EasySoC-Z7010 FPGA/SoC 설계 플랫폼

3. FPGA를 이용한 LED, 7-Segment 제어 실습

개요

- 하드웨어 기술 언어인 HDL을 이용한 로직 설계에 대한 첫 예제.
- 설계에서 검증까지의 순서와 개발환경 및 소프트웨어 사용법 숙지.
- LED와 7-Segment를 이용한 시계 설계.
  - ✓ EasySoC-Z7010를 이용하여 FPGA를 동작시키는 방법 이해.
  - ✓ Xilinx Vivado 소프트웨어 사용법 이해.

HUINS

LED와 7-segment 구성 및 동작 설명

- LED : Light Emitting Diode or Luminescent Diode.
  - 첨가되는 불순물의 종류에 따라 색에 차이를 보임.
  - 기종 광원에 비해 소형, 긴 수명, 전력 소모가 적음.
  - 전류 제한용 저항 값에 의해 밝기 조절이 가능함.

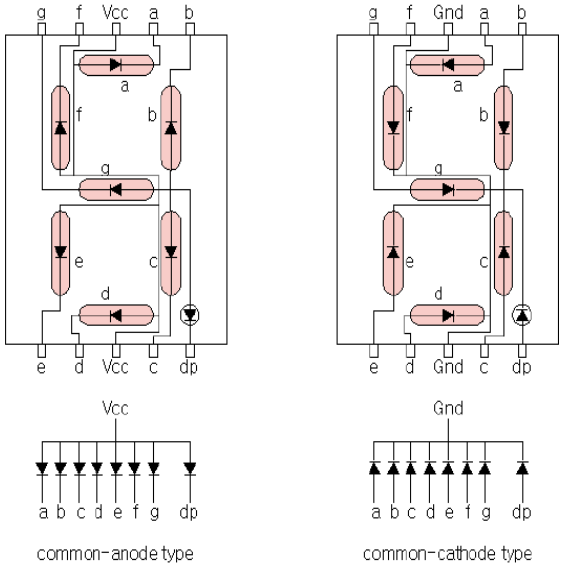
구 분	최소전압	최대전압	전류(일반)	전류(최대)
Red	1.8V	2.3V	20 mA	50 mA
Real Yellow	2.0V	2.8V	20 mA	50 mA
Real Green	3.0V	3.6V	20 mA	50 mA
Real Blue	3.4V	3.8V	20 mA	50 mA
White	3.4V	4.0V	20 mA	50 mA

  - EasySoC-Z7010의 Pmod에는 총 8개의 LED가 제공됨.
    - ✓ 각각의 LED는 Xilinx Zynq PL 영역의 I/O 핀에 연결되어 있음.
    - ✓ '1'(High)을 출력하면 LED가 켜지며, '0'(Low)를 출력하면 LED가 꺼짐.

HUINS

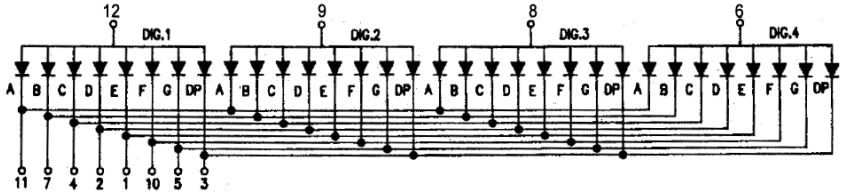
LED와 7-segment 구성 및 동작 설명

- 7-Segment : A-G와 DP의 8개의 LED로 구성.
  - FND(Flexible Numeric Display)의 최소한의 구성.
  - 숫자 뿐만 아니라 A-F의 영문도 표현 가능하다.
    - ✓ BCD나 16진수의 표현에 많이 사용됨.
  - LED의 연결 상태에 따라 두 가지 구조인 Anode형과 Cathode 형으로 나뉨.

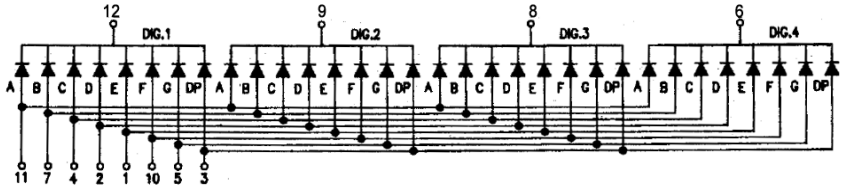


LED와 7-segment 구성 및 동작 설명

- A(Anode)형
  - ✓ COM 핀에 VCC를 연결하고 입력으로 논리적 '0'이 들어오면 동작.



- K(Cathode)형
  - ✓ COM 핀에 GND를 연결하고 입력으로 논리적 '1'이 들어오면 동작.



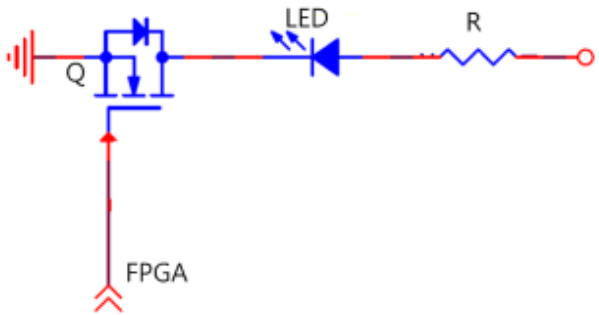
- Cathode 형 기준 데이터 값

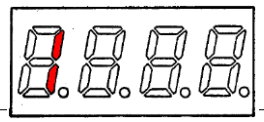
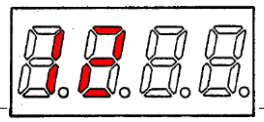
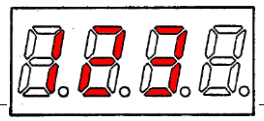
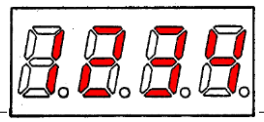
DATA 설정 값																
숫자	A	B	C	D	E	F	G	DP	숫자	A	B	C	D	E	F	G	DP
0	1	1	1	1	1	1	0	0	6	1	0	1	1	1	1	1	0
1	0	1	1	0	0	0	0	0	7	1	1	1	0	0	1	0	0
2	1	1	0	1	1	0	1	0	8	1	1	1	1	1	1	1	0
3	1	1	1	1	0	0	1	0	9	1	1	1	1	0	1	1	0
4	0	1	1	0	0	1	1	0	.	0	0	0	0	0	0	0	1
5	1	0	1	1	0	1	1	0									

HUINS

LED와 7-segment 동작 방법

- LED는 PL영역의 I/O 핀에 연결되어 있으며, Digital High가 입력될 경우 켜지고, Digital Low가 입력될 때 꺼지도록 설계되어 있다.
- 7-Segment를 순차적(DIG1→DIG2→DIG3→DIG4→DIG1)으로 반복해 켤 경우 잔상이 생겨 한꺼번에 디스플레이가 되는 것과 같이 보인다.



