の時, Q を反転させる(トグル状態)

という風になっていることがわかった.

このことから, たくしく JK-ff の回路が組めていることが分かる.

また, プログラムを書く際に, 最初の変数 A を使用しておらず, Q に直接代入するという書き方をしていたが, $Q \leq \text{not } Q$ という文でエラーが発生した.

これは変数 Q が出力用の変数型をしているためだと考えたため, 別の変数 A を経由して代入するように変更したらうまく動作した.

理由は不明だが entity の名前を jkff にすると正しく動作しないことがあり, 名前を変更することで解決した.

2.3. 課題 3

2.3.1. 課題内容

以下のプログラムを実行，動作確認し，どのような回路か考察しなさい。

2.3.2. プログラム

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity COUNT10 is
    port(RESET,CLK:in std_logic;
          Q :out std_logic_vector(3 downto 0)
    );
end COUNT10;
architecture RTL of COUNT10 is
    signal TQ:std_logic_vector(3 downto 0);
begin
    process (RESET,CLK)begin
        if(RESET='0')then
            TQ <= "0000";
        elsif(CLK'event and CLK='1')then
            if(TQ="1001")then
                TQ <= "0000";
            else
                TQ <= TQ+'1';
            end if;
        end if;
    end process;
    Q <= TQ;
end RTL;
```

2.3.3. ピン設定情報

ピン設定情報を図 3 に示す。







		Node Name	Direction /	Location
1		CLK	Input	PIN_G26
2		RESET	Input	PIN_N23
3		Q[0]	Output	PIN_AE23
4		Q[1]	Output	PIN_AF23
5		Q[2]	Output	PIN_AB21
6		Q[3]	Output	PIN_AC22

図 3 課題 3 ピン設定情報

2.3.4. 結果

RESET に設定されているボタンを押すと, Q=0000

CLK を押すたびに Q の値が 1 ずつ増えていく.

Q が 1001 になると次の Q は 0000 となる.

カウントされている様子を表 2 に示す.

CLK を押した回数の合計	Q[3]	Q[2]	Q[1]	Q[0]
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

2.3.5. 考察

実行結果から, この回路は 10 進カウンタであることがわかる.

RESET ボタンを押すと Q の値が何であろうと 0000 にすることができる.

2.4. 課題 4

2.4.1. 課題内容

以下のプログラムは「ステートマシン」と言われる記述で構成されています.このプログラムを実行, 動作確認し, どのような回路か考察しなさい.

2.4.2. プログラム

```
library IEEE;
use IEEE.std_logic_1164.all;
entity st_man is
    port(RESET,CLK,X:in std_logic;
         Y :out std_logic_vector(1 downto 0));
end st_man;

architecture RTL of st_man is
type STATE is(STATE0,STATE1,STATE2);
signal IS_STATE, NE_STATE:STATE;
begin
    process (RESET,CLK) begin
        if(RESET='0')then
```

```

        IS_STATE <= STATE0;
    elsif(CLK'event and CLK='1')then
        IS_STATE <= NE_STATE;
    end if;
end process;

process (IS_STATE,X) begin
    case IS_STATE is
        when STATE0 =>
            Y <= "00";
            if(X='1')then
                NE_STATE <= STATE1;
            else
                NE_STATE <= STATE0;
            end if;
        when STATE1 =>
            Y <= "01";
            if(X='1')then
                NE_STATE <= STATE2;
            else
                NE_STATE <= STATE1;
            end if;
        when STATE2 =>
            Y <= "10";
            if(X='1')then
                NE_STATE <= STATE0;
            else
                NE_STATE <= STATE2;
            end if;
        end case;
    end process;
end RTL;

```

2.4.3. ピン設定情報

ピン設定情報を図 4 に示す.






		Node Name /	Direction	Location
1		CLK	Input	PIN_N23
2		RESET	Input	PIN_G26
3		X	Input	PIN_N25
4		Y[0]	Output	PIN_AE23
5		Y[1]	Output	PIN_AF23

図 4 課題 4 ピン設定情報

2.4.4. 結果

X に割り当てられているボタンが押されるたびに Y が"00"→"01"→"10"→"00"という風に遷移していった。

2.4.5. 考察

状態遷移の詳細を以下に示す。

STATE0:

出力 Y は"00".

X が'1'の場合、次の状態は STATE1.

X が'0'の場合、次の状態は STATE0.

STATE1:

出力 Y は"01".

X が'1'の場合、次の状態は STATE2.

X が'0'の場合、次の状態は STATE1.

STATE2:

出力 Y は"10".

X が'1'の場合、次の状態は STATE0.

X が'0'の場合、次の状態は STATE2.

このようにステートマシンはある値が変化すると状態が遷移し、その状態の時に任意の動作を行うことができるシステムである。

2.5. 課題 5

2.5.1. 課題内容

4 進カウンタをステートマシン記述で作成し、動作確認しなさい。

2.5.2. プログラム

