

Open-It FPGA training course
Practice C1 manual (Nexys-4 board)

Tomohisa Uchida (KEK IPNS Esys)

Rev. 3.1
2017/08/09

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

1. Explanation on the content of S1

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.

2. Procedures of C1

When it is time for the exercise, please proceed by following the steps below. If you have any questions or concerns, please discuss with your neighbors. Sometimes the content of the material might be wrong, so feel free to ask the instructor for any questions.

The example source codes are also put below. You can refer to them to proceed.

2.1 Circuit design

We have already practiced the counter circuit, so no circuit design. Lecture material. Create the source code by merging the counter described in the lecture material "4.2 Logical Simulation" and the contents of C1 to proceed.

2.2 Create Vivado project

Refer to the lecture material "2.2 HDL input and RTL analysis" to make new Verilog sources. Use the following setup when you are creating the project:

- Project name
 - Project name: S1
 - Project location: C:\Temp\FPGA Seminar
- Device selection
 - xc7a100tcsq324-1

If the screen of the project is shown, proceed to the next step.

2.3 Create new source code

Refer to the lecture material "2.2 HDL input and RTL analysis" to make new source code.

- Items to setup
 - File type: Verilog
 - File name: S1
 - Module name: S1

Define the I/O ports based on the following contents. We will repeat the it, but please enter it again for practice.

Port name	Direction	Bus	MSB	LSB
OSC	input			
RST_SWn	input			
AN	output	check	7	0
CA	output			

CB	output			
CC	output			
CD	output			
CE	output			
CF	output			
CG	output			

2.4 Create code and RTL analysis

Step 1 Refer to the lecture material "4.2 Logic simulation" to make new S1 module, and make a 28-bit counter inside it. It is the same design as the one in practice, just the width is different.

- Clock is the OSC input.
- Reset is the RST_SWn input. It is effective when 0.
- Counter signal name is sync_counter.

Step 2 Incorporate the source code from C1. Proceed by following the steps:

- Connect the input I of the circuit created in C1 to the counter signal "sync_counter".
 - Declare internal signal I using wire statement.
 - Connect I[3:0] and sync counter[27:24] using assign statement

Step 3 Copy a part of the code created from C1.

- From the upper menu of Vivado, File → Open File.
- Select the file: C:\Temp\FPGA Seminar\C1.srcs\sources_1\new\C1.v
- C1.v is shown. Copy the code from the end of the port list up to the endmodule line.
- Move to S1.v. Paste the part at the part above endmodule.
- Close C1.v (nothing is changed in it)

Up to now, the preparation of codes is done. Move on to RTL analysis.

2.5 Logic simulation

Refer to the lecture material "4.2 Logic simulation" to make a test bench for the S1 module, and perform simulation.

- Info of test bench:
 - File type: Verilog
 - File name: S1_TB
 - Module name: S1_TB

Similar to LED15 in the previous exercise, it is time consuming and difficult to simulate until I changes. As in the previous exercise, confirm if the lower bits of the sync counter changes in the simulation. Check that there are no signals with blue waveforms in the simulation. Also, if the line remains red for a while after the reset is released, there is a problem. If you find something like this, please review the source code and correct it.

Once you have confirmed that it is working correctly, proceed to the next step.

2.6 Check the operation from synthesis

Step 1 Refer to the lecture manual "4.3 Implementation on FPGA" to do synthesis, and check the produced circuit diagram (RTL analysis).

Step 2 Please assign the pints as below while referring to the lecture materials. The constraint file name is "S1".

Name	Package Pin	I/O Std.	Drive Str.	Off Chip termination	Pull type	Stew
OSC	E3	LVCMOS33		NONE	NONE	
RST_SWn	C12	LVCMOS33		NONE	NONE	
AN[0]	N6	LVCMOS33	12	NONE	NONE	Slow
AN[1]	M6	LVCMOS33	12	NONE	NONE	Slow
AN[2]	M3	LVCMOS33	12	NONE	NONE	Slow
AN[3]	N5	LVCMOS33	12	NONE	NONE	Slow
AN[4]	N2	LVCMOS33	12	NONE	NONE	Slow
AN[5]	N4	LVCMOS33	12	NONE	NONE	Slow
AN[6]	L1	LVCMOS33	12	NONE	NONE	Slow
AN[7]	M1	LVCMOS33	12	NONE	NONE	Slow
CA	L3	LVCMOS33	12	NONE	NONE	Slow
CB	N1	LVCMOS33	12	NONE	NONE	Slow
CC	L5	LVCMOS33	12	NONE	NONE	Slow
CD	L4	LVCMOS33	12	NONE	NONE	Slow
CE	K3	LVCMOS33	12	NONE	NONE	Slow
CF	M2	LVCMOS33	12	NONE	NONE	Slow
CG	L6	LVCMOS33	12	NONE	NONE	Slow

Step 3 Do synthesis again.

