

# TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 – GJ (박현우)

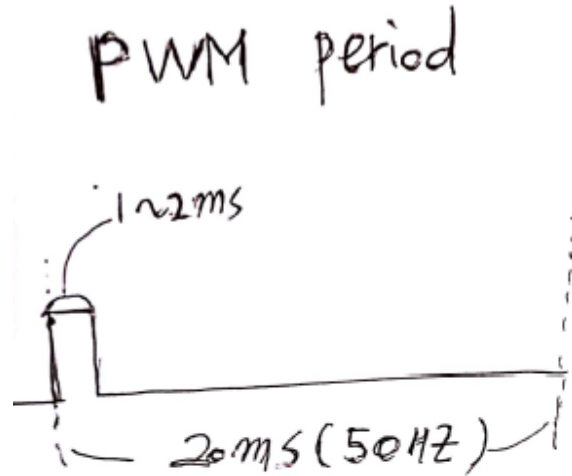
[uc820@naver.com](mailto:uc820@naver.com)

# 목차

## Zynq FPGA on Petalinux for PWM

- 1) pwm clock 분주 sequence
- 2) servo\_pwm with test\_bench & VHDL Code
- 3) pwm switch 만들기 & VHDL Code
- 4) vivado pwm ip 만들기 ( 카페 자료 참고 )
- 5) pwm ip로 내부 LED & 외부 LED 결과

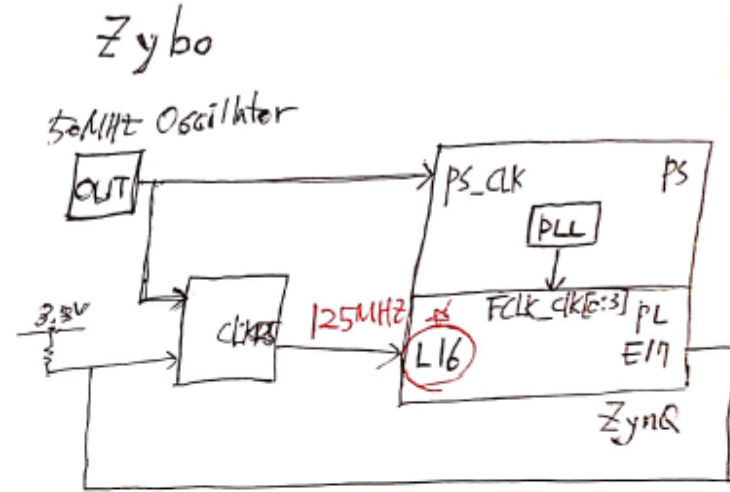
# 1) Pwm clock 분주 sequence



$$f = \frac{1}{T} \Rightarrow 1^\circ \text{단위제어} \Rightarrow f = \left(\frac{1\text{ms}}{180}\right)^{-1}$$

$$\therefore f = 180\text{kHz}$$

즉, 180kHz 신호를 만들어 주어야 함.



