

Open-It FPGA training course
Practice I1 manual

Tomohisa Uchida (KEK IPNS Esys)

Rev. 1.2
2016/09/30

Translated by Yun-Tsung Lai (KEK IPNS Esys)
2024/06/20

1. Explanation on the content of C1

First, the content will be explained during the lecture, so please listen to the explanation and understand the content of the circuit for the exercise. Please feel free to ask any questions, even if they are trivial. Otherwise, you cannot proceed to design if you don't understand what to do. The source codes as an example answer are included at the end. You can refer to it if you don't understand the content.

1.1 Introduction

It is based on your result on S1, so please finish S1 first to proceed.

1.2 Open the project of S1

Refer to the lecture material "4.2 Logic simulation" to open the Vivado project of S1..

1.3 Create module

To make a hierarchy, make these two modules below:

- S1_SYNC_COUNTER: 32-bit counter
- S1_ENCODER: Encoder for 7-segment LED

Step 1 Refer to the lecture material "6 Hierarchical structure design" to make a new module "S1 SYNC COUNTER" based on the following content.

- File name
 - File type: Verilog
 - File name: S1_SYNC_COUNTER
 - File location: Local to project
- Define module (I/O port)

Port Name	Direction	Bus	MSB	LSB
CLK	input			
RSTn	input			
Q	output	Check	31	0

Step 2 Make a new module "S1_ENCODER" based on the following content.

- File name
 - File type: Verilog
 - File name: S1_ENCODER
 - File location: Local to project

Define module (I/O port)

Port Name	Direction	Bus	MSB	LSB
I	input	Check	3	0
CA	output			
CB	output			
CC	output			
CD	output			

CE	output			
CF	output			
CG	output			

Step 3 Confirm that these two new modules have been added in the Hierarchy screen of the Project Manager's Source window.

Step 4 Move the content of the counter and the encode in S1.v to the new modules "S1_SYNC_COUNTER.v" and "S1_ENCODER.v" by copying and pasting.

Step 5 Open "TEST.v" by double-clicking it from the Source window.

Step 6 Copy the following part, and paste at the "S1_SYNC_COUNTER.v" above the line of endmodule.

```
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
```

Step 7 Increase the bit width from 28 to 32 for the counter.

Step 8 Modify the codes for the names of the ports: CLK, RSTn, Q. Please refer to the lecture material "6 Hierarchical structure design".

Step 9 Save the files when you finish them.

Step 10 Copy the following part, and paste at the "S1_ENCODER.v" above the line of endmodule.

```
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] ) ;

```

Step 11 Save the files when you finish them.

Step 12 In "S1.v", remove all the main code except for the port parts, and copy the following codes and paste.

```

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
```

Step 13 Incorporate the S1_SYNC_COUNTER module and the S1_ENCODER module into S1.v, and write the circuit to make it equivalent to the original S1 (refer to the lecture material "5 Hierarchical Structure Design"). Feel free to name the instance as you like. You can simply use the instance name U1 for the S1_SYNC_COUNTER module and U2 for the S1_ENCODER module.

Step 14 Save the files when you finish them.

Step 15 Check if there is no error.

Step 16 Check the Source window to see if the two modules are incorporated under S1 or not.

Up to here, all the codes are ready.

1.4 Synthesis to implementation

Generate the data and test it in FPGA. The operation should be the same as the one from S1. It is fine to ignore simulation for now. If you are interested in it, you can also do it.

2. Example of answers

The answers below are just one of the example. It might be different from yours. Please just refer to them. If yours can be working in simulation and FPGA, it is no problem.

2.1 Verilog source codes

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
```