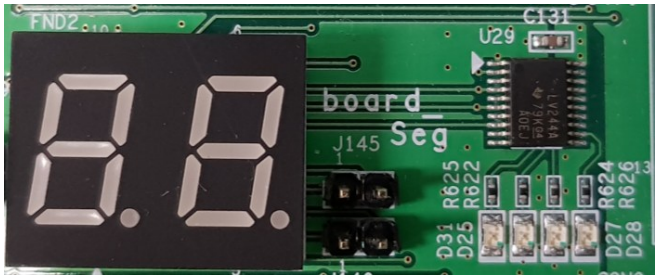
순서	숫자	DATA 설정 값								COM 설정 값			
		A	B	C	D	E	F	G	DP	DIG 1	DIG 2	DIG 3	DIG 4
1		0	1	1	0	0	0	0	0	0	1	1	1
2		1	1	0	1	1	0	1	0	1	0	1	1
3		1	1	1	1	0	0	1	0	1	1	0	1
4		0	1	1	0	0	1	1	0	1	1	1	0

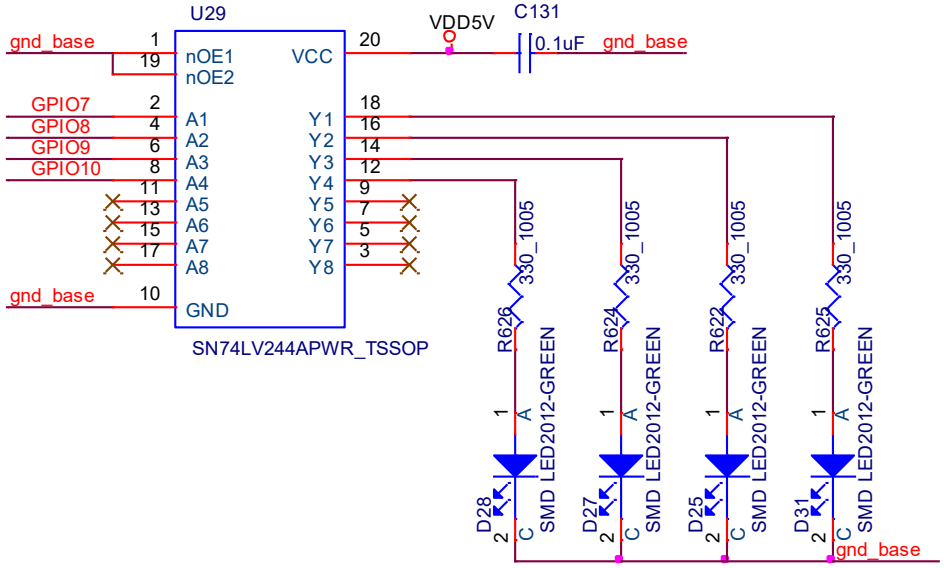
HUINS

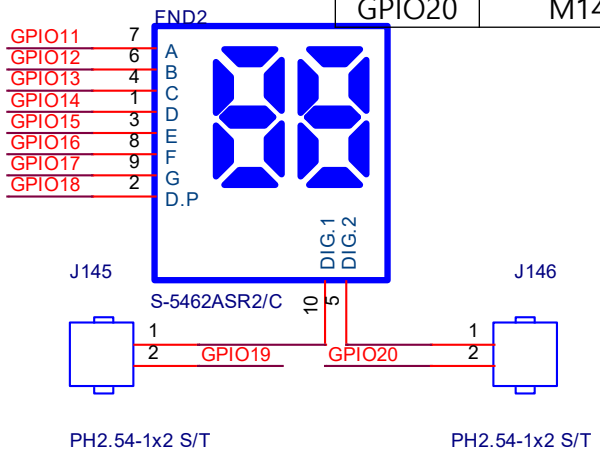
LED, 7-segment 회로도(Base Board)

- 베이스보드에 4개의 LED와 2개의 7-Segment가 존재한다.
- J145와 J146 두 개의 점퍼를 쇼트시켜 작동한다.

GPIO	Package Pin	Base Board
GPIO7	L14	LED_base
GPIO8	L15	LED_base
GPIO9	J13	LED_base
GPIO10	H14	LED_base
GPIO11	J15	7-seg[A]
GPIO12	M15	7-seg[B]
GPIO13	R13	7-seg[C]
GPIO14	M12	7-seg[D]
GPIO15	N13	7-seg[E]
GPIO16	L13	7-seg[F]
GPIO17	G11	7-seg[G]
GPIO18	H11	7-seg[D.P]
GPIO19	R12	7-seg[DIG1]
GPIO20	M14	7-seg[DIG2]



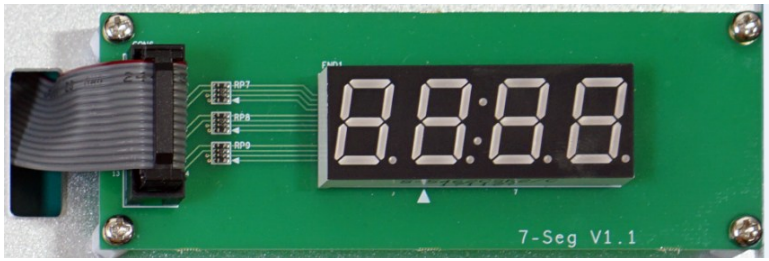


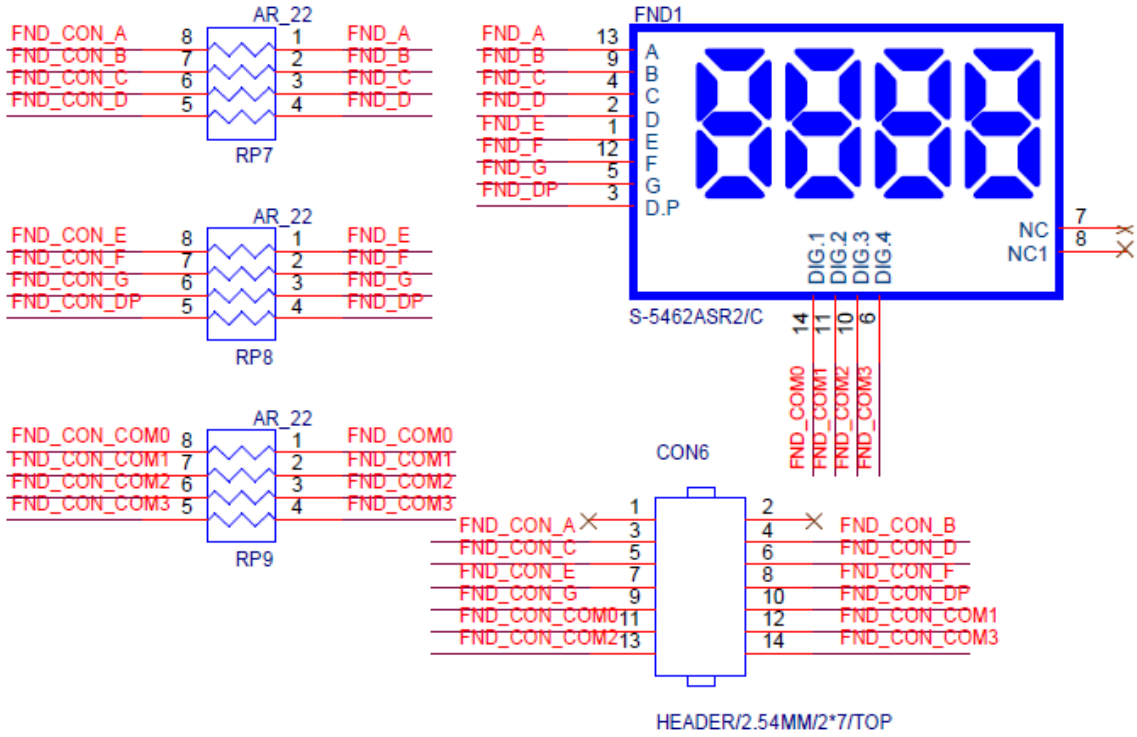


HUINS

7-segment 회로도(Ext. Module)

- Extension Module로 4개의 7-Segment를 확장하여 사용할 수 있다.



