

How to control RC servo motor with PWM

이대로

skseofhek@daum.net

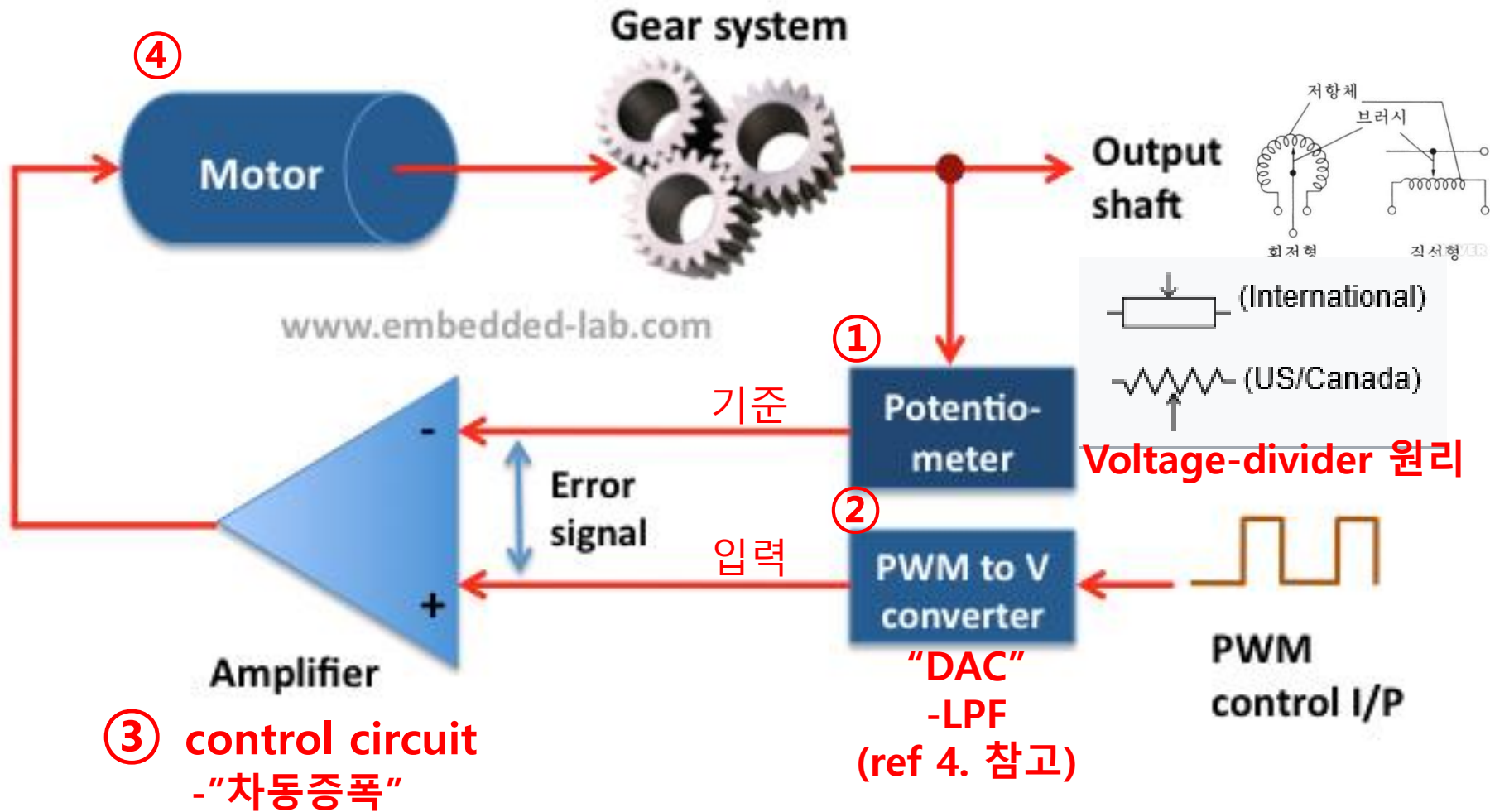
서보 모터란?

- 특정 위치로 이동하거나, 특정한 수치(속도 등)만큼 가동시킬 때 사용
- 모터로부터의 피드백을 통해 정확하게 제어할 수 있는 구조 : closed-loop system
- 모터와 기어박스 그리고 제어회로로 구성
- 자동화 생산 시스템, 로봇, 장난감, 가전제품 등 광범위하게 쓰인다.

