



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Lab 8

타이머

1. 소개

하드웨어 타이머는 마이크로 컨트롤러 및 SoC 에서 흔히 볼 수 있는 주변 장치입니다. 일반적으로 정확한 타이밍을 생성하는 데 사용됩니다. 타이머는 종종 구성 가능한 고정 주파수에서 카운터를 증가 또는 감소시킨 다음 카운터가 0 또는 사전 정의된 값에 도달하면 프로세서를 중단합니다. 보다 정교한 타이머는 펄스 폭 변조 (PWM) 파형을 생성하여 모터의 속도 또는 빛의 밝기를 제어하는 등의 다른 기능을 수행할 수도 있습니다.

이 LAB 에서는 이전 LAB 과 유사한 구조를 사용하여 먼저 RVfpga 시스템에 포함된 타이머의 상위 수준 사양을 설명한 다음 하위 수준 구현을 설명합니다. 타이머를 사용하고 수정하는 방법을 보여주는 기본 및 고급 연습이 모두 제안됩니다.

2. RVfpga 시스템에 포함된 타이머의 고급 사양

이 섹션에서는 먼저 RVfpga 시스템에서 사용되는 타이머의 고급 사양을 분석한 다음, 이 주변 장치를 사용하는 한 가지 연습을 제안합니다.

A. 타이머 고급 사양

RVfpga 시스템에서 사용되는 타이머 모듈은 OpenCores (<https://opencores.org/projects/ptc>)에서 가져왔습니다. 패키지를 다운로드하면 모듈의 상위 수준 사양을 설명하는 문서가 제공됩니다.

(*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc/docs/ptc_spec.pdf*) 여기서는 타이머 모듈의 주요 작동 및 기능을 요약합니다. 그러나 전체 정보는 위 문서에서 찾을 수 있습니다.

타이머 모듈에는 다음과 같은 주요 기능이 있습니다.

- Wishbone 상호 연결 사용
- 32 비트 카운터/타이머 기능
- PWM/타이머/카운터 (PTC)의 단일 실행 또는 연속 실행
- 프로그래밍 가능한 PWM (펄스 폭 변조) 모드
- 타이머 기능을 위한 시스템 클럭 및 외부 클럭 소스
- HI/LO 참조 및 캡처 레지스터
- PWM 출력 드라이버를 위한 3 단계 제어
- PTC 기능으로 인해 CPU 가 중단될 수 있습니다.

타이머 모듈 사양 문서의 섹션 4에서는 타이머 모듈 내에서 사용할 수 있는 제어 및 상태 레지스터에 대해 설명하며, 각 레지스터는 서로 다른 주소에 할당됩니다 (표 1 참조). RVfpga 시스템에서 타이머의 기본 주소는 **0x80001200** 입니다.

표 1. 타이머 레지스터

Name	Address	Width	Access	Description
RPTC_CNTR	0x80001200	1-32	R/W	Main PTC 카운터
RPTC_HRC	0x80001204	1-32	R/W	PTC HI 참조/캡처 레지스터
RPTC_LRC	0x80001208	1-32	R/W	PTC LO 참조/캡처 레지스터
RPTC_CTRL	0x8000120C	9	R/W	제어 레지스터

RPTC_CNTR 레지스터는 실제 카운터 레지스터이며 모든 카운터/타이머 클럭 사이클에서 증가합니다.

RPTC_CTRL 레지스터는 타이머 모듈을 제어하는데 사용됩니다. 표 2는 각 비트의 기능을 보여줍니다.

RPTC_HRC 및 RPTC_LRC는 참조/캡처 레지스터로 사용됩니다.