```
library IEEE;
use IEEE.std_logic_1164.all;

entity QuaternaryCounter is
    port(
        RESET, CLK, X: in std_logic;
        Y : out std_logic_vector(1 downto 0));
end QuaternaryCounter;

architecture rtl of QuaternaryCounter is
```



```

type STATE is (STATE0, STATE1, STATE2, STATE3);
signal IS_STATE, NE_STATE : STATE;

begin
    process(RESET, CLK)
    begin
        if(RESET='0')then
            IS_STATE <= STATE0;
        elsif(CLK'event and CLK='1')then
            IS_STATE <= NE_STATE;
        end if;
    end process;

    process(IS_STATE, X)
    begin
        case IS_STATE is
            when STATE0 =>
                Y <= "00";
                if(X='1')then
                    NE_STATE <= STATE1;
                else
                    NE_STATE <= STATE0;
                end if;
            when STATE1 =>
                Y <= "01";
                if(X='1')then
                    NE_STATE <= STATE2;
                else
                    NE_STATE <= STATE1;
                end if;
            when STATE2 =>
                Y <= "10";
                if(X='1')then
                    NE_STATE <= STATE3;
                else
                    NE_STATE <= STATE2;
                end if;
            when STATE3 =>
                Y <= "11";
                if(X='1')then
                    NE_STATE <= STATE0;
                else

```

```

        NE_STATE <= STATE3;
    end if;

    end case;
end process;
end rtl;

```

2.5.3. ピン設定情報

ピン設定情報を図 5 に示す。






		Node Name #	Direction	Location
1		CLK	Input	PIN_N23
2		RESET	Input	PIN_G26
3		X	Input	PIN_N25
4		Y[0]	Output	PIN_AE23
5		Y[1]	Output	PIN_AF23

図 5 課題 5 ピン設定情報

2.5.4. 結果

X に割り当てられているボタンが押されるたびに Y が”00”→”01”→”10”→”11”→”00”という風に遷移していった。

2.5.5. 考察

課題 4 を参考にして作ったため、コード自体はほとんど変わらず STATE3 の時の動作を追加しただけである。

実行結果もほとんど変わらず課題 4 と違って”11”までカウントできるようになった、実行結果から 4 進カウンタが正しく実装できていることがわかった。

2.6. 課題 6

2.6.1. 課題内容

VHDL 実験ボードの入出力を考慮し、オリジナルの順序回路を構成しなさい。レポートには、回路の仕様とプログラムリスト、ピン設定、実行結果を示すこと。

今回は、JK-ff を用いて同期式 16 進カウンタを作成し、7 セグメントに表示させる回路を構成した。

2.6.2. プログラム

```

library ieee;
use ieee.std_logic_1164.all;

entity kadai46 is
    port (
        CLK1 : in std_logic;
        Z1 : out std_logic_vector(6 downto 0)
    );

```