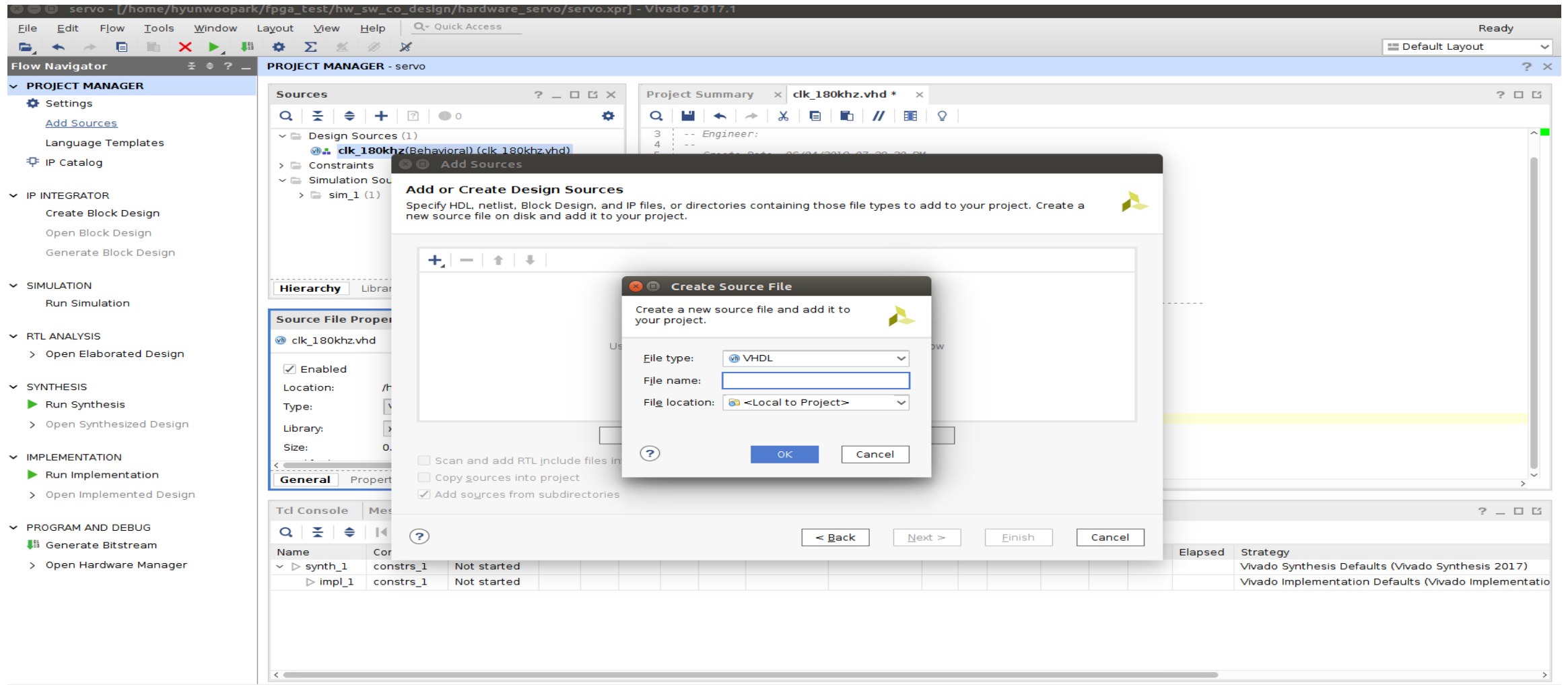
\* 125MHz를 180kHz 만들기

$$125 \times 10^6 = 180 \times 10^3 \times \text{scale}$$

$$\therefore \text{scale} = 694$$

즉, 125MHz를 694번 count하면  
180kHz가 만들어짐.

## 2) servo\_pwm with test\_bench & VHDL Code 1



## 2) servo\_pwm with test\_bench & VHDL Code 2

The screenshot displays the Vivado 2017.1 IDE interface for a project named 'servo'. The top menu bar includes File, Edit, Flow, Tools, Window, Layout, View, and Help. The top right corner shows 'Synthesis Complete' and 'Default Layout'.

The left sidebar contains the Flow Navigator with the following sections:

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design
- PROGRAM AND DEBUG
  - Generate Bitstream
  - Open Hardware Manager

The center area is divided into three main sections:

- Project Manager - servo**: Shows the Sources tree with Design Sources (1) containing 'clk\_180khz(Behavioral) (clk\_180khz.vhd)', Constraints, and Simulation Sources (1) containing 'sim\_1 (1)'.
- Source File Properties**: Shows properties for 'clk\_180khz.vhd'. It is Enabled, located at '/home/hyunwoopark/fpga\_test/hw\_sw\_co', Type is 'VHDL', Library is 'xil\_defaultlib', and Size is 1.2 KB.
- Project Summary**: Displays the VHDL code for 'clk\_180khz.vhd'.

The VHDL code in the Project Summary is as follows:

```
24
25 entity clk_180khz is
26     port(
27         clk_in : in std_logic;
28         reset : in std_logic;
29         clk_out : out std_logic
30     );
31 end clk_180khz;
32
33 architecture Behavioral of clk_180khz is
34
35     signal temp : std_logic;
36     signal counter : integer range 0 to 346 := 0;
37 begin
38     frequency_divider : process(reset, clk_in)
39     begin
40         if(reset = '1') then
41             temp <= '0';
42             counter <= 0;
43         elsif rising_edge(clk_in) then
44             if(counter = 346) then
45                 temp <= NOT(temp);
46                 counter <= 0;
47             else
48                 counter <= counter + 1;
49             end if;
50         end if;
51     end process;
52
53     clk_out <= temp;
54
55 end Behavioral;
56
```

The bottom section is the Design Runs table, which shows the status of the synthesis and implementation runs.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMS	URAM	DSP	Start	Elapsed	Strategy
✓ synth_1	constrs_1	synth_design Complete!								13	10	0.00	0	0	6/4/18 7:36 PM	00:00:20	Vivado Synthesis Defaults (Viva
▷ impl_1	constrs_1	Not started															Vivado Implementation Defaults

## 2) servo\_pwm with test\_bench & VHDL Code 3

The screenshot displays the Vivado 2017.1 IDE interface for a project named 'servo'. The left sidebar shows the 'Flow Navigator' with various project stages: PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. The 'PROJECT MANAGER' tab is active, showing a tree view of 'Design Sources' and 'Simulation Sources'. The 'Design Sources' tree lists 'clk\_180khz(Behavioral) (clk\_180khz.vhd)' and 'server\_pwm(Behavioral) (server\_pwm.vhd)'. The 'Simulation Sources' tree lists 'sim\_1 (2)'. The 'Source File Properties' panel for 'server\_pwm.vhd' is open, showing it is 'Enabled' and located at '/home/hyunwoopark/fpga\_test/hw\_sw\_co'. The 'VHDL' type is selected, and the library is 'xilinx\_defaultlib'. The 'General' tab is active, showing the file size as 0.9 KB. The 'Design Runs' table at the bottom shows the synthesis status.