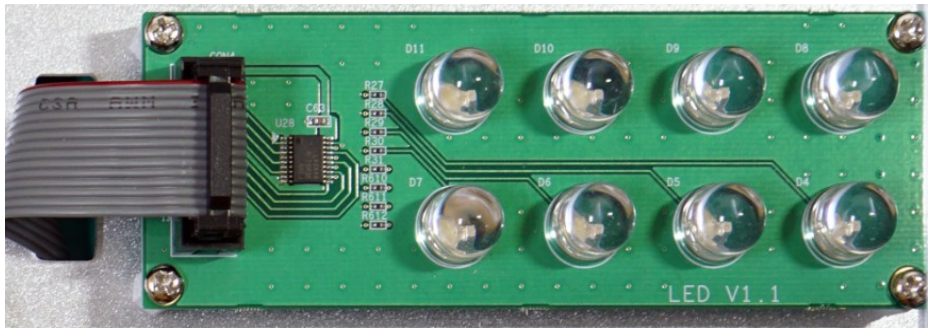


GPIO	Package Pin	External Module
GPIO2	K15	CLK
GPIO3	J14	7-seg[A]
GPIO4	H12	7-seg[B]
GPIO5	N14	7-seg[C]
GPIO6	R15	7-seg[D]
GPIO7	L14	7-seg[E]
GPIO8	L15	7-seg[F]
GPIO9	J13	7-seg[G]
GPIO10	H14	7-seg[D.P]
GPIO11	J15	7-seg[DIG1]
GPIO12	M15	7-seg[DIG2]
GPIO13	R13	7-seg[DIG3]
GPIO14	M12	7-seg[DIG4]

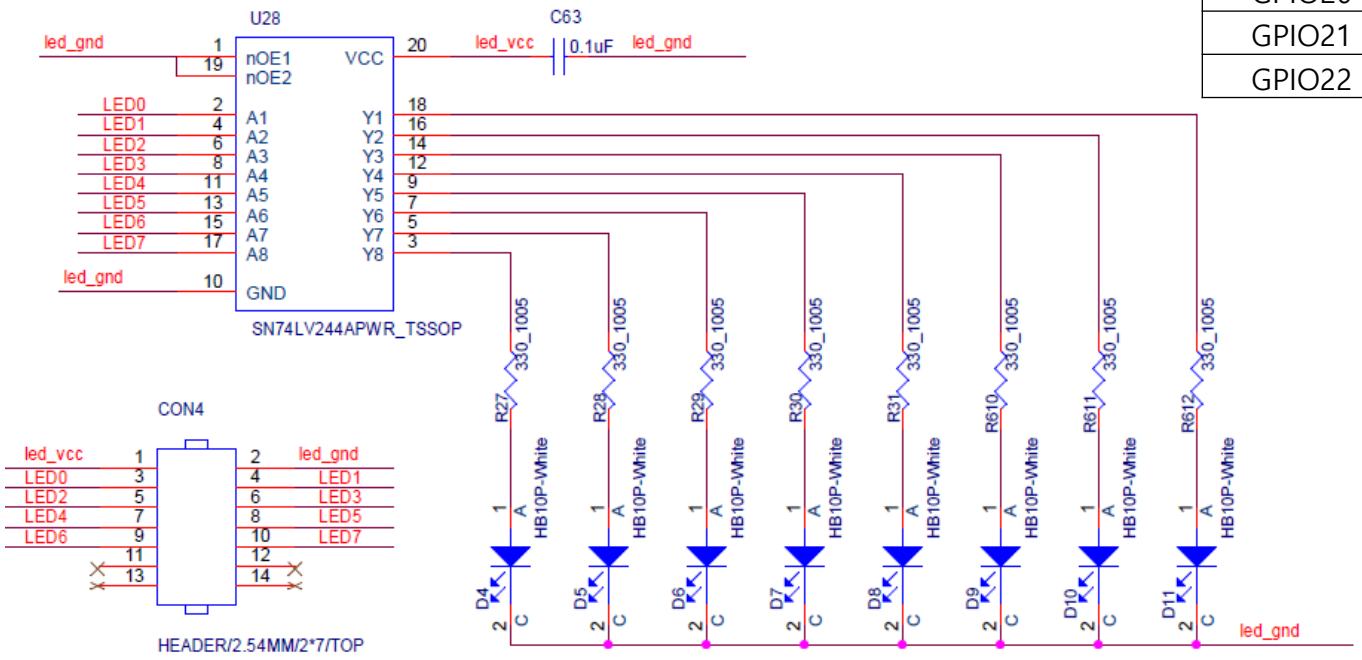
HUINS

LED 회로도(Ext. Module)

- Extension Module로 8개의 LED를 확장하여 사용할 수 있다.



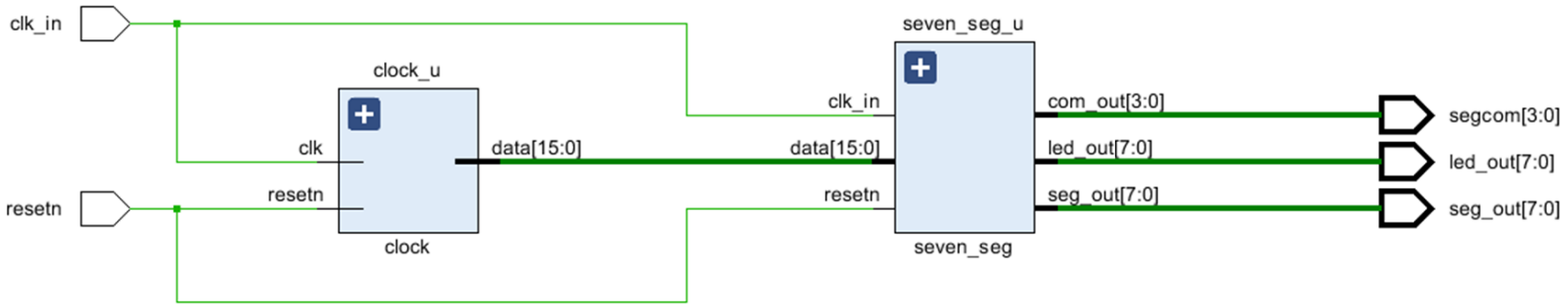
GPIO	Package Pin	External Module
GPIO2	K15	CLK
GPIO15	N13	LED
GPIO16	L13	LED
GPIO17	G11	LED
GPIO18	H11	LED
GPIO19	R12	LED
GPIO20	M14	LED
GPIO21	P15	LED
GPIO22	H13	LED