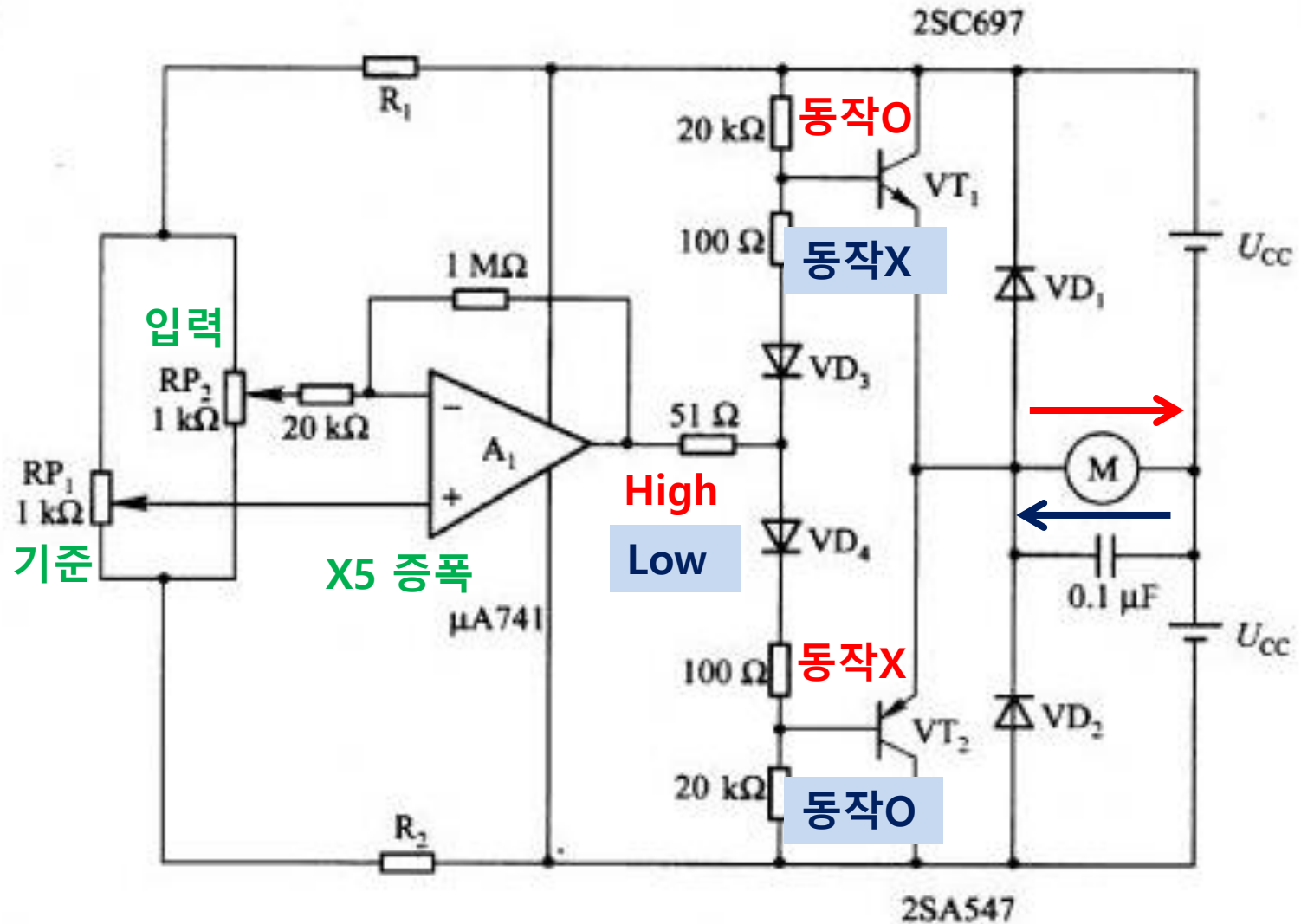
산업용 vs RC용



서보 모터 구조 및 작동원리



③ Control circuit 예시



PWM 신호 발생

SG90 9 g Micro Servo



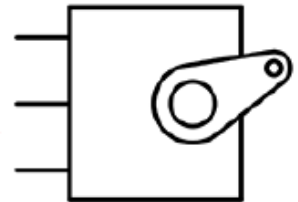
Specifications

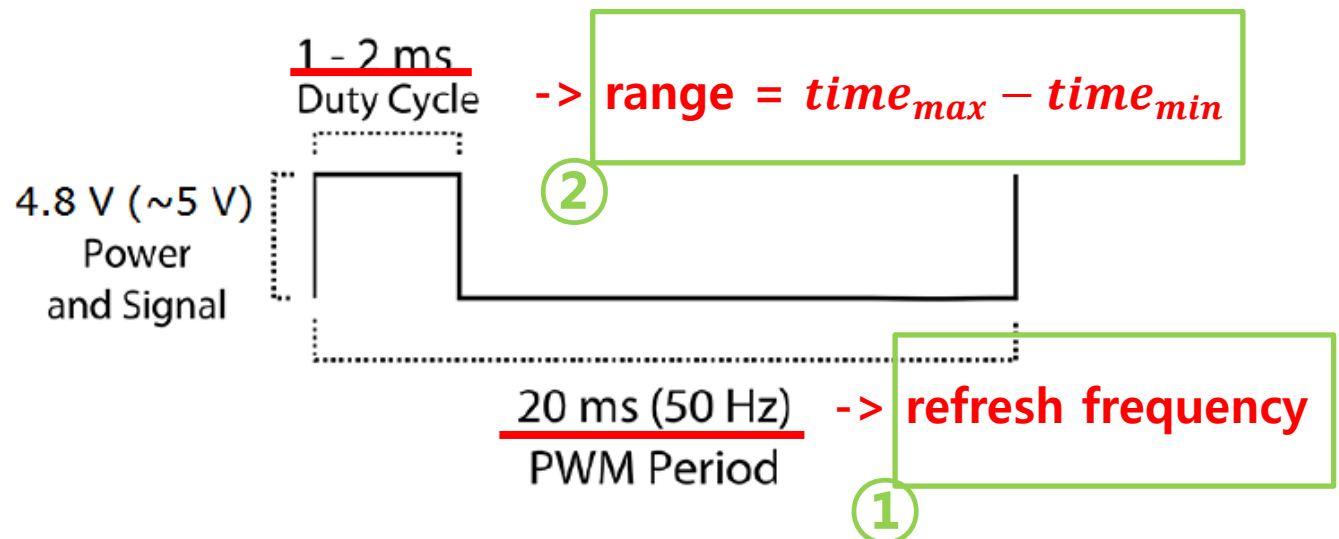
- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 μ s
- Temperature range: 0 °C – 55 °C

PWM=Orange ()

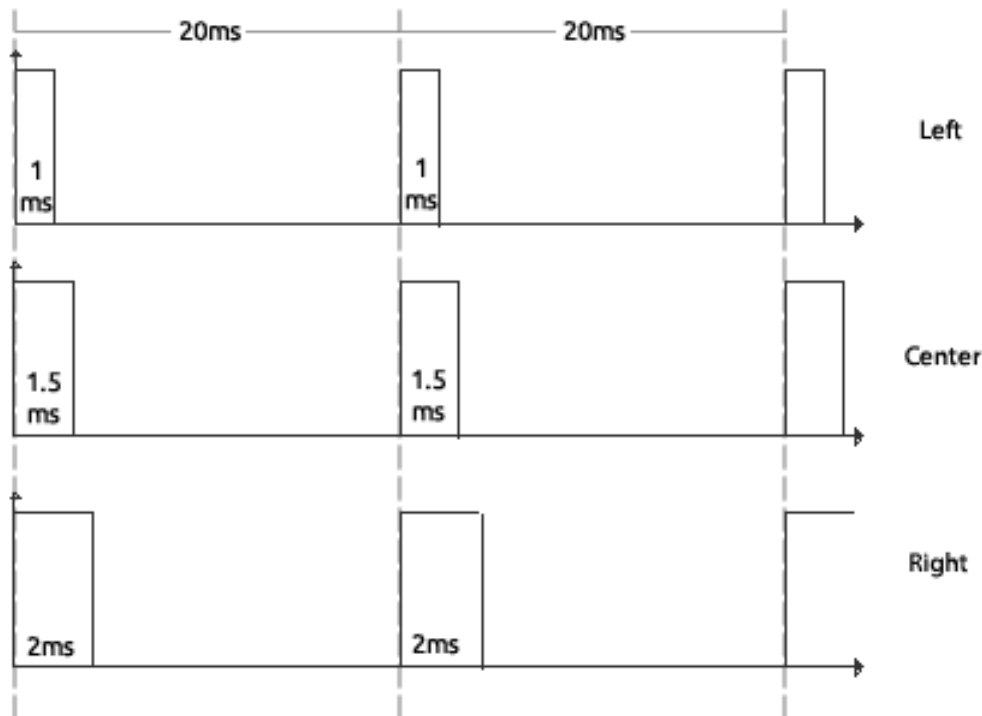
Vcc = Red (+)

Ground=Brown (-)





Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.



$$-90^\circ = 1\text{ms}$$

$$0^\circ = 1.5\text{ms}$$

$$90^\circ = 2\text{ms}$$

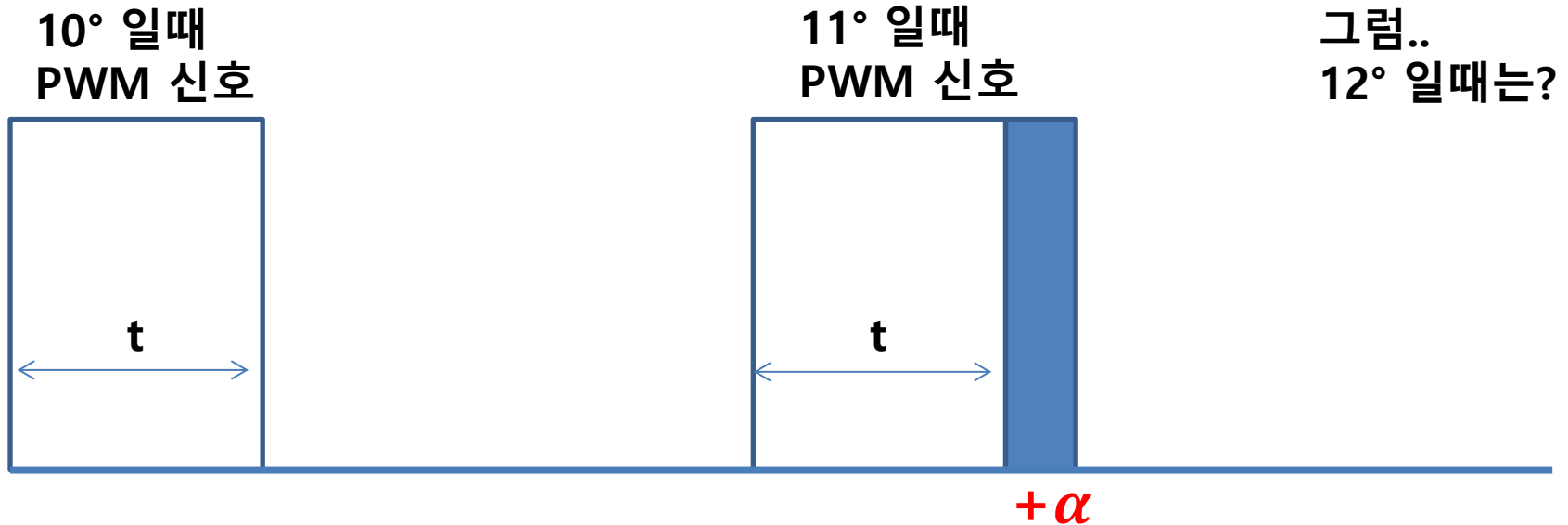
각 duty cycle 마다 각도가 정해져 있음!