**Project Manager - servo**

**Sources**

- Design Sources (2)
  - clk\_180khz(Behavioral) (clk\_180khz.vhd)
  - server\_pwm(Behavioral) (server\_pwm.vhd)
- Constraints
- Simulation Sources (2)
  - sim\_1 (2)

**Source File Properties**

server\_pwm.vhd

- ☒ Enabled
- Location: /home/hyunwoopark/fpga\_test/hw\_sw\_co
- Type: VHDL
- Library: xilinx\_defaultlib
- Size: 0.9 KB

**General** Properties

**Design Runs**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Strategy
synth_1	constrs_1	Synthesis Out-of-date								13	10	0.00	0	0	6/4/18 7:36 PM	00:00:20	Vivado Synthesis Defaults (Vivado)
impl_1	constrs_1	Not started															Vivado Implementation Defaults

## 2) servo\_pwm with test\_bench & VHDL Code 4

The screenshot displays the Xilinx Vivado IDE interface. The 'Project Manager' on the left shows the project 'servo' with sources 'clk\_180khz.vhd' and 'server\_pwm.vhd'. The 'Define Module' dialog box is open, prompting the user to define a module and specify I/O ports. The dialog includes fields for 'Entity name' (set to 'servo\_pwm\_180khz') and 'Architecture name' (set to 'Behavioral'). Below these, the 'I/O Port Definitions' table is shown, with one port defined as 'In' with MSB and LSB values of 0. The background shows the 'Tcl Console' at the bottom with a table of synthesis results.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Strategy
synth_1	constrs_1	Synthesis Out-of-date								13	10	0.00	0	0	6/4/18 7:36 PM	00:00:20	Vivado Synthesis Defaults (Vivado)
impl_1	constrs_1	Not started															Vivado Implementation Defaults



## 2) servo\_pwm with test\_bench & VHDL Code 5

The screenshot displays the Vivado IDE interface for a project named "servo". The left sidebar contains the Project Manager and Flow Navigator. The top toolbar shows various editing and simulation tools. The main window is split into three panes: Sources, Source File Properties, and a code editor.

**Sources Pane:** Shows the project hierarchy. Design Sources (3) include clk\_180khz(Behavioral) (clk\_180khz.vhd), server\_pwm(Behavioral) (server\_pwm.vhd), and servo\_pwm\_180khz(Behavioral) (servo\_pwm\_180khz.vhd). Simulation Sources (3) include sim\_1 (3), which contains servo\_pwm\_180khz(Behavioral) (servo\_pwm\_180khz.vhd), server\_pwm(Behavioral) (server\_pwm.vhd), and servo\_pwm\_clk180khz\_tb(Behavioral) (servo\_pwm\_clk180khz\_tb.vhd).

**Source File Properties Pane:** Shows properties for servo\_pwm\_clk180khz\_tb.vhd. It is enabled, located at /home/hyunwoopark/fpga\_test/hw\_sw\_co, type is VHDL, library is xil\_defaultlib, and size is 1.1 KB.

**Code Editor:** Displays the VHDL code for the test bench. The code is as follows:

```
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity servo_pwm_clk180khz_tb is
26 end servo_pwm_clk180khz_tb;
27
28
29 architecture Behavioral of servo_pwm_clk180khz_tb is
30
31     component servo_pwm_180khz
32     port(
33         clk : in std_logic;
34         reset : in std_logic;
35         pos : in std_logic_vector(9 downto 0);
36         servo : out std_logic
37     );
38     end component;
39
40     signal clk : std_logic := '0';
41     signal reset : std_logic := '0';
42     signal pos : std_logic_vector(9 downto 0) := (others => '0');
43
44     signal servo : std_logic;
45
46     constant clk_period : time := 8 ns; -- f = 125 Mhz => 8 ns;
47
48 begin
49     -- Unit under Test
50     uut: servo_pwm_180khz port map(
51         clk => clk,
52         reset => reset,
```

**Design Runs Table:**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMS	URAM	DSP	Start	Elapsed	Strategy
synth_1	constrs_1	Synthesis Out-of-date								13	10	0.00	0	0	6/4/18 7:36 PM	00:00:20	Vivado Synthesis Defaults (Vivado)
impl_1	constrs_1	Not started															Vivado Implementation Default



## 2) servo\_pwm with test\_bench & VHDL Code 6

The screenshot shows the Vivado 2017.1 IDE interface. The 'Settings' dialog is open, displaying the 'Simulation' tab. The 'Simulation' section is highlighted in the left sidebar. The 'Simulation' tab in the dialog shows the following settings:

