Open-It FPGA トレーニングコース
演習 I1 手順書

内田 智久 (KEK IPNS Esys)

Rev. 1.1
2015 年 12 月 4 日

1　演習内容説明

最初に講師が演習内容について説明しますので、説明を聞き演習で設計する回路の内容について理解してください。何を設計すれば良いのか理解していないと演習を正しく進める事ができません。分からないことがあれば些細な事でも構いませんので質問してください。演習時間になったら手順に従って演習を進めてください。設計やソースコードの記述で調べても分からない時は最後にソースコードを解答例として載せましたので参考にしながら先へ進んでください。

1.1　概要

演習 I1 は演習 S1 の結果を修正する事で進めます。演習 S1 が終わっていない方は終わらせてから演習 I1 を行ってください。

1.2　S1 プロジェクトを開く

講義資料「4.2 論理シミュレーション」を見ながら演習 S1 の Vivado プロジェクトを開いてください。

1.3　モジュール作成

階層化するために下の二つのモジュールを新たに作成します。

- S1_SYNC_COUNTER: 32 ビットカウンター
- S1_ENCODER: 7 セグ LED 符号器

手順 1　講義資料「5 階層構造設計」を見ながら以下の内容で新しいモジュール「S1_SYNC_COUNTER」を作成してください。

- ファイル名
  - File type: Verilog
  - File name: S1_SYNC_COUNTER
  - File lccation: ¡Local to Project¿
- Difine Module（IO ポート）

Port Name	Direction	Bus	MSB	LSB
CLK	input		3	0
RSTn	input		3	0
Q	output	チェックを入れる	31	0

手順 2　以下の内容で新しいモジュール「S1_ENCODER」を作成してください。

- ファイル名
  - File type: Verilog
  - File name: S1_ENCODER
  - File lccation: ¡Local to Project¿
- Difine Module（IO ポート）

Port Name	Direction	Bus	MSB	LSB
I	input	チェックを入れる	3	0
CA	output			
CB	output			
CC	output			
CD	output			
CE	output			
CF	output			
CG	output			

手順 3　Project Manager の Source 窓の Hierachy 画面で 2 つの新しいモジュールが追加された事を確認してください。

手順 4　次に S1.v のカウンターとエンコーダ記述を「S1_SYNC_COUNTER.v」「S1_ENCODER.v」へコピペして移動させます。

手順 5　Source 窓から「TEST.v」をダブルクリックして開いてください。

手順 6　以下の部分をコピーし、「S1_SYNC_COUNTER.v」の endmodule の上にペーストしてください。

```
reg [27:0]  sync_counter;

always @(posedge OSC or negedge RST_SWn)begin
    if(!RST_SWn)begin
        sync_counter[27:0] <= 28'd0;
    end else begin
        sync_counter[27:0] <= sync_counter[27:0] + 28'd1;
    end
end
```

手順 7　28 ビットカウンターのビット数を増やして 32 ビットカウンターに変更してください。

手順 8　モジュールポート（CLK, RSTn, Q）に合わせてコードを修正してください（講義資料「5 階層構造設計」参照）。

手順 9　修正が終わったら保存してください。

手順 10　以下の部分をコピーし、「S1_ENCODER.v」の endmodule の上にペーストしてください。

```
assign  CA =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                ( I[3] & ~I[2] &  I[1] &  I[0]) |
                ( I[3] &  I[2] & ~I[1] &  I[0]) ;

assign  CB =    (~I[3] &  I[2] & ~I[1] &  I[0]) |
                (~I[3] &  I[2] &  I[1] & ~I[0]) |
                ( I[3] & ~I[2] &  I[1] &  I[0]) |
                ( I[3] &  I[2] & ~I[1] & ~I[0]) |
                ( I[3] &  I[2] &  I[1] & ~I[0]) |
                ( I[3] &  I[2] &  I[1] &  I[0]) ;
```

```
assign  CC =    (~I[3] & ~I[2] &  I[1] & ~I[0]) |
                ( I[3] &  I[2] & ~I[1] & ~I[0]) |
                ( I[3] &  I[2] &  I[1] & ~I[0]) |
                ( I[3] &  I[2] &  I[1] &  I[0]) ;

assign  CD =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                (~I[3] &  I[2] &  I[1] &  I[0]) |
                ( I[3] & ~I[2] &  I[1] & ~I[0]) |
                ( I[3] &  I[2] &  I[1] &  I[0]) ;

assign  CE =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                (~I[3] & ~I[2] &  I[1] &  I[0]) |
                (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                (~I[3] &  I[2] & ~I[1] &  I[0]) |
                (~I[3] &  I[2] &  I[1] &  I[0]) |
                ( I[3] & ~I[2] & ~I[1] &  I[0]) ;
assign  CF =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                (~I[3] & ~I[2] &  I[1] & ~I[0]) |
                (~I[3] & ~I[2] &  I[1] &  I[0]) |
                (~I[3] &  I[2] &  I[1] &  I[0]) |
                ( I[3] &  I[2] & ~I[1] &  I[0]) ;
assign  CG =    (~I[3] & ~I[2] & ~I[1] & ~I[0]) |
                (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                (~I[3] &  I[2] &  I[1] &  I[0]) |
                ( I[3] &  I[2] & ~I[1] & ~I[0]) ;
```