③

-90° ~ 90° 제어 가능
(180°)

Q) “1°단위로 제어하고 싶다”

-> 10°와 11°를 구분



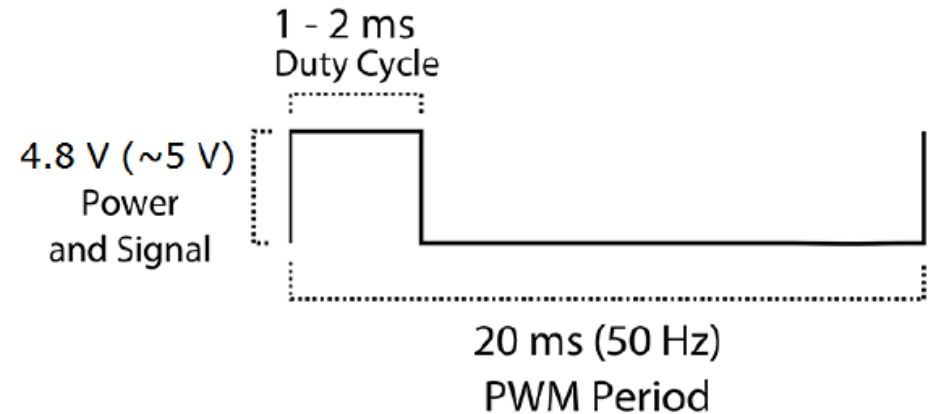
$+\alpha$ 를 기준 clock -> “counter”

기준 clock은 얼마로 해야 하나?

1) $refresh\ frequency = 20ms$

2) $range = time_{max} - time_{min}$
 $= 2ms - 1ms = 1ms$

3) $f_{needed} = \left(\frac{range}{resolution}\right)^{-1}$



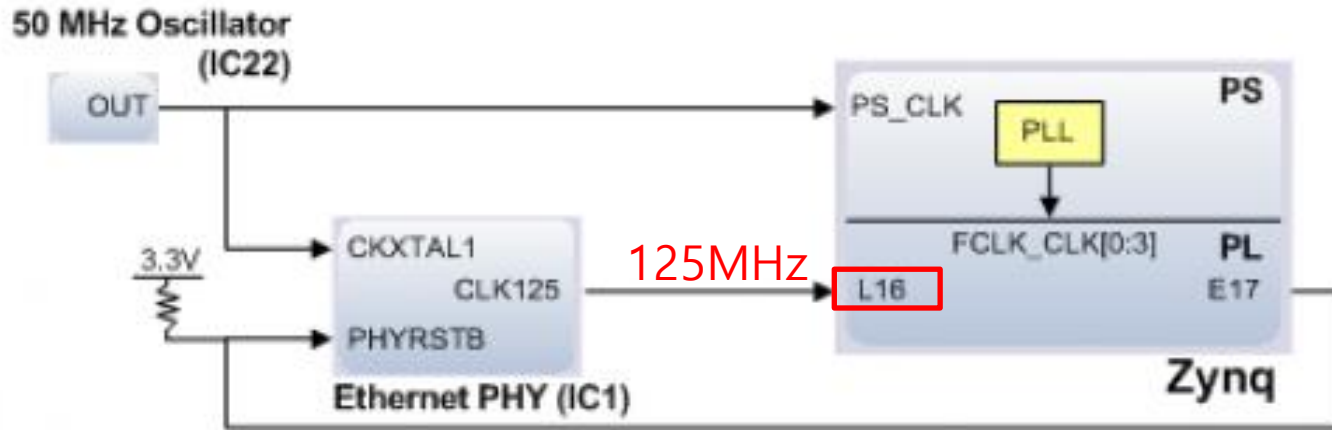
1°단위로 제어 -> resolution = 180

//그럼.. 10°단위로 제어하고 싶을 땐? -> resolution = 18

$f_{needed} = \left(\frac{1ms}{180}\right)^{-1} = 180kHz$ -> 이 clock 신호를 만들어 줘야 됨..

frequency_divider : clk_180kHz

ZYBO 외부 125MHz 클럭(L16핀)이용 -> 180kHz 클럭 생성



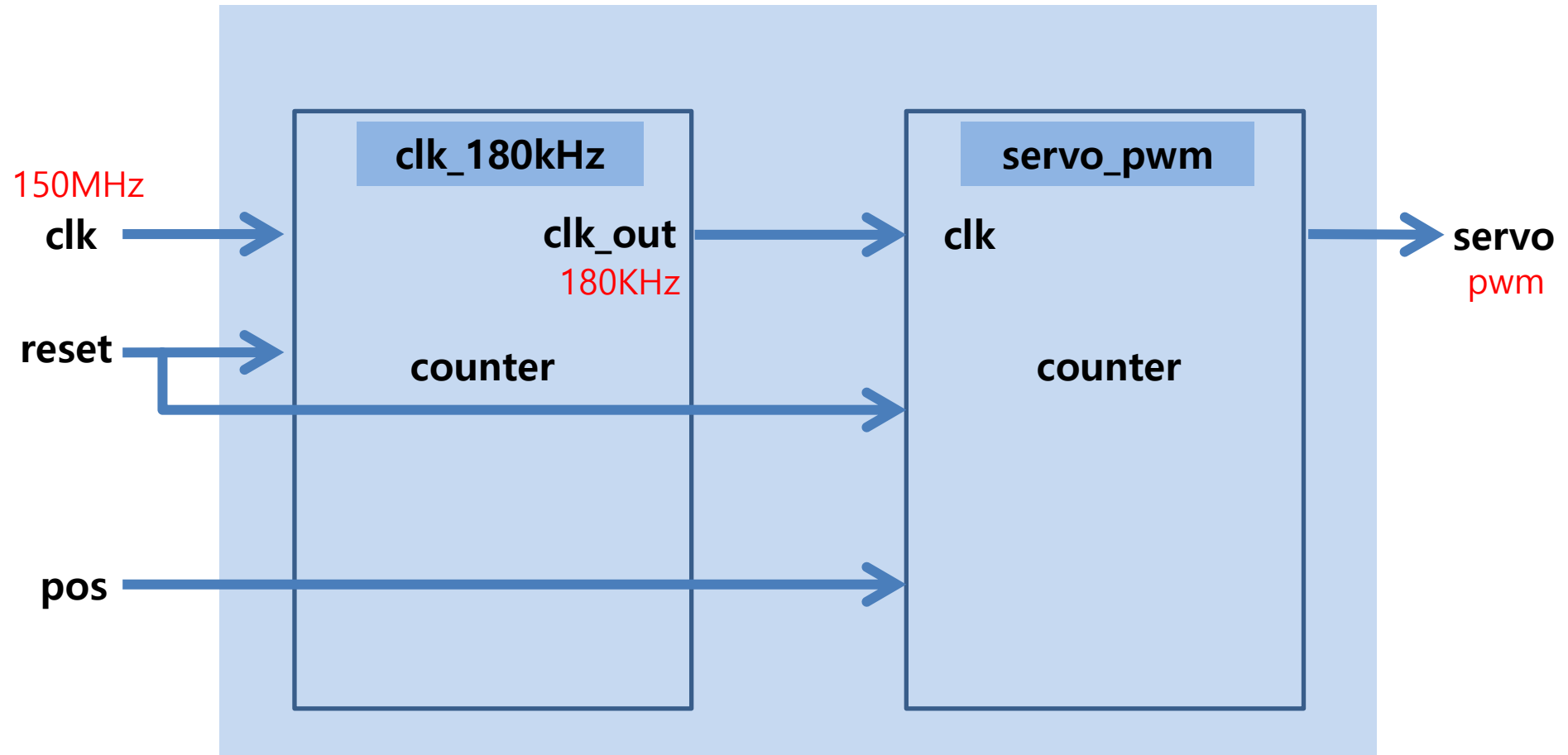
125MHz로 180kHz 만들기

$$\begin{aligned} \text{Scale} &= \frac{f_{in}}{f_{out}} \\ &= \frac{125\text{MHz}}{180\text{kHz}} = 694.4444 \dots \approx 694 \end{aligned}$$