HUINS

LED와 7-Segment를 이용한 시계 설계하기

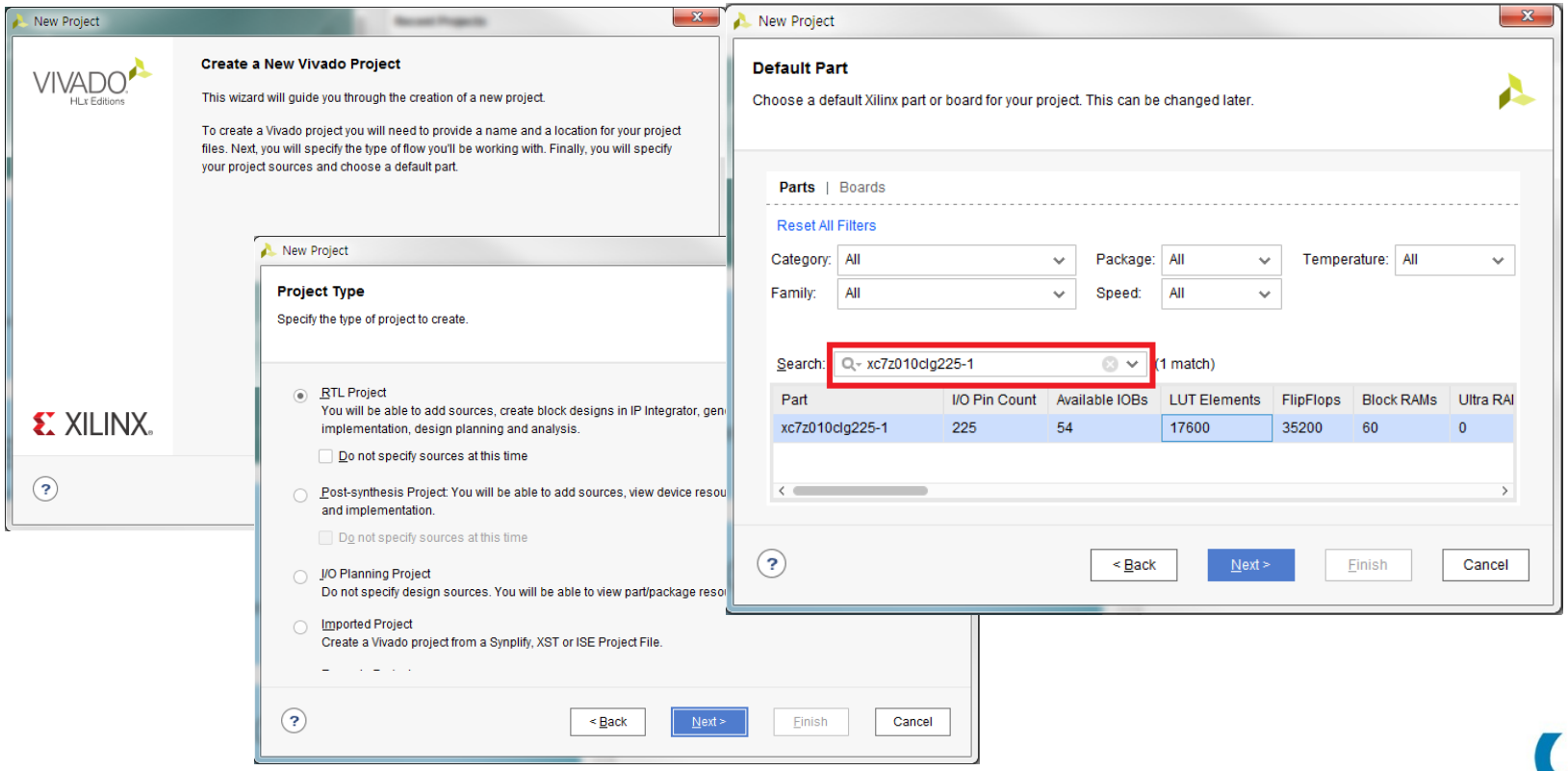
- 시계회로 블록 다이어그램
  - clock 블록의 기능
    - ✓ 25MHz 클록 입력을 1초 단위로 분주한다.
    - ✓ 분주된 1초 단위로 1초, 10초, 분, 10분의 데이터를 생성한다.
  - Seven_seg 블록의 기능
    - ✓ Clock 블록에서 만들어진 시간의 데이터를 7-Segment에 순차적으로 출력한다.



HUINS

LED와 7-Segment를 이용한 시계 설계하기

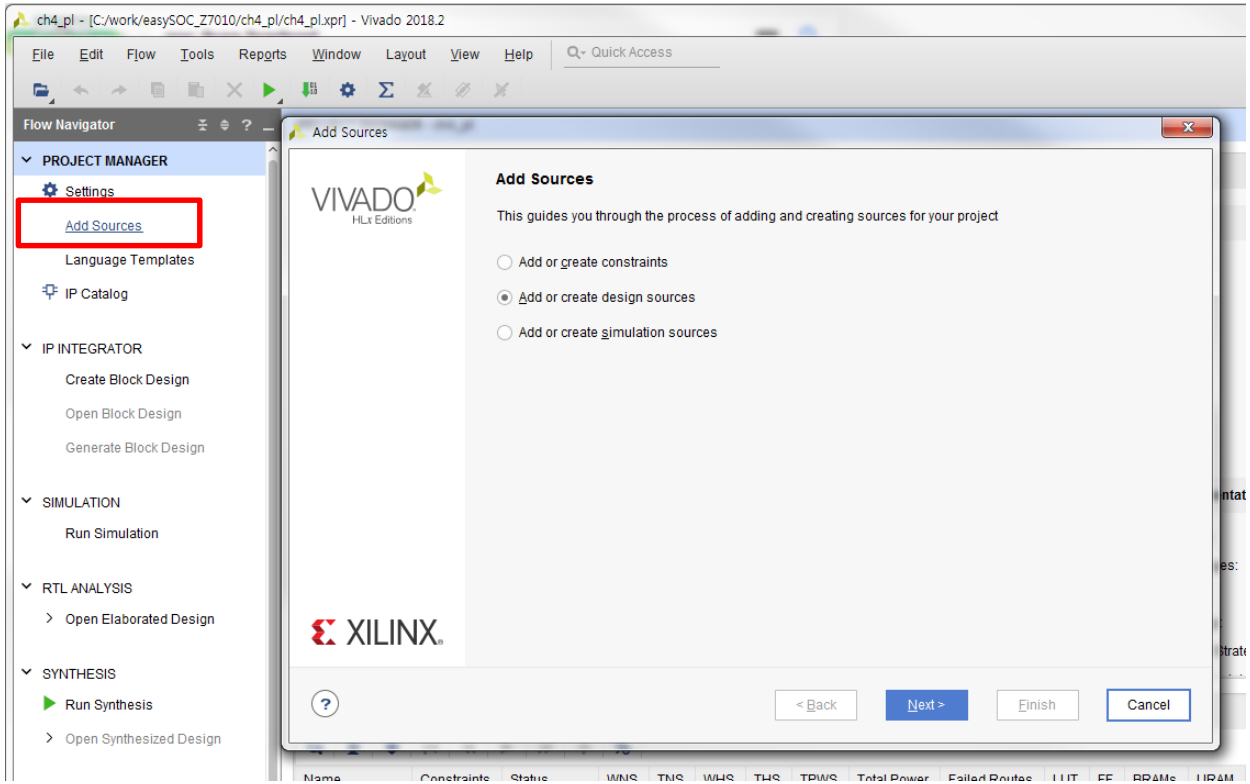
- 새 프로젝트를 생성한다.
- Project type은 RTL Project로, Xilinx part는 xc7z010clg225-1로 설정한다.