```

end kadai46;

architecture rtl of kadai46 is
    signal A0, A1, A2, A3 : std_logic := '0';
    signal tmp0, tmp1 : std_logic := '0';

    component kadai42
        port (
            J, K, CLK : in std_logic;
            Q : out std_logic
        );
    end component;

    component segment
        port (
            A: in std_logic_vector(3 downto 0);
            Z: out std_logic_vector(6 downto 0)
        );
    end component;

begin
    jkff1: kadai42 port map(J => '1', K => '1', CLK => CLK1, Q => A0);
    jkff2: kadai42 port map(J => A0, K => A0, CLK => CLK1, Q => A1);
    tmp0 <= A1 and A0;
    jkff3: kadai42 port map(J => tmp0, K => tmp0, CLK => CLK1, Q => A2);
    tmp1 <= tmp0 and A2;
    jkff4: kadai42 port map(J => tmp1, K=> tmp1, CLK => CLK1, Q => A3);

    seg1: segment port map(A(0) => A0, A(1) => A1, A(2) => A2, A(3) => A3, Z =>
Z1);

end rtl;

```

2.6.3. ピン設定情報

ピン設定情報を図 6 に示す。













		Node Name /	Direction	Location
1		CLK1	Input	PIN_G26
2		Q1[0]	Unknown	PIN_AE23
3		Q1[1]	Unknown	PIN_AF23
4		Q1[2]	Unknown	PIN_AB21
5		Q1[3]	Unknown	PIN_AC22
6		Z1[0]	Output	PIN_AF10
7		Z1[1]	Output	PIN_AB12
8		Z1[2]	Output	PIN_AC12
9		Z1[3]	Output	PIN_AD11
10		Z1[4]	Output	PIN_AE11
11		Z1[5]	Output	PIN_V14
12		Z1[6]	Output	PIN_V13

図 6 課題 6 ピン設定情報

2.6.4. 結果

CLK が押されるたびにカウントが進んでいた。

Z1 を 7 セグメントとしてみた時の実行結果を表 3 に示す。

表 3 課題 6 実行結果

CLK を押した回数の合計	Z1
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	b
12	C
13	d
14	E
15	F
16	0

2.6.5. 考察

まず今回作成した同期式 16 進カウンタの回路図を図 7 に示す.

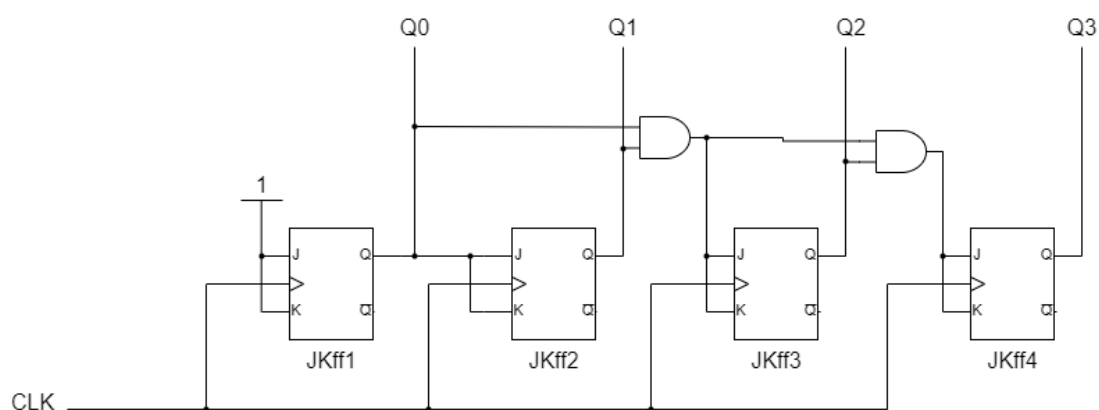


図 7 同期式 16 進カウンタ

課題 2 で作成した JKff の回路と前の実験で作成した 7 セグメントに出力する回路をコンポーネントとして宣言し, 今回の回路を作成した.

実行結果から CLK が押されるたびにカウンタが 1 ずつ進んでいることが分かるため, 回路は正しく実装できていることが分かる.

JKff の回路をコンポーネントとして宣言するときに, どれをどれに代入すればいいかわかりにくく, 苦労した.

3. 感想

今回の実験では process を用いて順序回路を作成したが, 2 年生の時に学習した D-FF や JK-FF などの順序回路が出てきて, どのような回路かが思い出せず調べながらの実験となったため大変だった.

課題 6 では今まで学習したことを利用して回路を作成できたためよかった.