-> 150MHz를 694번 씩 세면(=>counter)
180kHz가 만들어짐

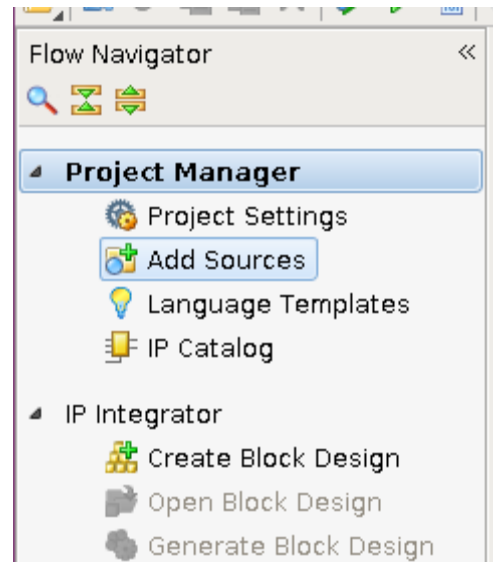
PWM 생성 flow

servo_pwm_clk180kHz

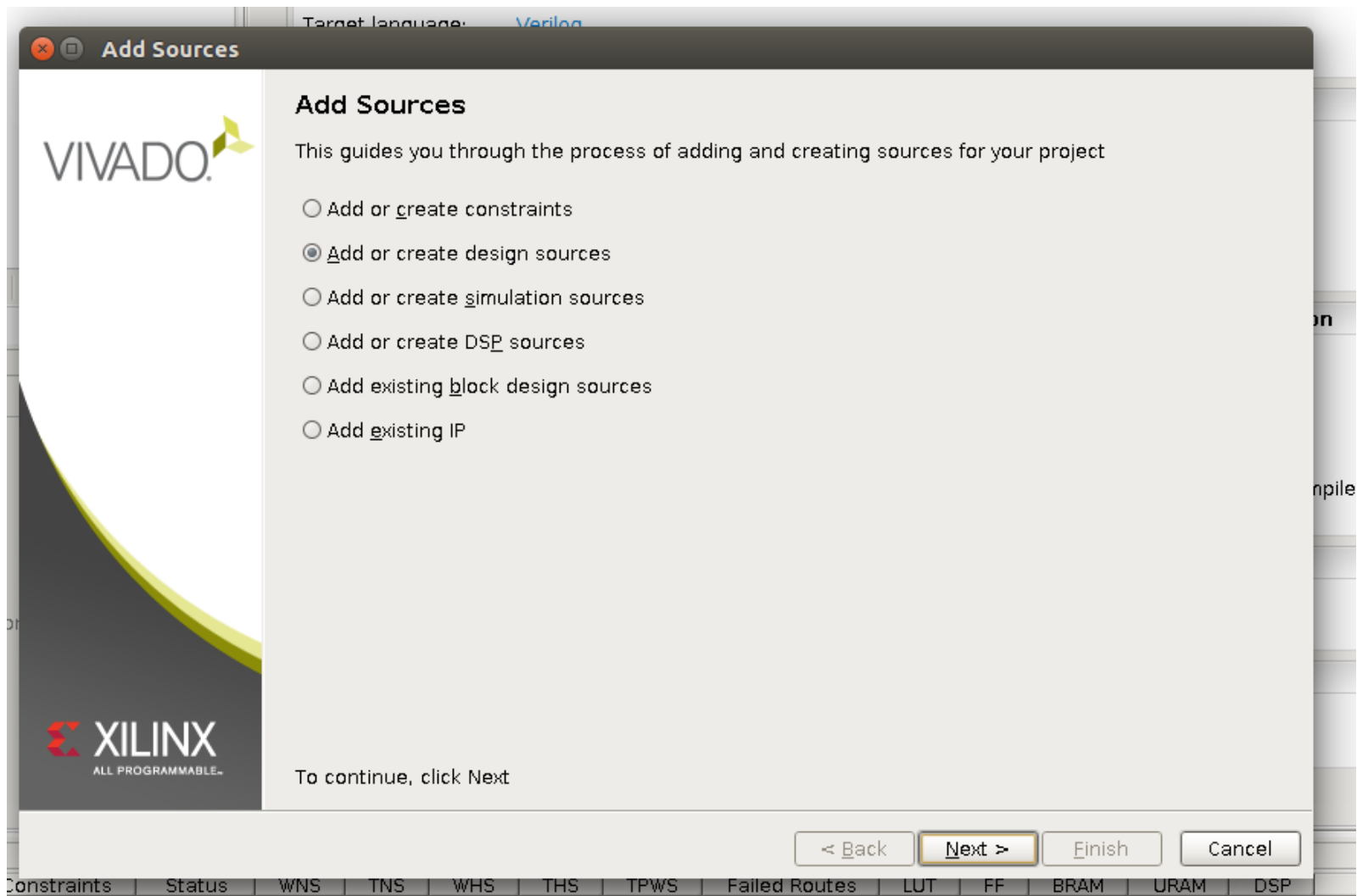


Vivado 실습

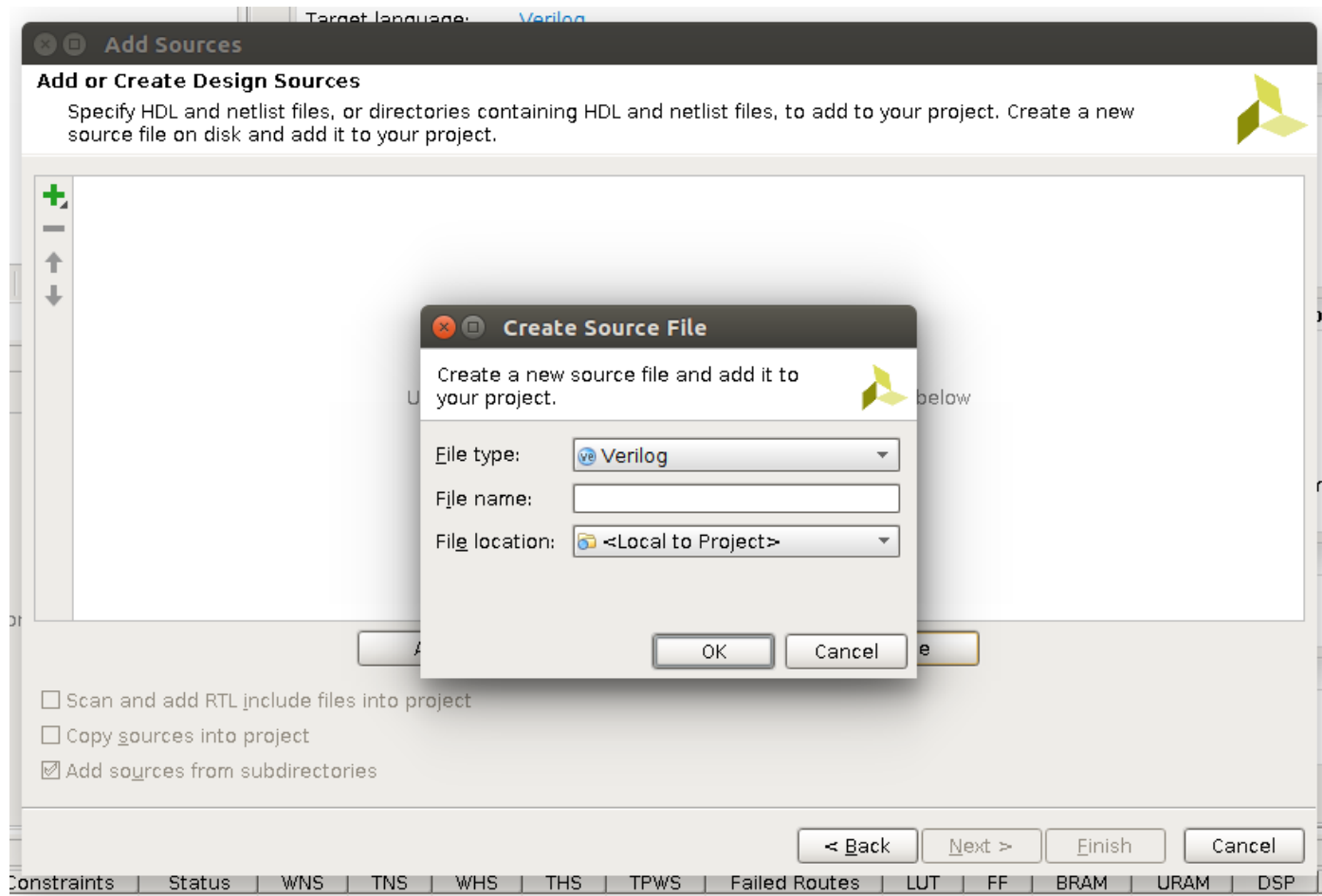
1. 새 project를 생성
 - Target language와 Simulator language는 VHDL 로 하고
board는 zybo로 설정