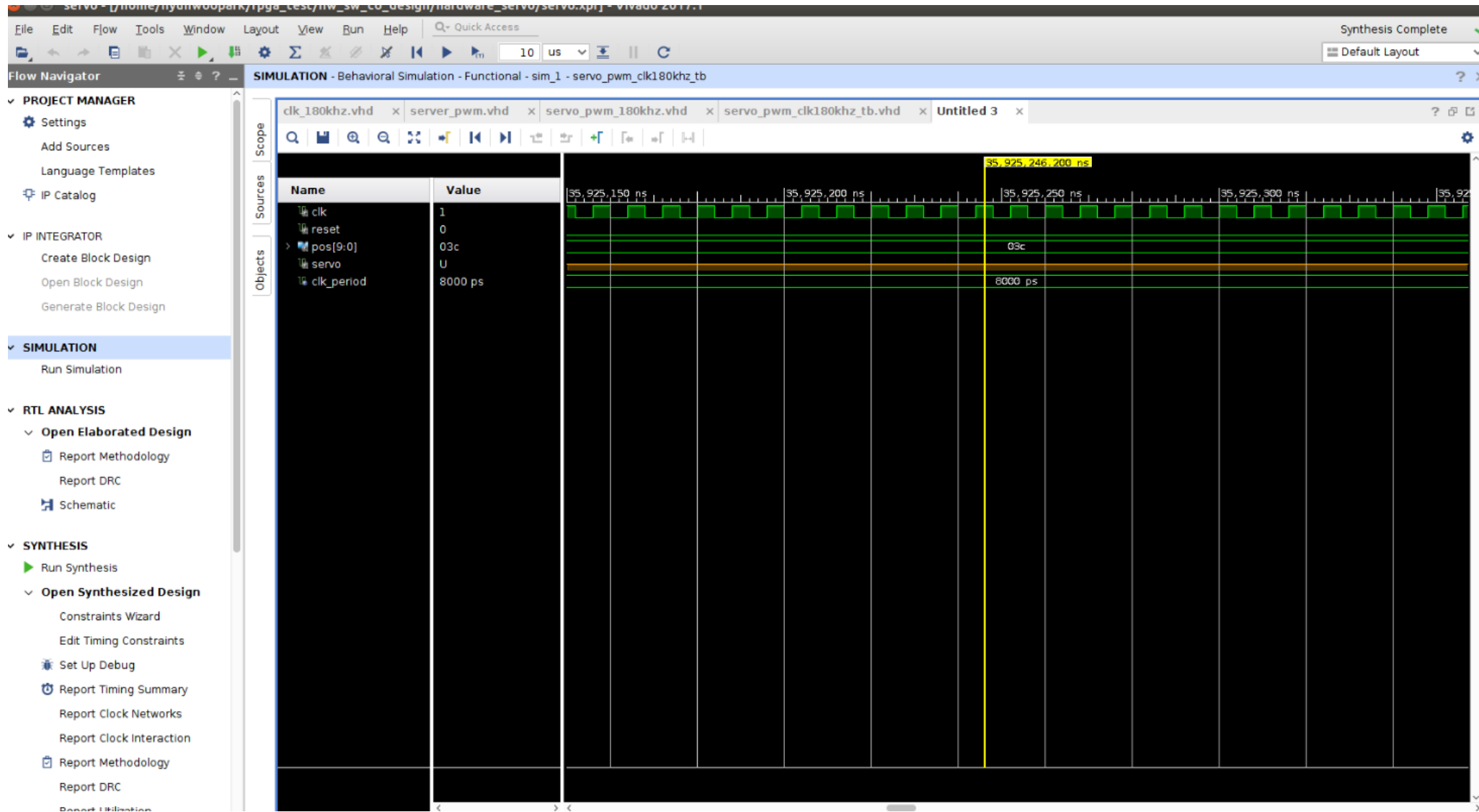
- Target simulator: Vivado Simulator
- Simulator language: Mixed
- Simulation set: sim\_1
- Simulation top module name: servo\_pwm\_clk180khz\_tb
- ☒ Clean up simulation files

The 'Simulation' tab also contains a table of simulation options:

Option	Value
xsim.simulate.tcl.post	
xsim.simulate.runtime*	100ms
xsim.simulate.log_all_signals	<input type="checkbox"/>
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	

The '100ms' value in the 'xsim.simulate.runtime\*' row is highlighted with a red box. The 'General' tab in the left sidebar is also highlighted with a red box.

## 2) servo\_pwm with test\_bench & VHDL Code 7



## 2) servo\_pwm with test\_bench & VHDL Code 8

### Clk\_180kHz.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clk_180kHz is
port(
clk : in std_logic;
reset : in std_logic;
clk_out : out std_logic
);
end clk_180kHz;

architecture Behavioral of clk_180kHz is

signal temp : std_logic;
signal counter : integer range 0 to 346 := 0;

begin
frequency_divider : process(reset, clk)

begin
if(reset = '1') then
temp <= '0';
counter <= 0;
elsif rising_edge(clk) then
if(counter = 346) then
temp <= NOT(temp);
counter <= 0;
else
counter <= counter + 1;
end if;
end if;
end process;

clk_out <= temp;