Step 4 Proceed place & route, and data generation.

Step 5 Download the data file to FPGA, press the reset button, and see the operation result.

3. 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.

3.1 Verilog source code

```
'timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2015/06/23 12:00:24
// Design Name:
// Module Name: C1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

module C1(
    input OSC,
    input RST_SWn,
    output [7:0] AN,
    output CA,
    output CB,
    output CC,
    output CD,
    output CE,
    output CF,
    output CG
);

reg [27:0] sync_counter;

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

```

    end else begin
        sync_counter[27:0] <= sync_counter[27:0] + 28'd1;
    end
end

```

```

wire [3:0] I;

```

```

assign I[3:0] = sync_counter[27:24];

```

```

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

```

```

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
```

3.3 Test bench code

```
'timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2015/06/25 11:20:39
// Design Name:
// Module Name: S1_TB
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

module S1_TB;
reg CLK100M;
reg RST_SWn;
wire [7:0] AN;
wire CA;
wire CB;
wire CC;
wire CD;
wire CE;
wire CF;
wire CG;

// You can also write in this way:
// wire CA, CB, CC, CD, CE, CF, CG;

S1 uut(
    .OSC(CLK100M),
    .RST_SWn(RST_SWn),
    .AN(AN),
    .CA(CA),
    .CB(CB),
    .CC(CC),
```

```
        .CD(CD),
        .CE(CE),
        .CF(CF),
        .CG(CG)
);

parameter PERIOD = 10;

always begin
    CLK100M = 1'b0;
    #(PERIOD/2);
    CLK100M = 1'b1;
    #(PERIOD/2);
end

initial begin
    RST_SWn = 1'b0;
    #700 RST_SWn = 1'b1;
end

endmodule
```

3.4 Constraint file (XDC file)

The order of lines in XDC file does not matter.

```
set_property PACKAGE_PIN N6 [get_ports {AN[0]}]
set_property PACKAGE_PIN M6 [get_ports {AN[1]}]
set_property PACKAGE_PIN M3 [get_ports {AN[2]}]
set_property PACKAGE_PIN N5 [get_ports {AN[3]}]
set_property PACKAGE_PIN N2 [get_ports {AN[4]}]
set_property PACKAGE_PIN N4 [get_ports {AN[5]}]
set_property PACKAGE_PIN L1 [get_ports {AN[6]}]
set_property PACKAGE_PIN M1 [get_ports {AN[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property PACKAGE_PIN L3 [get_ports CA]
set_property PACKAGE_PIN N1 [get_ports CB]
set_property PACKAGE_PIN L5 [get_ports CC]
set_property PACKAGE_PIN L4 [get_ports CD]
set_property PACKAGE_PIN K3 [get_ports CE]
set_property PACKAGE_PIN M2 [get_ports CF]
set_property PACKAGE_PIN L6 [get_ports CG]
set_property PACKAGE_PIN E3 [get_ports OSC]
set_property PACKAGE_PIN C12 [get_ports RST_Sw]
set_property IOSTANDARD LVCMOS33 [get_ports CA]
set_property IOSTANDARD LVCMOS33 [get_ports CB]
set_property IOSTANDARD LVCMOS33 [get_ports CC]
set_property IOSTANDARD LVCMOS33 [get_ports CD]
set_property IOSTANDARD LVCMOS33 [get_ports CE]
set_property IOSTANDARD LVCMOS33 [get_ports CF]
set_property IOSTANDARD LVCMOS33 [get_ports CG]
set_property IOSTANDARD LVCMOS33 [get_ports RST_Sw]
set_property IOSTANDARD LVCMOS33 [get_ports OSC]
set_property OFFCHIP_TERM NONE [get_ports AN[0]]
set_property OFFCHIP_TERM NONE [get_ports AN[1]]
set_property OFFCHIP_TERM NONE [get_ports AN[2]]
set_property OFFCHIP_TERM NONE [get_ports AN[3]]
set_property OFFCHIP_TERM NONE [get_ports AN[4]]
set_property OFFCHIP_TERM NONE [get_ports AN[5]]
set_property OFFCHIP_TERM NONE [get_ports AN[6]]
set_property OFFCHIP_TERM NONE [get_ports AN[7]]
create_clock -period 10.000 -name OSC -waveform {0.000 5.000} [get_ports OSC]
```