2. 왼쪽에 있는 Flow Navigator -> Project Manager -> Add Sources클릭



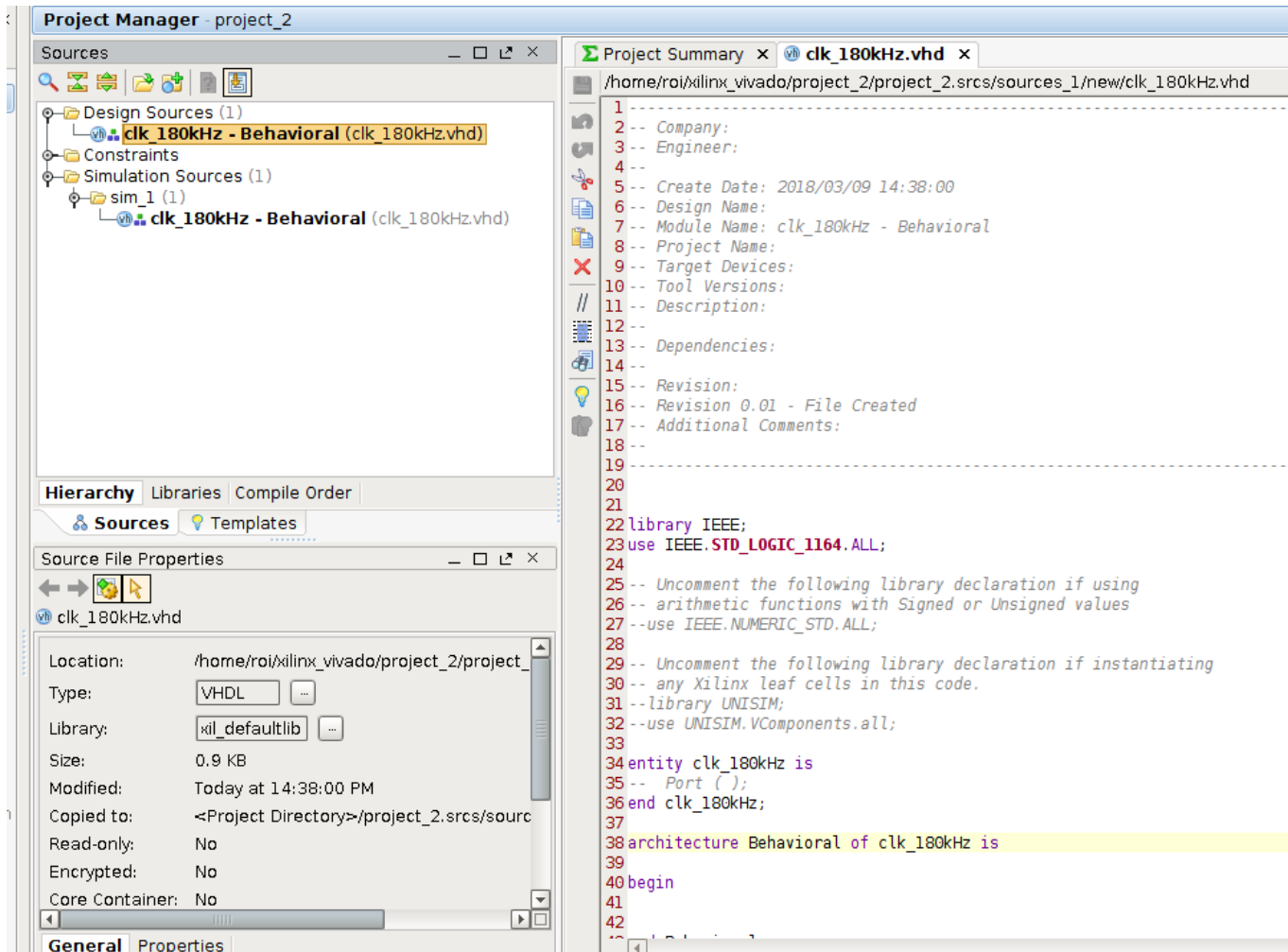
3. 'Add or create design sources' 체크 후 Next> 버튼



4. 'Create File'누르고 'Create Source File'창 뜨면 File name에 'clk_180kHz' 입력, 'Ok'버튼 누르고 나와서 'Finish'클릭



5. 'Define Module' 창이 뜨면 그냥 'Ok'클릭 -> 'Yes'클릭
6. Design Sources' -> clk_180kHz 더블 클릭해서 코드 입력.