end Behavioral;
```

### Pwm\_servo.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity servo_pwm is
port(
clk : in std_logic;
reset : in std_logic;
pos : in std_logic_vector(7 downto 0);
servo : out std_logic
);
end servo_pwm;

architecture Behavioral of servo_pwm is
signal cnt : unsigned(11 downto 0);
signal pwmi : unsigned(9 downto 0);
begin
pwmi <= "00" & unsigned(pos) + 180;
counter : process(reset, clk) begin
if(reset = '1') then
cnt <= (others => '0');
elsif rising_edge(clk) then
if(cnt = 3599) then
cnt <= (others => '0');
else
cnt <= cnt + 1;
end if;
end if;
end process;

servo <= '1' when (cnt < pwmi) else '0';

end Behavioral;
```

### Servo\_pwm\_clk\_180kHz.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity servo_pwm_clk_180kHz is
port(
clk : in std_logic;
reset : in std_logic;
sw : in std_logic_vector(3 downto 0);

servo : out std_logic
);
end servo_pwm_clk_180kHz;

architecture Behavioral of servo_pwm_clk_180kHz is
component switch
port(
sw : in std_logic_vector(3 downto 0);
pos : out std_logic_vector(7 downto 0)
);
end component;

component clk_180kHz
port(
clk : in std_logic;
reset : in std_logic;
clk_out : out std_logic
);
end component;

component servo_pwm
port(
clk : in std_logic;
reset : in std_logic;
pos : in std_logic_vector(7 downto 0);
servo : out std_logic
);
end component;
```

```
signal clk_out : std_logic := '0';
signal pos : std_logic_vector(7 downto 0);
begin
switch_in : switch port map(
sw => sw, pos => pos
);

clk_180kHz_map : clk_180kHz port map(
clk => clk, reset => reset, clk_out => clk_out
);

servo_pwm_map : servo_pwm port map(
clk => clk_out, reset => reset,
pos => pos, servo => servo
);

end Behavioral;
```

## 2) servo\_pwm with test\_bench & VHDL Code 9

### Servo\_pwm\_clk\_180kHz\_tb.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity servo_pwm_clk180kHz_tb is
end servo_pwm_clk180kHz_tb;

architecture Test of servo_pwm_clk180kHz_tb is
    component servo_pwm_clk180kHz
    port(
        clk : in std_logic;
        reset : in std_logic;
        pos : in std_logic_vector(9 downto 0);
        servo : out std_logic
    );
    end component;

    signal clk : std_logic := '0';
    signal reset : std_logic := '0';
    signal pos : std_logic_vector(9 downto 0) := (others => '0');

    signal servo : std_logic;

    constant clk_period : time := 8 ns;
```

```
begin
    -- Unit Under Test
    uut: servo_pwm_clk180kHz port map(

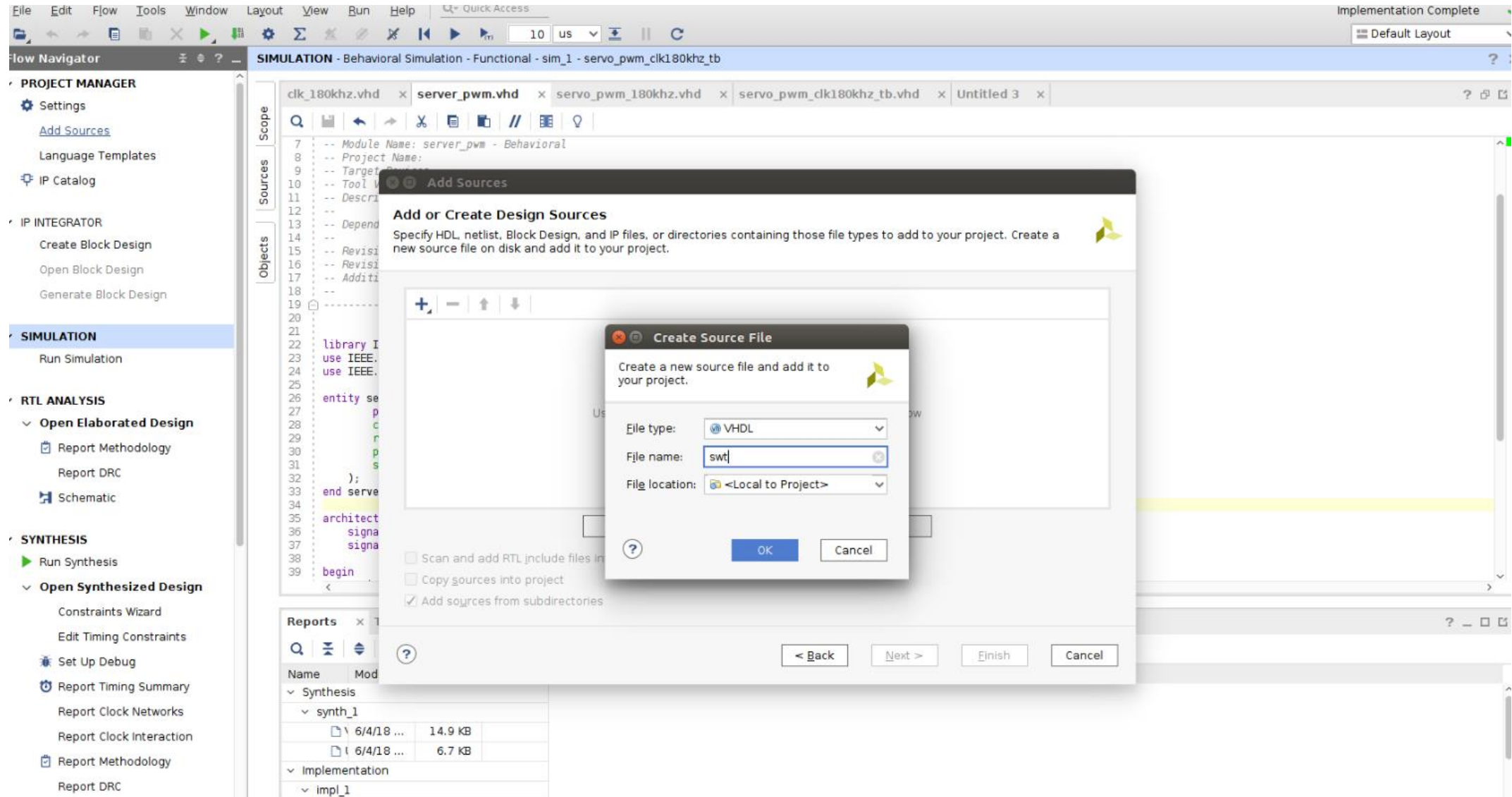
        clk => clk,
        reset => reset,
        pos => pos,
        servo => servo
    );

    clk_process : process begin
        clk <= '0';
        wait for clk_period / 2;
        clk <= '1';
        wait for clk_period / 2;
    end process;

    stimuli : process begin
        reset <= '1';
        wait for 50 ns;
        reset <= '0';
        wait for 50 ns;
        pos <= b"00_0000_0000";
        wait for 20 ms;
        pos <= b"00_0011_1100";
        wait for 20 ms;
        pos <= b"00_0111_1000";
        wait for 20 ms;
        pos <= b"00_1011_0100";
        wait;
    end process;

end Test;
```