HUINS

LED와 7-Segment를 이용한 시계 설계하기

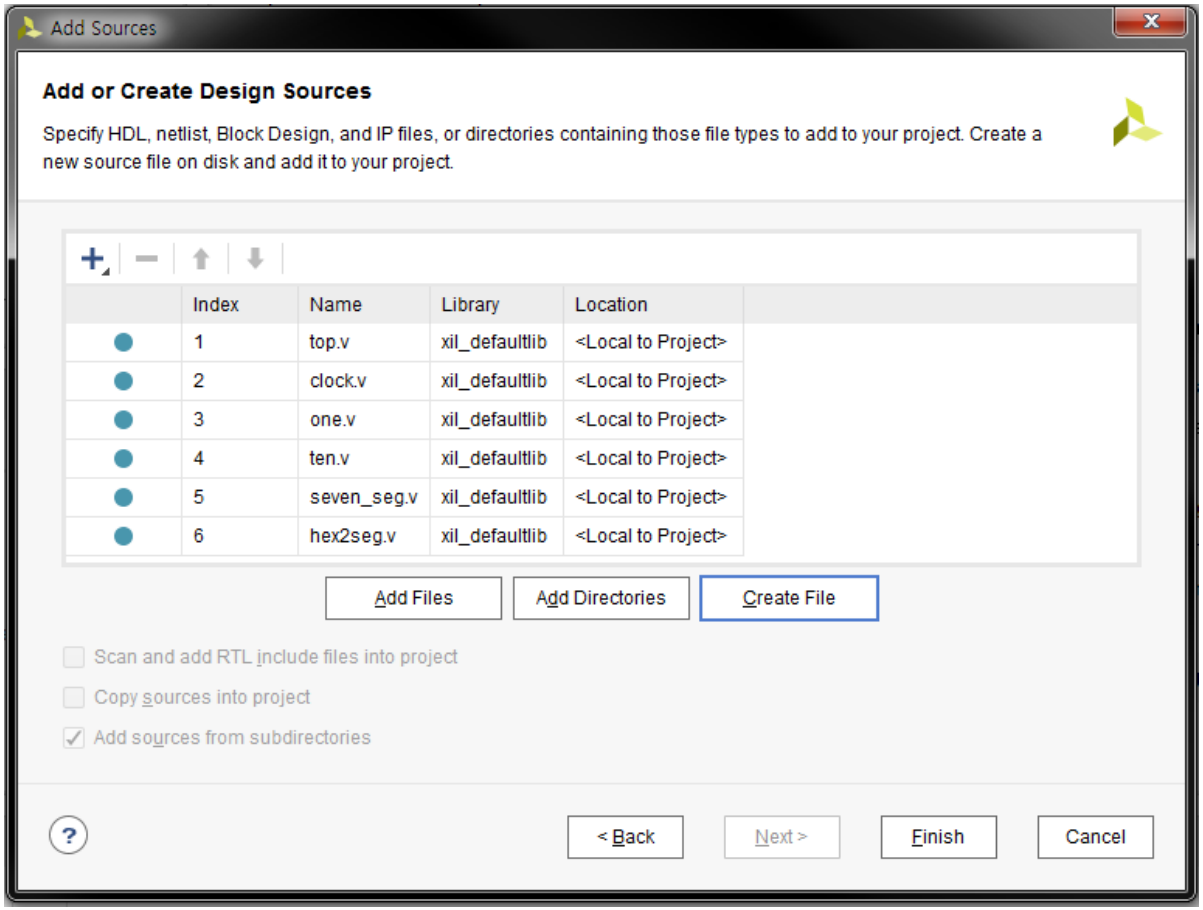
- 시계회로 설계를 위한 RTL 코드 파일 생성
  - File=> Add sources 또는 PROJECT MANAGER에서 Add Sources를 선택한다.
  - "Add sources" 대화상자에서 "Add or Create Design Sources"를 선택한다.



HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 시계회로 설계를 위한 RTL 코드 파일 생성
  - "Create File" 버튼을 눌러 다음과 같이 RTL 코드(Verilog) 파일을 생성한다.



HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - 소스코드는 총 6개의 파일로 구성되어 있으며 hierarchy는 아래와 같음.

HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - hex2seg.v 작성하기(7-Segment 구동부)

```
`timescale 1ns / 1ps
module hex2seg(
    input [3:0] hex,
    output reg [7:0] seg_data
);
always @ (*) begin
    case(hex)
        4'd0 : seg_data <= 8'b11111100 ;
        4'd1 : seg_data <= 8'b01100000 ;
        4'd2 : seg_data <= 8'b11011010 ;
        4'd3 : seg_data <= 8'b11110010 ;
        4'd4 : seg_data <= 8'b01100110 ;
        4'd5 : seg_data <= 8'b10110110 ;
        4'd6 : seg_data <= 8'b10111110 ;
        4'd7 : seg_data <= 8'b11100100 ;
        4'd8 : seg_data <= 8'b11111110 ;
        4'd9 : seg_data <= 8'b11110110 ;
        4'd10 : seg_data <= 8'b00000010 ;
        default : seg_data <= 8'b00111010;
    endcase
end
endmodule
```

HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - seven_seg.v 작성하기(7-Segment 구동부)

```
`timescale 1ns / 1ps
module seven_seg(
    input clk_in,
    input resetn,
    input [15:0] data,
    output reg [7:0] seg_out,
    output reg [3:0] com_out,
    output [7:0] led_out
);
reg [14:0] cnt;
reg [1:0] com_cnt;
wire [7:0] seg0;
wire [7:0] seg1;
wire [7:0] seg2;
wire [7:0] seg3;

hex2seg hex2seg_0(.hex(data[15:12]), .seg_data(seg0));
hex2seg hex2seg_1(.hex(data[11:8]), .seg_data(seg1));
hex2seg hex2seg_2(.hex(data[7:4]), .seg_data(seg2));
hex2seg hex2seg_3(.hex(data[3:0]), .seg_data(seg3));

assign led_out = {4'b0000,data[3:0]};
```

HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기

```
always @ (posedge clk_in, negedge resetn) begin
    if(!resetn) begin
        cnt <= 15'd0;
        com_cnt <= 2'd0;
    end
    else begin
        if(cnt < 16384)
            cnt <= cnt + 1;
        else begin
            cnt <= 0;
            com_cnt <= com_cnt + 1;
        end
    end
end
```

```
always @ (*) begin
    case(com_cnt)
        2'd0 : begin
            seg_out <= seg0;
            com_out <= ~4'b0001; end
        2'd1 : begin
            seg_out <= seg1;
            com_out <= ~4'b0010; end
        2'd2 : begin
            seg_out <= seg2;
            com_out <= ~4'b0100; end
        2'd3 : begin
            seg_out <= seg3;
            com_out <= ~4'b1000; end
        default : begin
            seg_out <= 8'b11111100;
            com_out <= ~4'b0000; end
    endcase
end

Endmodule
```

HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - clock.v 작성하기

```
`timescale 1ns / 1ps
module clock(
    input clk,
    input resetn,
    output [15:0] data
);
reg [32:0] counter;
reg sec_clk;

wire tensec_clk;
wire min_clk;
wire tenmin_clk;

always @ (posedge clk, negedge resetn) begin
    if(!resetn) begin
        counter <= 0;
        sec_clk <= 0;
    end
```

```
    else begin
        if (counter < 33'd12499999)
            counter <= counter + 33'd1;
        else begin
            counter <= 33'd0;
            sec_clk <= ~sec_clk;
        end
    end
end

one one_sec(.one_clk(sec_clk), .reset(resetn), .one(data[3:0]),
.ten_clk(tensec_clk));
ten ten_sec(.ten_clk(tensec_clk), .reset(resetn), .ten(data[7:4]),
.one_clk(min_clk));
one one_min(.one_clk(min_clk), .reset(resetn), .one(data[11:8]),
.ten_clk(tenmin_clk));
ten ten_min(.ten_clk(tenmin_clk), .reset(resetn),
.ten(data[15:12]));

endmodule
```

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - one.v 작성하기(0-9)

```
`timescale 1ns / 1ps
module one(
    input one_clk,
    input reset,
    output reg[3:0] one,
    output reg ten_clk
);

always @ (posedge one_clk, negedge reset) begin
    if(!reset) begin
        one <= 0;
        ten_clk <= 0;
    end
```

```
else begin
        if(one < 9) begin
            one <= one + 1;
            ten_clk <= 0;
        end
        else begin
            one <= 0;
            ten_clk <= 1;
        end
    end
end
endmodule
```

HUINS

LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - ten.v 작성하기(0-5)

```
`timescale 1ns / 1ps
module ten(
    input ten_clk,
    input reset,
    output reg[3:0] ten,
    output reg one_clk
);

always @ (posedge ten_clk, negedge reset) begin
    if(!reset) begin
        ten <= 0;
        one_clk <= 0;
    end
```

```
else begin
        if(ten < 5) begin
            ten <= ten + 1;
            one_clk <= 0;
        end
        else begin
            ten <= 0;
            one_clk <= 1;
        end
    end
end
endmodule
```