clk_180kHz VHDL 코드

```
Project Summary x vh clk_180kHz.vhd * x
/home/roi/xilinx_vivado/project_2/project_2.srscs/sources_1/new/clk_180kHz.vhd
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20
21
22 library IEEE;                                --라이브러리 및 패키지 선언
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity clk_180kHz is                          --entity : 모듈 이름, 입/출력 port 설정
26 -- Port ( );
27     port(
28         clk_in : in std_logic;
29         reset : in std_logic;
30         clk_out : out std_logic
31     );
32
33 end clk_180kHz;
```

```

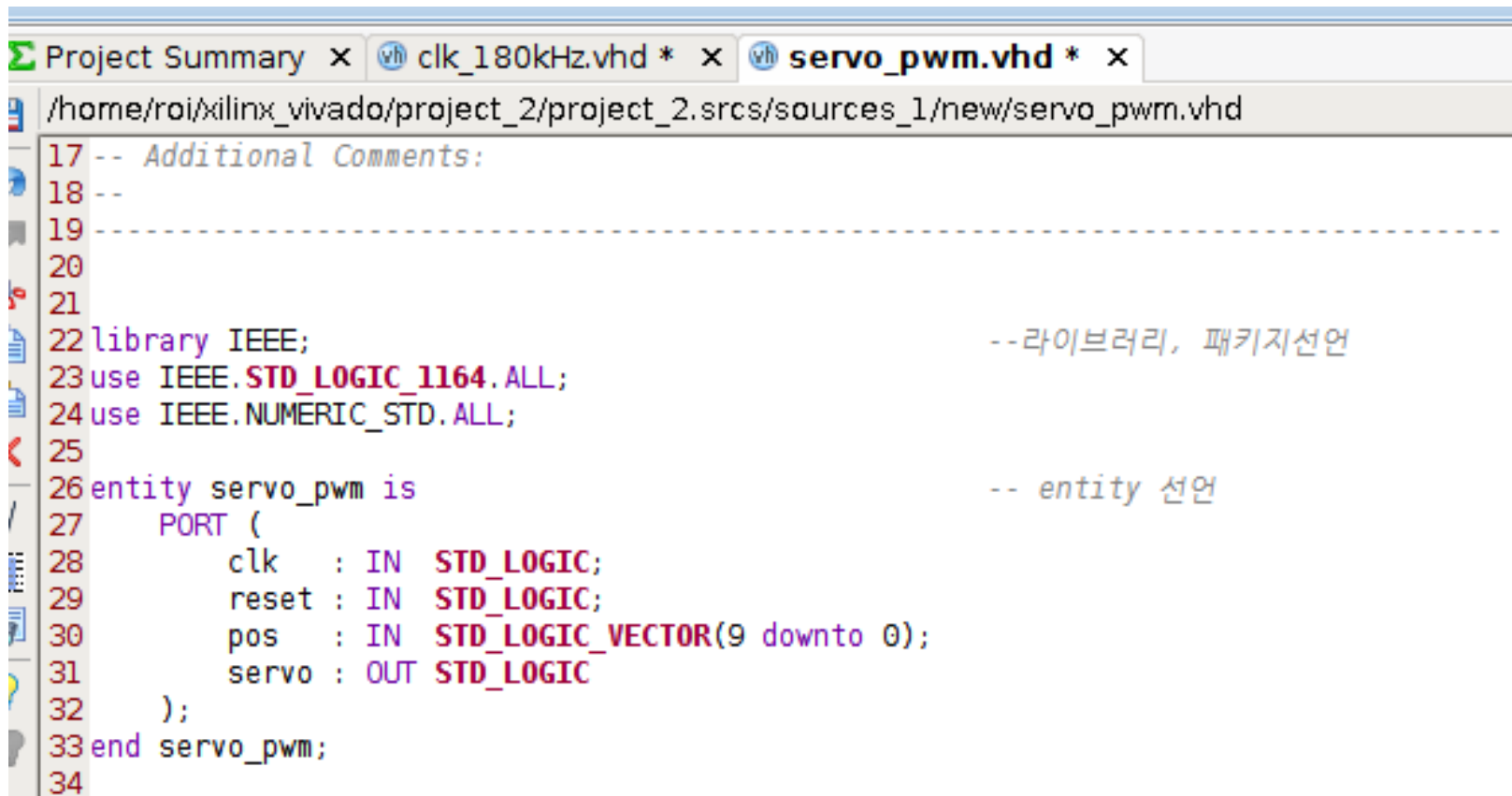
34
35 architecture Behavioral of clk_180KHz is
36     signal temporal : std_logic;
37     signal counter : integer range 0 to 346 :=0;
38 begin
39     frequency_divider : process (reset, clk_in)
40     begin
41         if(reset ='1') then
42             temporal <= '0';
43             counter <= 0;
44         elsif rising_edge(clk_in) then
45             if (counter = 346) then
46                 temporal <= NOT(temporal);
47                 counter <= 0;
48             else
49                 counter <= counter +1;
50             end if;
51         end if;
52     end process;
53
54     clk_out <= temporal;
55
56 end Behavioral;
57

```

--architecture
--선언부 : 데이터 타입, 신호 및 컴포넌트 등 선언
-- 구현부 : 회로 구현
-- reset 설정
-- counter = 346일때 까지 clk의 rising edge를 세겠다!
-- counter = 346이면 temporal을 반전
-- counter < 346일 때, 1씩 증가
-- temporal 신호를 clk_out으로!

위에서 150MHz를 694번 세어 준다고 했는데,,
Duty 비가 50%인 clock을 만들어 주기 위해
절반인 347(0부터 시작하니까 346이 되어야 함)
이 되면 temporal를 (0에서 1로, 1에서 0으로)
반전시켜 준다.

7. 2-6 반복해서 File name01 servo_pwm인 source 생성 후 코드 입력.



```
17 -- Additional Comments:
18 --
19 -----
20
21
22 library IEEE;                                -- 라이브러리, 패키지선언
23 use IEEE.STD_LOGIC_1164.ALL;
24 use IEEE.NUMERIC_STD.ALL;
25
26 entity servo_pwm is                            -- entity 선언
27     PORT (
28         clk    : IN  STD_LOGIC;
29         reset  : IN  STD_LOGIC;
30         pos    : IN  STD_LOGIC_VECTOR(9 downto 0);
31         servo  : OUT STD_LOGIC
32     );
33 end servo_pwm;
34
```

```

34
35 architecture Behavioral of servo_pwm is
36     signal cnt : unsigned(11 downto 0);
37     signal pwmi: unsigned(9 downto 0);
38 begin
39     pwmi <= unsigned(pos) + 180;
40 counter: process (reset, clk) begin
41     if (reset = '1') then
42         cnt <= (others => '0');
43     elsif rising_edge(clk) then
44         if (cnt = 3599) then
45             cnt <= (others => '0');
46         else
47             cnt <= cnt + 1;
48         end if;
49     end if;
50 end process;
51
52 servo <= '1' when (cnt < pwmi) else '0';
53 end Behavioral;
54

```

-- Counter, from 0 to 3599. 20ms를 만들기 위해서 180x20 = 3600이 필요
 -- Temporal signal used to generate the PWM pulse.
 -- 최소 duty cycle이 1ms가 되어야 하므로 180을 더해줌.
 -- Counter process, from 0 to 3599.
 -- reset 설정
 -- cnt=3599에 도달하면 cnt를 0으로!
 -- cnt < 3599 이면 +1
 -- pwm 출력

cnt를 3600(0부터 3599)번 세어주는 이유

- refresh frequency 에 해당하는 20ms를 만들어주기 위함
- 180kHz -> 1s : 180k 개를 세면 1s
 180 -> 1ms : 180개를 세면 1ms
 => 20ms를 만들어주기 위해서는 180x20 = 3600개를 세어야 함.