### 3) pwm switch 만들기 & VHDL Code 1





### 3) pwm switch 만들기 & VHDL Code 2

The screenshot displays the Xilinx IDE interface with the following components:

- Sources Panel:** Shows a project hierarchy with files like `swtich.vhd`, `servo_pwm_180khz.vhd`, `clk_180khz_map.vhd`, `servo_pwm_map.vhd`, `server_pwm.vhd`, `zybo.xdc`, and `Simulation Sources`.
- Source File Properties Panel:** Shows details for `zybo.xdc`, including its location, type (XDC), size (11.4 KB), and modification date.
- Project Summary Panel:** Displays the VHDL code for `servo_pwm_180khz.vhd`, `swtich.vhd`, `server_pwm.vhd`, and `zybo.xdc`. The code is organized into sections for different pins (JB, JC, JD) and includes properties for package pins and I/O standards.

The VHDL code in the Project Summary panel is as follows:

```
80 ##Pmod Header JB
81 #set_property -dict { PACKAGE_PIN T20 IOSTANDARD LVCMOS33 } [get_ports { jb_p[0] }]; #IO_L15P_T2_DQS_34 Sch=JB1_p
82 #set_property -dict { PACKAGE_PIN U20 IOSTANDARD LVCMOS33 } [get_ports { jb_n[0] }]; #IO_L15N_T2_DQS_34 Sch=JB1_N
83 #set_property -dict { PACKAGE_PIN V20 IOSTANDARD LVCMOS33 } [get_ports { jb_p[1] }]; #IO_L16P_T2_34 Sch=JB2_P
84 #set_property -dict { PACKAGE_PIN W20 IOSTANDARD LVCMOS33 } [get_ports { jb_n[1] }]; #IO_L16N_T2_34 Sch=JB2_N
85 #set_property -dict { PACKAGE_PIN Y18 IOSTANDARD LVCMOS33 } [get_ports { jb_p[2] }]; #IO_L17P_T2_34 Sch=JB3_P
86 #set_property -dict { PACKAGE_PIN Y19 IOSTANDARD LVCMOS33 } [get_ports { jb_n[2] }]; #IO_L17N_T2_34 Sch=JB3_N
87 #set_property -dict { PACKAGE_PIN W18 IOSTANDARD LVCMOS33 } [get_ports { jb_p[3] }]; #IO_L22P_T3_34 Sch=JB4_P
88 #set_property -dict { PACKAGE_PIN W19 IOSTANDARD LVCMOS33 } [get_ports { jb_n[3] }]; #IO_L22N_T3_34 Sch=JB4_N
89
90
91 ##Pmod Header JC
92 set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { jc_p[0] }]; #IO_L10P_T1_34 Sch=JC1_P
93 #set_property -dict { PACKAGE_PIN W15 IOSTANDARD LVCMOS33 } [get_ports { jc_n[0] }]; #IO_L10N_T1_34 Sch=JC1_N
94 #set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { jc_p[1] }]; #IO_L1P_T0_34 Sch=JC2_P
95 #set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { jc_n[1] }]; #IO_L1N_T0_34 Sch=JC2_N
96 #set_property -dict { PACKAGE_PIN W14 IOSTANDARD LVCMOS33 } [get_ports { jc_p[2] }]; #IO_L8P_T1_34 Sch=JC3_P
97 #set_property -dict { PACKAGE_PIN Y14 IOSTANDARD LVCMOS33 } [get_ports { jc_n[2] }]; #IO_L8N_T1_34 Sch=JC3_N
98 #set_property -dict { PACKAGE_PIN T12 IOSTANDARD LVCMOS33 } [get_ports { jc_p[3] }]; #IO_L2P_T0_34 Sch=JC4_P
99 #set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { jc_n[3] }]; #IO_L2N_T0_34 Sch=JC4_N
100
101
102 ##Pmod Header JD
103 #set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { jd_p[0] }]; #IO_L5P_T0_34 Sch=JD1_P
104 #set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { jd_n[0] }]; #IO_L5N_T0_34 Sch=JD1_N
105 #set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { jd_p[1] }]; #IO_L6P_T0_34 Sch=JD2_P
106 #set_property -dict { PACKAGE_PIN R14 IOSTANDARD LVCMOS33 } [get_ports { jd_n[1] }]; #IO_L6N_T0_VREF_34 Sch=JD2_N