手順 11　修正が終わったら保存してください。

手順 12　「S1.v」のポートリストから endmodule の間は AN に関する記述以外は全て削除し、下のようにしてください。

```
module S1(
    input OSC,
    input RST_SWn,
    output [7:0] AN,
    output CA,
    output CB,
    output CC,
    output CD,
    output CE,
    output CF,
    output CG
    );

    assign  AN[7:0] = 8'b1111_1110;

endmodule
```

手順 13　S1.v に S1_SYNC_COUNTER モジュールと S1_ENCODER モジュールを組み込み、元の S1 と同等の回路になるように記述してください（講義資料「5 階層構造設計」参照)。インスタンス名は自由に名付けてください。思いつかない時は、S1_SYNC_COUNTER モジュールのインスタンス名は U1、S1_ENCODER モジュールは U2 としてください。

手順 14　修正が終わったら保存してください。

手順 15　エラーが無い事を確認してください。

手順 16　「Source」画面で S1 モジュールの下に新しいモジュールが組み込まれていることを確認してください。

以上でコード作成は終了です。

1.4　論理合成から実機確認

データを生成し FPGA にデータをダウンロードして動作確認してください。演習 S1 の時と同じように動作するか確認してください。ここでは、シミュレーションは省略しても構いません。興味がある方はもちろんミュレーションしていただいても構いません。

2　解答例

以下の回答は一つの例です。皆さんが設計した回路や記述と異なっているかもしれません。異なっていてもシミュレーションと実機でも正しく動作しているのなら、それも正解です。以下の例は参考程度に見てください。

2.1　Verilog ソースコード

2.1.1　S1.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2015/06/25 10:45:42
// Design Name:
// Module Name: S1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module S1(
    input OSC,
    input RST_SWn,
    output [7:0] AN,
    output CA,
    output CB,
    output CC,
    output CD,
    output CE,
    output CF,
    output CG
    );

    assign  AN[7:0] = 8’b1111_1110;
```

```
    wire    [31:0]  Q;

    S1_SYNC_COUNTER U1(
        .CLK    (OSC),
        .RSTn   (RST_SWn),
        .Q      (Q[31:0])
    );

    S1_ENCODER U2(
        .I  (Q[27:24]),
        .CA (CA),
        .CB (CB),
        .CC (CC),
        .CD (CD),
        .CE (CE),
        .CF (CF),
        .CG (CG)
    );

endmodule
```

2.1.2 S1_SYNC_COUNTER.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2015/06/26 14:01:38
// Design Name:
// Module Name: S1_SYNC_COUNTER
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module S1_SYNC_COUNTER(
    input CLK,
    input RSTn,
    output [31:0] Q
    );

    reg [31:0]  sync_counter;

    always @(posedge CLK or negedge RSTn)begin
        if(!RSTn)begin
            sync_counter[31:0] <= 32’d0;
        end else begin
            sync_counter[31:0] <= sync_counter[31:0] + 32’d1;
        end
    end

    assign Q[31:0] = sync_counter[31:0];

endmodule
```

2.1.3 S1_ENCODER.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2015/06/26 14:06:58
// Design Name:
// Module Name: S1_ENCODER
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module S1_ENCODER(
    input [3:0] I,
    output CA,
    output CB,
    output CC,
    output CD,
    output CE,
    output CF,
    output CG
    );

assign  CA =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                    (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                    ( I[3] & ~I[2] &  I[1] &  I[0]) |
                    ( I[3] &  I[2] & ~I[1] &  I[0]) ;

    assign  CB =    (~I[3] &  I[2] & ~I[1] &  I[0]) |
                    (~I[3] &  I[2] &  I[1] & ~I[0]) |
                    ( I[3] & ~I[2] &  I[1] &  I[0]) |
                    ( I[3] &  I[2] & ~I[1] & ~I[0]) |
                    ( I[3] &  I[2] &  I[1] & ~I[0]) |
                    ( I[3] &  I[2] &  I[1] &  I[0]) ;

    assign  CC =    (~I[3] & ~I[2] &  I[1] & ~I[0]) |
```

```
                    ( I[3] &  I[2] & ~I[1] & ~I[0]) |
                    ( I[3] &  I[2] &  I[1] & ~I[0]) |
                    ( I[3] &  I[2] &  I[1] &  I[0]) ;

    assign  CD =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                    (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                    (~I[3] &  I[2] &  I[1] &  I[0]) |
                    ( I[3] & ~I[2] &  I[1] & ~I[0]) |
                    ( I[3] &  I[2] &  I[1] &  I[0]) ;

    assign  CE =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                    (~I[3] & ~I[2] &  I[1] &  I[0]) |
                    (~I[3] &  I[2] & ~I[1] & ~I[0]) |
                    (~I[3] &  I[2] & ~I[1] &  I[0]) |
                    (~I[3] &  I[2] &  I[1] &  I[0]) |
                    ( I[3] & ~I[2] & ~I[1] &  I[0]) ;
    assign  CF =    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                    (~I[3] & ~I[2] &  I[1] & ~I[0]) |
                    (~I[3] & ~I[2] &  I[1] &  I[0]) |
                    (~I[3] &  I[2] &  I[1] &  I[0]) |
                    ( I[3] &  I[2] & ~I[1] &  I[0]) ;
    assign  CG =    (~I[3] & ~I[2] & ~I[1] & ~I[0]) |
                    (~I[3] & ~I[2] & ~I[1] &  I[0]) |
                    (~I[3] &  I[2] &  I[1] &  I[0]) |
                    ( I[3] &  I[2] & ~I[1] & ~I[0]) ;

endmodule
```