56줄에 보면 cnt < pwmi 일 때 servo로 출력이 '1'나감

- pwmi = 최소 duty cycle인 1ms(180개 세면 1ms이므로 180)
 + 원하는 각도의 이진 표현

8. 2-6 반복해서 File name이 servo_pwm_180kHz인 source 생성 후 코드 입력.

```
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 entity servo_pwm_clk180kHz is                                -- entity
26     PORT(
27         clk : IN  STD_LOGIC;
28         reset: IN  STD_LOGIC;
29         pos  : IN  STD_LOGIC_VECTOR(9 downto 0);
30         servo: OUT STD_LOGIC
31     );
32 end servo_pwm_clk180kHz;
33
34 architecture Behavioral of servo_pwm_clk180kHz is            --architecture
35     COMPONENT clk_180kHz                                       --clk_180kHz 의 인스턴스 생성
36     PORT(
37         clk      : in  STD_LOGIC;
38         reset    : in  STD_LOGIC;
39         clk_out: out STD_LOGIC
40     );
41     END COMPONENT;
42
```

```

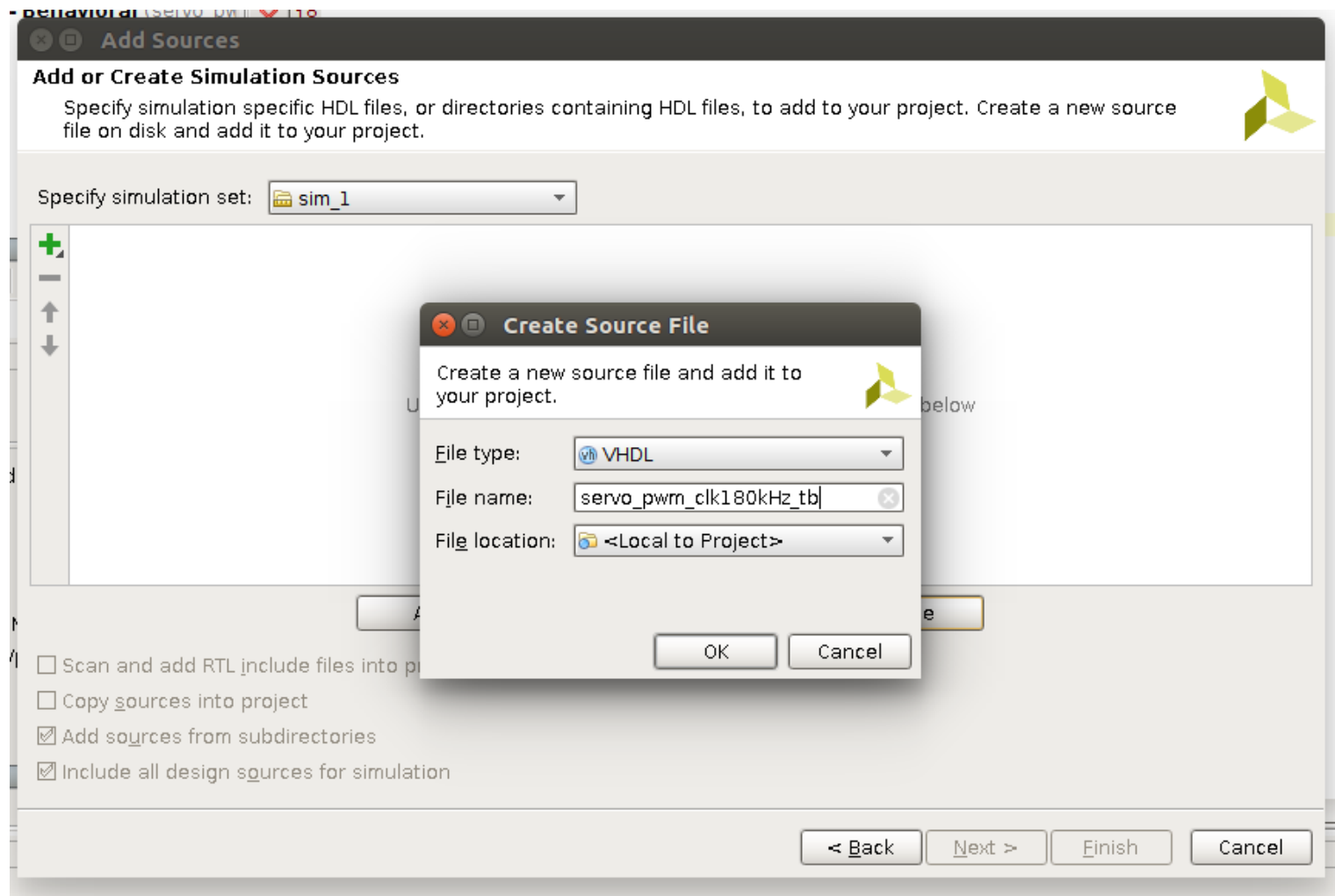
42
43 COMPONENT servo_pwm                                --servo_pwm의 인스턴스 생성
44     PORT (
45         clk    : IN  STD_LOGIC;
46         reset  : IN  STD_LOGIC;
47         pos    : IN  STD_LOGIC_VECTOR(9 downto 0);
48         servo  : OUT STD_LOGIC
49     );
50 END COMPONENT;
51
52 signal clk_out : STD_LOGIC := '0';
53 begin
54     clk180kHz_map: clk_180kHz PORT MAP(              --인스턴스 이름 : 하위레벨 entity 이름
55         clk=>clk, reset=>reset, clk_out=>clk_out      --포트 연결
56     );
57
58     servo_pwm_map: servo_pwm PORT MAP(               --인스턴스 이름 : 하위레벨 entity 이름
59         clk=>clk_out, reset=>reset, pos=>pos, servo=>servo --포트 연결
60     );
61 end Behavioral;
62

```

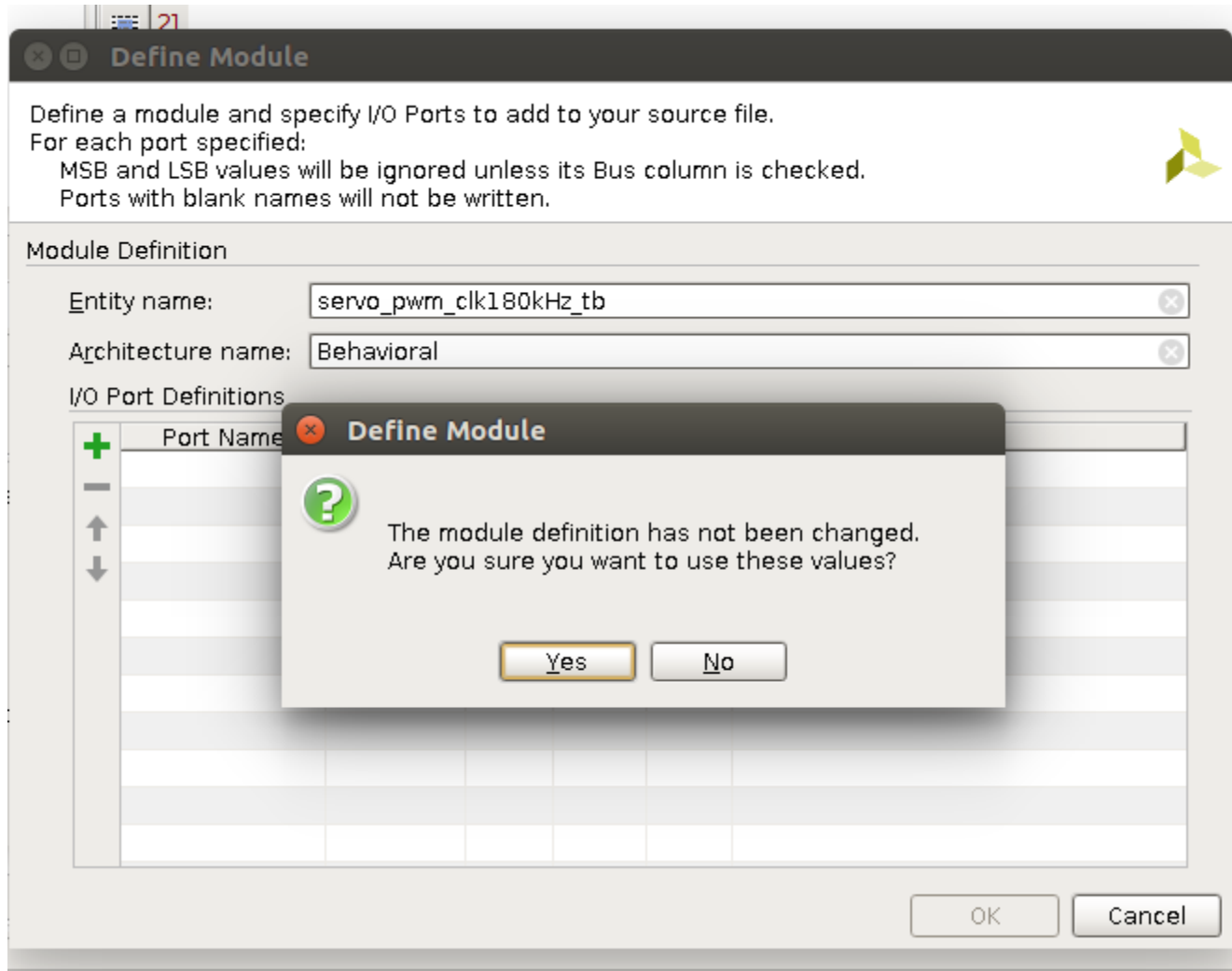
9. Flow Navigator > Project Manager> Add Sources 클릭