107 #set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { jd_p[2] }]; #IO_L11P_T1_SRCC_34 Sch=JD3_P
108 #set_property -dict { PACKAGE_PIN U15 IOSTANDARD LVCMOS33 } [get_ports { jd_n[2] }]; #IO_L11N_T1_SRCC_34 Sch=JD3_N
109 #set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { jd_p[3] }]; #IO_L21P_T3_DQS_34 Sch=JD4_P
110 #set_property -dict { PACKAGE_PIN V18 IOSTANDARD LVCMOS33 } [get_ports { jd_n[3] }]; #IO_L21N_T3_DQS_34 Sch=JD4_N
111
```



### 3) pwm switch 만들기 & VHDL Code 3

The screenshot displays the Xilinx Vivado IDE interface for a project named 'servo'. The top window shows the 'Project Manager' with a list of sources including 'servo\_pwm\_180khz', 'clk\_180khz\_map', 'servo\_pwm\_map', 'server\_pwm', 'constrs\_1', and 'zybo.xdc'. The 'zybo.xdc' file is selected, and its properties are shown in the 'Source File Properties' window. The 'zybo.xdc' file is located at '/home/hyunwoopark/fpga\_test/hw\_sw\_co', is 11.4 KB, and was modified today at 13:49:24 PM. The 'General' tab is active, showing the file is enabled and located in the project directory.

The 'Design Runs' window at the bottom shows the synthesis and implementation status. The 'synth\_1' run is 'Out-of-date', and the 'impl\_1' run has not started.

The 'zybo.xdc' file content is shown in the 'zybo.xdc \*' window. It is a general .xdc file for the ZYBO Rev B board, containing constraints for clock, switches, buttons, and LEDs.

```
1  ## This file is a general .xdc for the ZYBO Rev B board
2  ## To use it in a project:
3  ## - uncomment the lines corresponding to used pins
4  ## - rename the used signals according to the project
5
6
7  ##Clock signal
8  set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L11P_T1_SRCC_35 Sch=sysclk
9  create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];
10
11
12  ##Switches
13  set_property -dict { PACKAGE_PIN G15 IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L19N_T3_VREF_35 Sch=SW0
14  set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L24P_T3_34 Sch=SW1
15  set_property -dict { PACKAGE_PIN W13 IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=SW2
16  set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3
17
18
19  ##Buttons
20  set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L20N_T3_34 Sch=BTN0
21  set_property -dict { PACKAGE_PIN P16 IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L24N_T3_34 Sch=BTN1
22  set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L18P_T2_34 Sch=BTN2
23  set_property -dict { PACKAGE_PIN Y16 IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L7P_T1_34 Sch=BTN3
24
25
26  ##LEDs
27  set_property -dict { PACKAGE_PIN M14 IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L23P_T3_35 Sch=LED0
28  set_property -dict { PACKAGE_PIN M15 IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L23N_T3_35 Sch=LED1
29  set_property -dict { PACKAGE_PIN G14 IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_0_35=Sch=LED2
30  set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L3N_T0_DQS_AD1N_35 Sch=LED3
31
```