HUINS

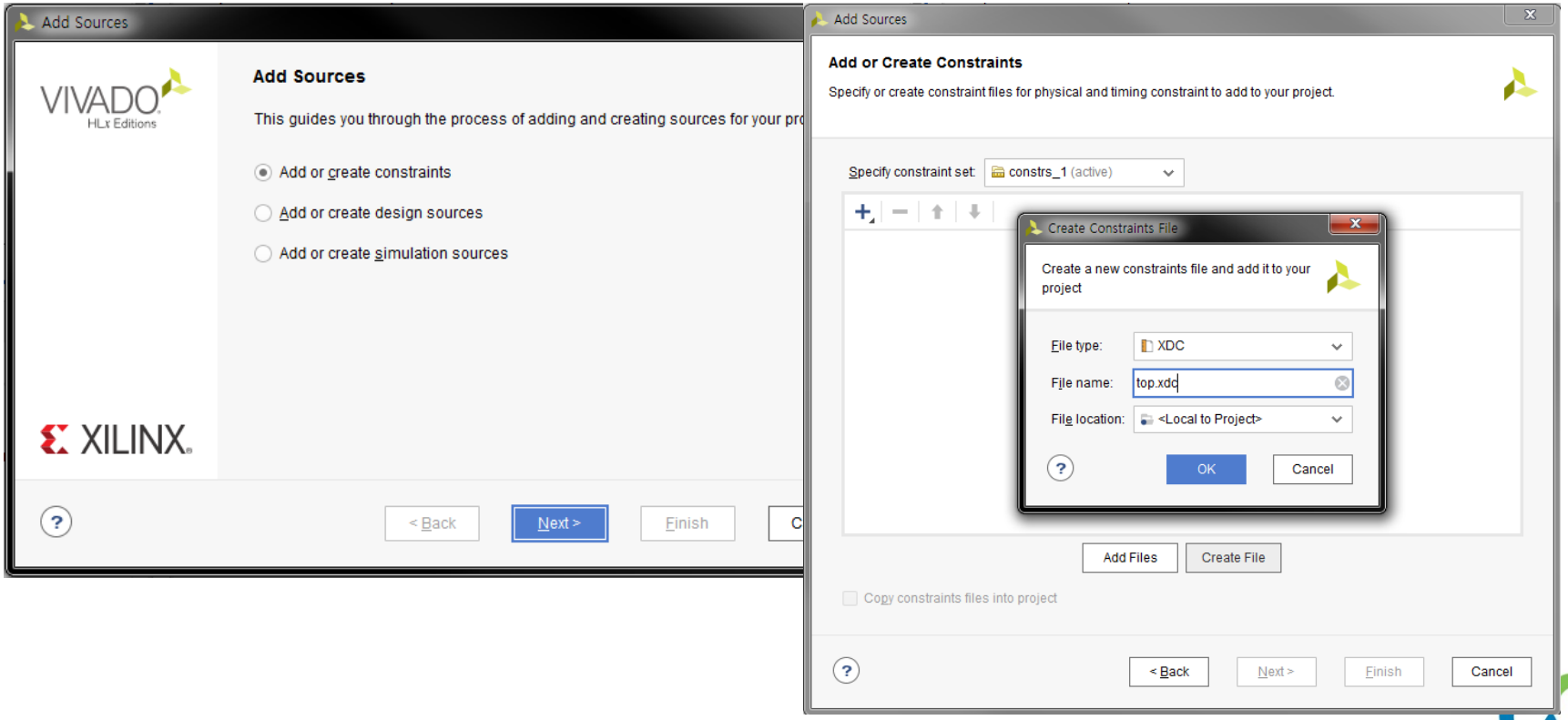
LED와 7-Segment를 이용한 시계 설계하기

- 소스코드 작성하기
  - top.v 작성하기

```
`timescale 1ns / 1ps
module top(
    input clk_in,
    input resetn,
    output [7:0] led_out,
    output [7:0] seg_out,
    output [3:0] segcom
);

wire [15:0] data;
clock clock_u(.clk(clk_in), .resetn(resetn), .data(data));
seven_seg seven_seg_u(.clk_in(clk_in), .resetn(resetn), .data(data), .seg_out(seg_out), .com_out(segcom), .led_out(led_out));

endmodule
```

LED와 7-Segment를 이용한 시계 설계하기

- 프로젝트 컴파일 하기
  - File => Add source(Alt + A) 선택 후 "Add or Create Constraints"를 선택해 새로운 Constraints 파일(top.xdc)을 생성한다.



HUINS

LED와 7-Segment를 이용한 시계 설계하기

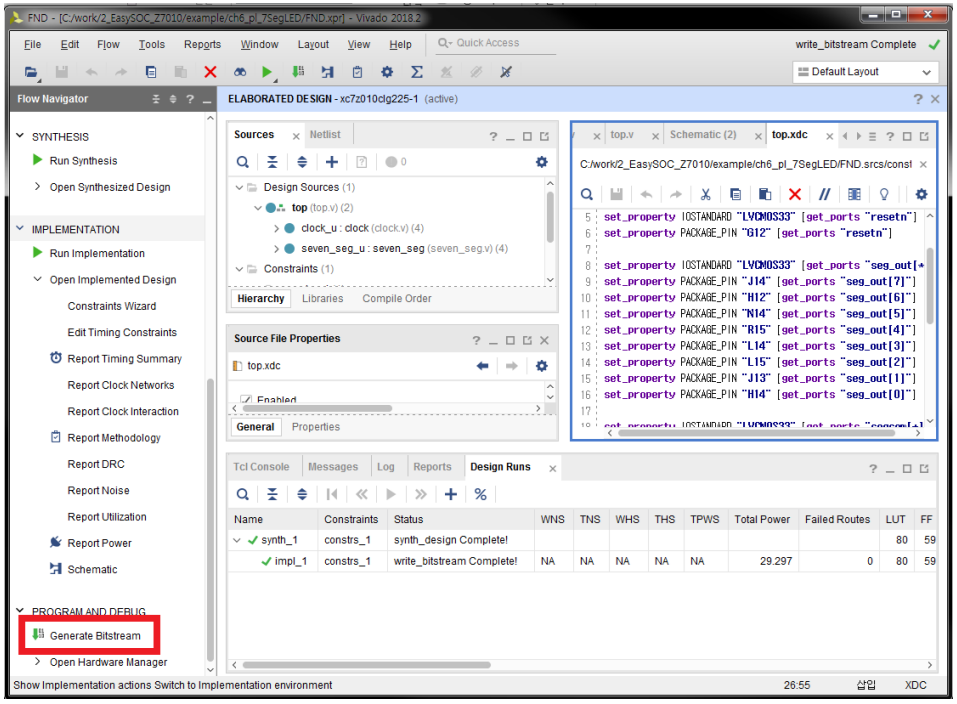
- 프로젝트 컴파일 하기
  - 생성된 constraints파일 top.xdc에 Zynq 핀 번호와 설계 파일의 I/O 이름을 맵핑 시키는 내용을 추가한다.

```
set_property IOSTANDARD "LVCMOS33" [get_ports "clk_in"]
set_property PACKAGE_PIN "K15" [get_ports "clk_in"]

set_property IOSTANDARD "LVCMOS33" [get_ports "resetn"]
set_property PACKAGE_PIN "G12" [get_ports "resetn"]

set_property IOSTANDARD "LVCMOS33" [get_ports "seg_out[*]"]
set_property PACKAGE_PIN "J14" [get_ports "seg_out[7]"]
set_property PACKAGE_PIN "H12" [get_ports "seg_out[6]"]
set_property PACKAGE_PIN "N14" [get_ports "seg_out[5]"]
set_property PACKAGE_PIN "R15" [get_ports "seg_out[4]"]
set_property PACKAGE_PIN "L14" [get_ports "seg_out[3]"]
set_property PACKAGE_PIN "L15" [get_ports "seg_out[2]"]
set_property PACKAGE_PIN "J13" [get_ports "seg_out[1]"]
set_property PACKAGE_PIN "H14" [get_ports "seg_out[0]"]
```

```
set_property IOSTANDARD "LVCMOS33" [get_ports "segcom[*]"]
set_property PACKAGE_PIN "M12" [get_ports "segcom[3]"]
set_property PACKAGE_PIN "R13" [get_ports "segcom[2]"]
set_property PACKAGE_PIN "M15" [get_ports "segcom[1]"]
set_property PACKAGE_PIN "J15" [get_ports "segcom[0]"]

set_property IOSTANDARD "LVCMOS33" [get_ports "led_out[*]"]
set_property PACKAGE_PIN "N13" [get_ports "led_out[7]"]
set_property PACKAGE_PIN "L13" [get_ports "led_out[6]"]
set_property PACKAGE_PIN "G11" [get_ports "led_out[5]"]
set_property PACKAGE_PIN "H11" [get_ports "led_out[4]"]
set_property PACKAGE_PIN "R12" [get_ports "led_out[3]"]
set_property PACKAGE_PIN "M14" [get_ports "led_out[2]"]
set_property PACKAGE_PIN "P15" [get_ports "led_out[1]"]
set_property PACKAGE_PIN "H13" [get_ports "led_out[0]"]
```