10. 'Add or create simulation sources'체크 후 Next>버튼

11. 'Create Source File'창에서 File name에 servo_pwm_clk180kHz_tb입력 후
'Ok'클릭 > 'Finish' 클릭



12. 'Define Module'창 에서 'Ok'클릭 > 'Yes' 클릭



13. 'Simulation Sources'>sim_1>servo_pwm_clk180kHz_tb 더블 클릭 후 코드 입력

```
rvo_pwm_clk180kHz_tb.vhd*
/home/roi/xilinx_vivado/project_2/project_2.srscs/sim_1/new/servo_pwm_clk180kHz_tb.vhd
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  -----
20
21
22  LIBRARY ieee;                                -- 라이브러리 및 패키지 선언
23  USE ieee.std_logic_1164.ALL;
24
25  ENTITY servo_pwm_clk180kHz_tb IS              -- 테스트 벤치는 port 선언이 필요 없음
26  END servo_pwm_clk180kHz_tb;
27
28  ARCHITECTURE behavior OF servo_pwm_clk180kHz_tb IS
29    -- Unit under test.
30    COMPONENT servo_pwm_clk180kHz              -- 검증하고자 하는 모듈을 인스턴스로 생성하기 위해 component 선언
31    PORT(
32        clk      : IN  std_logic;
33        reset    : IN  std_logic;
34        pos      : IN  std_logic_vector(9 downto 0);
35        servo    : OUT std_logic
36    );
37  END COMPONENT;
38
```

```

38
39 -- Inputs.
40 signal clk : std_logic := '0';
41 signal reset: std_logic := '0';
42 signal pos : std_logic_vector(9 downto 0) := (others => '0');
43 -- Outputs.
44 signal servo : std_logic;
45 -- Clock definition.
46 constant clk_period : time := 8 ns;
47 BEGIN
48 -- Instance of the unit under test.
49 uut: servo_pwm_clk180kHz PORT MAP (
50     clk => clk,
51     reset => reset,
52     pos => pos,
53     servo => servo
54 );
55
56
57 clk_process :process begin
58     clk <= '0';
59     wait for clk_period/2;
60     clk <= '1';
61     wait for clk_period/2;
62 end process;
63
64
65 stimuli: process begin
66     reset <= '1';
67     wait for 50ns;
68     reset <= '0';
69     wait for 50ns;
70     pos <= B"00_0000_0000";
71     wait for 20ms;
72     pos <= B"00_0011_1100";
73     wait for 20ms;
74     pos <= B"00_0111_1000";
75     wait for 20ms;
76     pos <= B"00_1011_0100";
77     wait;
78 end process;
79 END;

```

-- 생성한 인스턴스와 포트 연결할 시그널 선언

-- 외부 125MHz의 클럭 주기 $1/125\text{MHz} = 8\text{ns}$

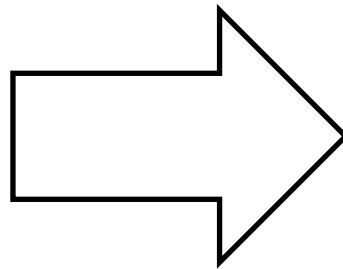
-- 인스턴스 생성

-- clock 프로세스 정의

-- Stimuli process : 검증하고자 하는 모듈의

-- 입력포트에 인가할 입력 신호들을 만들어 줌

1.5ms pwm 신호를 0도라 할 때



-90도 => 00_0000_0000

-30도 => 00_0011_1100

+30도 => 00_0111_1000

+90도 => 00_1011_0100

14. Ctrl+S 눌러 저장

15. Flow Manager>Simulation>Simulation Settings>Simulation 에서 runtime을 100ms로 설정 >Apply 클릭 > Ok 클릭

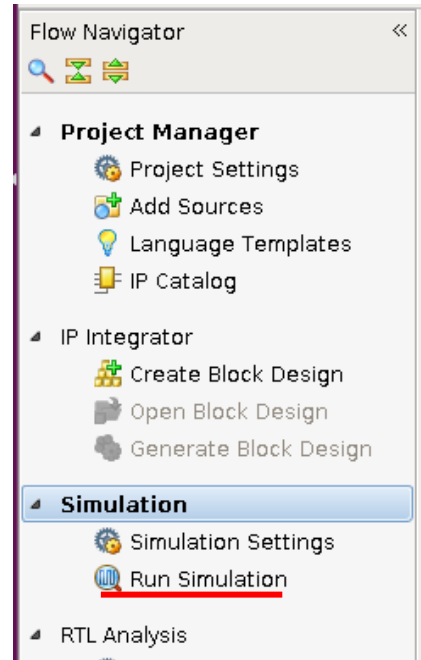
The screenshot displays the Xilinx Vivado IDE interface. On the left is the Flow Navigator, showing the project hierarchy with 'Simulation' > 'Simulation Settings' highlighted. The main window shows the Project Manager for 'project_2', listing sources like 'servo_pwm_clk180kHz - Behavioral' and 'sim_1'. The 'Project Settings' dialog is open, with the 'Simulation' tab selected. In the 'Simulation' tab, the 'Target simulator' is 'Vivado Simulator', 'Simulator language' is 'VHDL', and 'Simulation set' is 'sim_1'. The 'Simulation top module name' is 'servo_pwm_clk180kHz_tb'. The 'Clean up simulation files' checkbox is checked. The 'Compilation' tab is also visible, showing a table of simulation settings.

Compilation	Elaboration	Simulation	Netlist	Advanced
xsim.simulate.runtime*		100ms		
xsim.simulate.uut				
xsim.simulate.wdb				
xsim.simulate.saif				
xsim.simulate.saif_all_signals				<input type="checkbox"/>
xsim.simulate.xsim.more_options				

At the bottom of the 'Simulation' tab, there is a section for 'xsim.simulate.runtime*' with the text 'Specify simulation run time'.

The 'Tcl Console' at the bottom shows the command '# run 100ms' and the output: 'run: Time (s): cpu = 00:00:14 ; elapsed', 'xsim: Time (s): cpu = 00:00:14 ; elapsed', and 'INFO: [USF-XSim-96] XSim completed. Des', 'INFO: [USF-XSim-93] XSim simulation complete'.

16. Flow Manager>Simulation>Run Simulation > Run Behavioral Simulation 클릭
하여 시뮬레이션 시작



reference

1. <https://en.wikipedia.org/wiki/Servomotor>
2. www.thomas.co.kr/tec/download.php?fn=1.서보의%20종류.pdf@A1161682707
3. <http://embedded-lab.com/blog/lab-21-servo-motor-control/>
4. www.ti.com/lit/an/spraa88a/spraa88a.pdf
5. [http://www.seekic.com/circuit diagram/Basic Circuit/DC motor servo circuit composed of %CE%BCA741.html](http://www.seekic.com/circuit_diagram/Basic_Circuit/DC_motor_servo_circuit_composed_of_%CE%BCA741.html)
6. <https://www.codeproject.com/Articles/513169/Servomotor-Control-with-PWM-and-VHDL>
7. <https://reference.digilentinc.com/reference/programmable-logic/zybo/reference-manual>