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Strategy
synth_1	constrs_1	Synthesis Out-of-date								13	10	0.00	0	0	6/4/18 1:30 PM	00:00:17	Vivado Synthesis Defaults (Vivado)
impl_1	constrs_1	Not started															Vivado Implementation Defaults


### 3) pwm switch 만들기 & VHDL Code 4


```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity clk_180kHz is
5     port(
6         clk : in std_logic;
7         reset : in std_logic;
8         clk_out : out std_logic
9     );
10 end clk_180kHz;
11
12 architecture Behavioral of clk_180kHz is
13     signal temp : std_logic;
14     signal counter : integer range 0 to 346 := 0;
15 begin
16     frequency_divider : process(reset, clk)
17     begin
18         if(reset = '1') then
19             temp <= '0';
20             counter <= 0;
21         elsif rising_edge(clk) then
22             if(counter = 346) then
23                 temp <= NOT(temp);
24                 counter <= 0;
25             else
26                 counter <= counter + 1;
27             end if;
28         end if;
29     end process;
30
31     clk_out <= temp;
32
33 end Behavioral;
```

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.NUMERIC_STD.ALL;
4
5 entity servo_pwm is
6     port(
7         clk : in std_logic;
8         reset : in std_logic;
9         pos : in std_logic_vector(7 downto 0);
10        servo : out std_logic
11    );
12 end servo_pwm;
13
14 architecture Behavioral of servo_pwm is
15     signal cnt : unsigned(11 downto 0);
16     signal pwmi : unsigned(9 downto 0);
17 begin
18     pwmi <= "00" & unsigned(pos) + 180;
19     counter : process(reset, clk) begin
20         if(reset = '1') then
21             cnt <= (others => '0');
22         elsif rising_edge(clk) then
23             if(cnt = 3599) then
24                 cnt <= (others => '0');
25             else
26                 cnt <= cnt + 1;
27             end if;
28         end if;
29     end process;
30
31     servo <= '1' when (cnt < pwmi) else '0';
32
33 end Behavioral;
```


```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity servo_pwm_clk_180kHz is
5     port(
6         clk : in std_logic;
7         reset : in std_logic;
8         sw : in std_logic_vector(3 downto 0);
9         servo : out std_logic
10    );
11 end servo_pwm_clk_180kHz;
12
13 architecture Behavioral of servo_pwm_clk_180kHz is
14     component switch
15     port(
16         sw : in std_logic_vector(3 downto 0);
17         pos : out std_logic_vector(7 downto 0)
18     );
19 end component;
20
21 component clk_180kHz
22     port(
23         clk : in std_logic;
24         reset : in std_logic;
25         clk_out : out std_logic
26     );
27 end component;
28
29 component servo_pwm
30     port(
31         clk : in std_logic;
32         reset : in std_logic;
33         pos : in std_logic_vector(7 downto 0);
34         servo : out std_logic
35     );
36 end component;
37
38 signal clk_out : std_logic := '0';
39 signal pos : std_logic_vector(7 downto 0);
```

 clk\_180kHz.vhd

 pwm\_servo.vhd

 servo\_pwm\_clk\_180kHz.vhd

 switch.vhd

 zybo.xdc

### 3) pwm switch 만들기 & VHDL Code 5

```
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;

34  entity switch is
35      port(
36          sw : in std_logic_vector(3 downto 0);
37          pos : out std_logic_vector(7 downto 0)
38      );
39  end switch;
40
41  architecture Behavioral of switch is
42
43  begin
44      P1 : process begin
45          switch : case sw is
46              when "0000" => pos <= X"00";
47              when "0001" => pos <= X"10";
48              when "0010" => pos <= X"20";
49              when "0011" => pos <= X"30";
50              when "0100" => pos <= X"40";
51              when "0101" => pos <= X"50";
52              when "0110" => pos <= X"60";
53              when "0111" => pos <= X"70";
54              when "1000" => pos <= X"80";
55
56              when "1001" => pos <= X"90";
57              when "1010" => pos <= X"A0";
58              when "1011" => pos <= X"B0";
59              when "1100" => pos <= X"C0";
60              when "1101" => pos <= X"D0";
61              when "1110" => pos <= X"E0";
62              when others => pos <= X"00";
63          end case switch;
64      end process;
65
66  end Behavioral;
```

```
~
7  ##Clock signal
8  set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports clk]
9  create_clock -period 8.000 -name sys_clk_pin -waveform {0.000 4.000} -add [get_ports clk]
10
11
12  ##Switches
13  set_property -dict {PACKAGE_PIN G15 IOSTANDARD LVCMOS33} [get_ports {sw[0]}]
14  set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports {sw[1]}]
15  set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {sw[2]}]
16  set_property -dict {PACKAGE_PIN T16 IOSTANDARD LVCMOS33} [get_ports {sw[3]}]
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148  set_property IOSTANDARD LVCMOS33 [get_ports reset]
149
150
151  set_property PACKAGE_PIN T19 [get_ports reset]
```



5) PWM ip로 내부 LED & 외부 LED 결과