LED와 7-Segment를 이용한 시계 설계하기

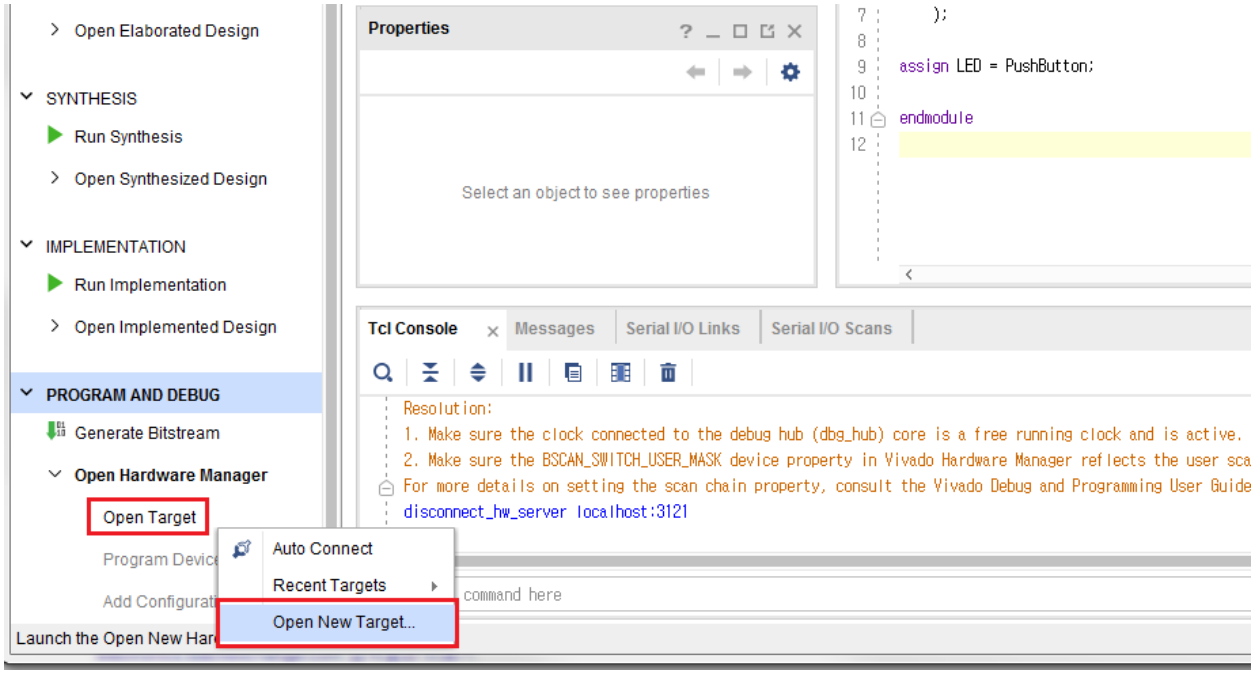
- 프로젝트 컴파일 하기
  - PL영역에 프로그램 할 Bitstream파일을 생성한다.
  - Generate Bitstream을 실행할 경우 순서대로 Synthesis, Implementation, Generate Bitstream 순으로 진행이 된다.



HUINS

Programmable Logic을 포함한 프로젝트 생성 및 동작시키기

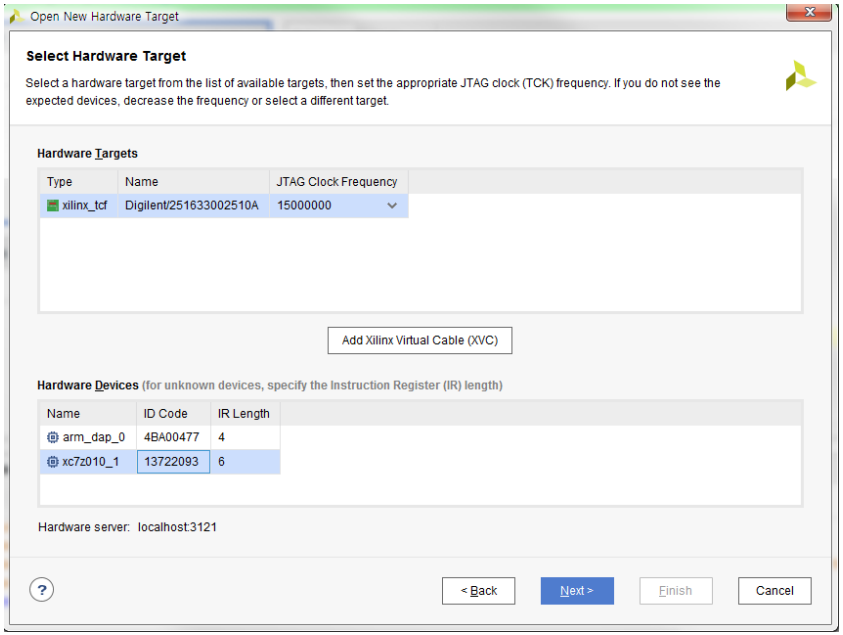
- Programmable Logic(PL)을 이용한 실습
  - Micro usb로 보드와 host 컴퓨터를 연결한다.
  - Vivado 툴에서 Open Hardware Manager => Open Target 항목을 클릭한 후 Open New Target을 선택한다.

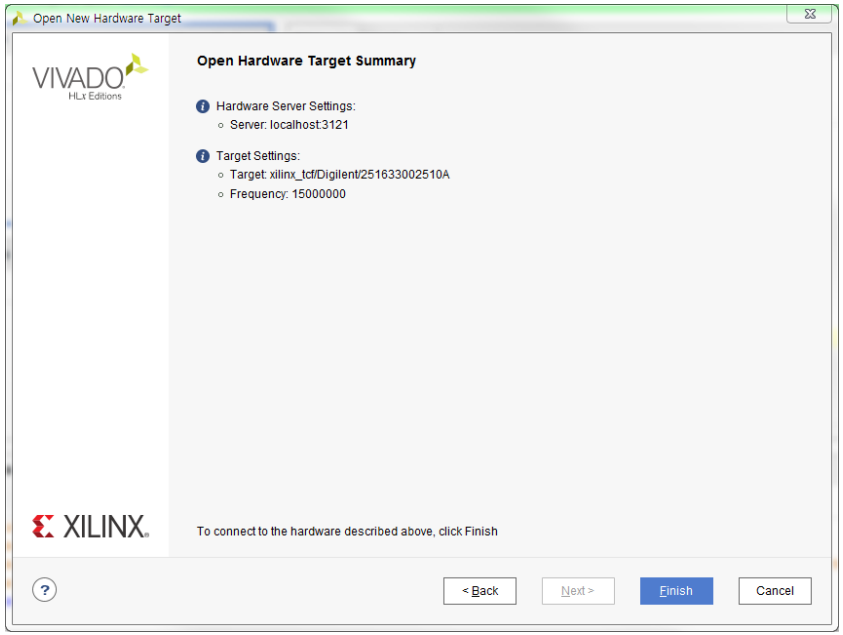


HUINS

Programmable Logic을 포함한 프로젝트 생성 및 동작시키기

- Programmable Logic(PL)을 이용한 실습
  - Hardware Devices에 XC7Z010_1을 선택하고 next 버튼을 클릭한다.
  - ZYNQ-7000 PL에 Bitstream 파일을 다운로드 하기 위한 설정 내용을 알려주는 대화상자에서 Finish버튼을 클릭해 연결을 완료한다.

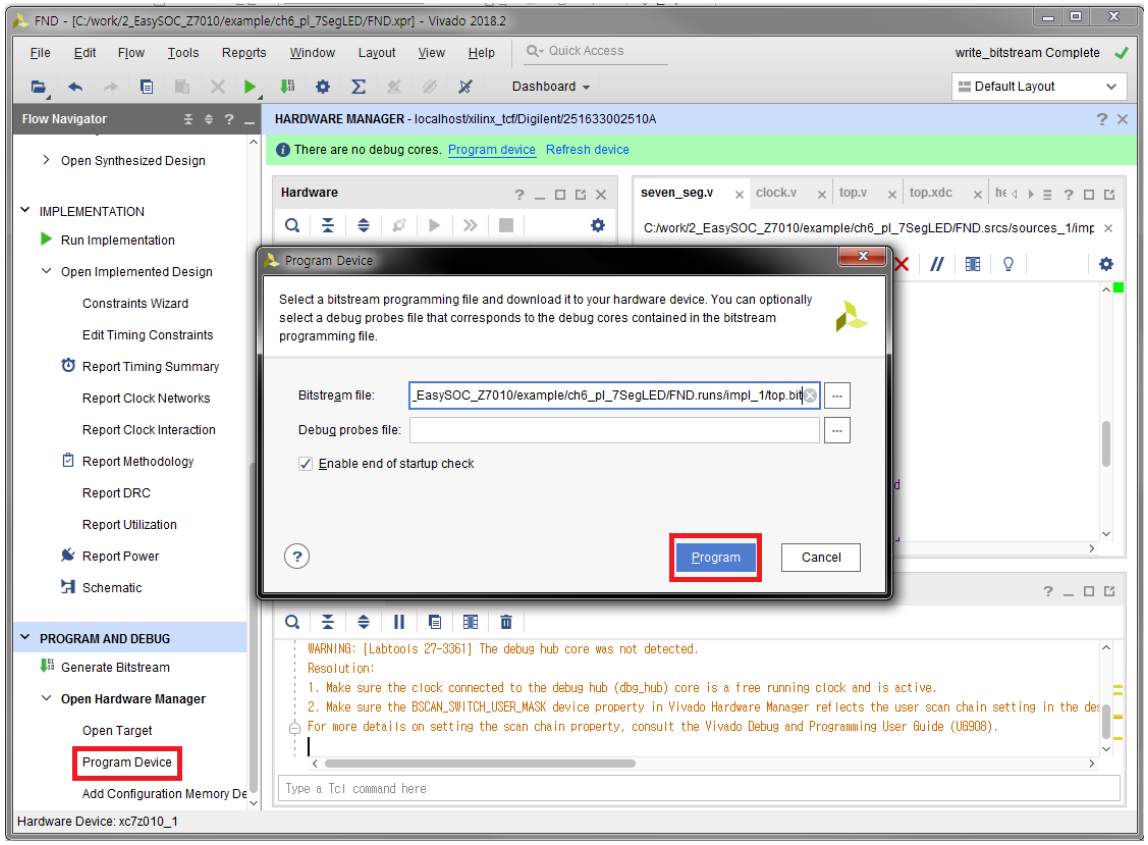




HUINS

Programmable Logic을 포함한 프로젝트 생성 및 동작시키기

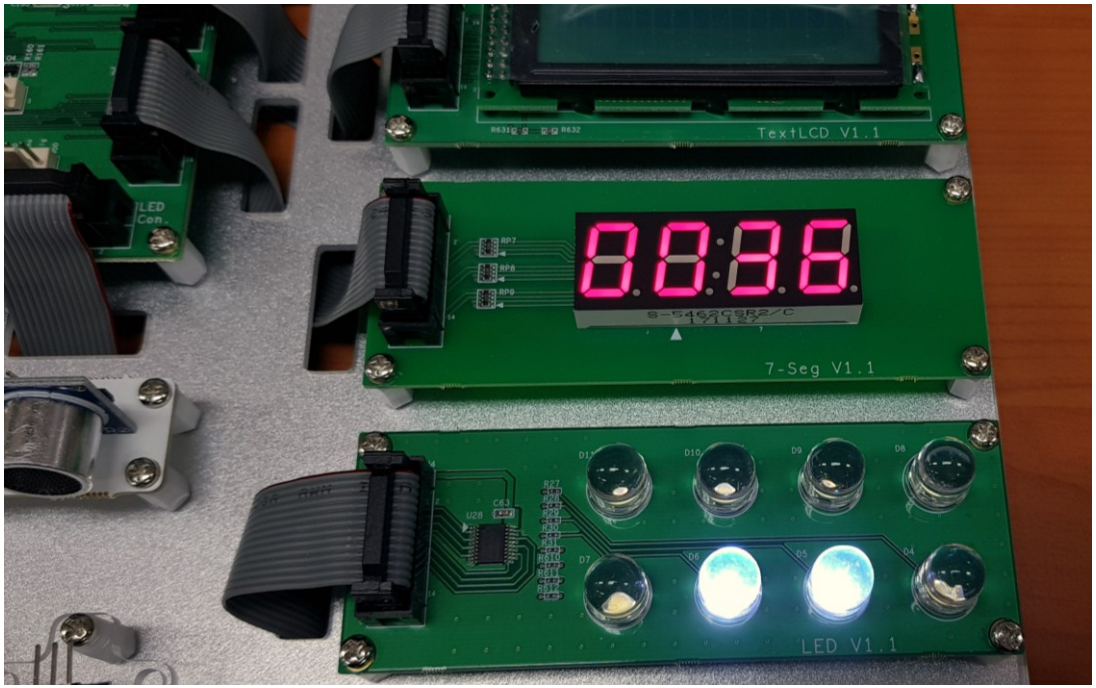
- Programmable Logic(PL)을 이용한 실습
  - Hardware Manager => Program Device 선택 후 xc7z10_1 항목을 선택한다.
  - 생성된 Bitstream 파일을 선택하여 다운로드 한다.



HUINS

Programmable Logic을 포함한 프로젝트 생성 및 동작시키기

- 7-Segment, LED 구동
  - 7-Segment는 매초 1씩 증가하며 59분 59초 이후 00분 00초로 다시 리셋된다.
  - 하단의 4개의 LED는 0-9초를 4bit로 표현한다.



HUINS

감사합니다.

HUINS