표 2. RPTC_CTRL 비트

Bit	Access	Reset	Name & Description
0	R/W	0	EN 설정되면 RPTC_CNTR 이 증가합니다.
1	R/W	0	ECLK 클럭 신호를 선택합니다: 외부 클럭, ptc_ecgt (1) 또는 시스템 클럭 (0)을 통해서.
2	R/W	0	NEC 외부 클럭 (ptc_ecgt)의 네거티브/포지티브 엣지 및 Low/High 기간을 선택하는 데 사용됩니다.
3	R/W	0	OE PWM 출력 드라이버를 활성화 합니다.
4	R/W	0	SINGLE 설정되면 RPTC_CNTR 은 RPTC_LRC 값과 같은 값에 도달합니다. 해제되면 RPTC_CNTR 이 RPTC_LCR 레지스터의 값에 도달한 후 다시 시작됩니다.
5	R/W	0	INTE 설정되면 PTC 는 RPTC_CNTR 값이 RPTC_LRC 또는 RPTC_HRC 의 값과 같을 때 인터럽트를 발생시킵니다. 신호가 지워지면 인터럽트를 mask 됩니다.
6	R/W	0	INT 읽을 때 이 비트는 보류중인 인터럽트를 나타냅니다. 설정되면 인터럽트가 보류됩니다. 이 비트를 '1'로 쓰면 인터럽트 요청이 해제됩니다.
7	R/W	0	CNTRRST 설정되면 RPTC_CNTR 이 재설정됩니다. 지워지면 카운터가 정상적으로 작동합니다.
8	R/W	0	CAPTE 설정되면 RPTC_CNTR 이 RPTC_LRC 또는 RPTC_HRC 레지스터로 캡처됩니다. 해제되면 캡처 기능이 mask 됩니다.

작업: 타이머 모듈에서 RPTC_CNTR, RPTC_HRC, RPTC_LRC 및 RPTC_CTRL 레지스터 선언과 주소 정의 (각각 0x80001200, 0x80001204, 0x80001208 및 0x8000120C)를 찾습니다. 타이머 모듈은 *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc* 폴더에서 사용할 수 있습니다.

타이머는 다양한 모드에서 작동할 수 있습니다 (다음으로 이번 LAB 에서 사용할 모드에 대해 간략하게 설명합니다. 자세한 내용은 타이머 모듈 사양 문서의 섹션 3 참조):

- **Timer/Counter mode** (타이머/카운터 모드): 이 모드에서는 카운터가 활성화된 경우 (RPTC_CTRL [EN] = 1) 시스템 클럭 또는 외부 클럭 레퍼런스가 레지스터 RPTC_CNTR 을 증가 시킵니다. RPTC_CNTR 이 RPTC_LRC 와 같을 때 RPTC_CTRL [INTE]가 설정되면 RPTC_CTRL [INT]가 High 가 됩니다.
- **PWM 모드:** PWM (Pulse Width Modulation) 신호는 디지털 소스를 사용하여 아날로그 신호를 생성하는 방법입니다. PWM 신호는 동작을 정의하는 두 가지 값인 *듀티(duty) 사이클과 주파수*로 구성됩니다. 듀티

사이클은 신호가 높은 시간을 한 사이클을 완료하는 데 걸리는 총 시간의 백분율로 나타냅니다. 빈도는 해당 주기가 반복되는 빈도입니다. 디지털 신호를 충분히 빠른 속도로 켜고 끄고 특정 듀티 사이클을 사용하면 출력이 장치에 전원을 공급할 때 정전압 아날로그 신호처럼 동작하는 것처럼 보입니다. 예를 들어 듀티 사이클이 50 %이고 (사이클 시간의 절반이 높음) 3.3V 의 고전압 신호는 아날로그 부하에 1.67V (사이클 전체의 평균 전압)로 나타납니다. 듀티 사이클이 33 % 인 동일한 신호는 1.1V 로 나타납니다. PWM 모드에서 동작하려면 RPTC_CTRL [OE]를 설정해야 합니다. 레지스터 RPTC_HRC 및 RPTC_LRC 는 PWM 출력 신호의 높고 낮은 기간의 값으로 설정되어야 합니다. PWM 신호는 재설정 후 (RPTC_CNTR 의) 높은 RPTC_HRC 클럭 사이클이 되어야 합니다. 그리고 PWM 신호는 (RPTC_CNTR 의) 리셋 후 낮은 RPTC_LRC 클럭 사이클이 되어야 합니다.

3. 기본 실습

연습 1. 8 자리 7 세그먼트 디스플레이에 오름차순 카운트를 표시하는 프로그램을 작성하십시오. 이 값은 초당 약 한 번 변경되어야 하며 이 지연을 생성하려면 timer 모듈을 사용해야 합니다.

- 먼저 RISC-V 어셈블리 언어로 프로그램을 작성하고 Nexys A7 보드에서 실행합니다.
- 그런 다음 동일한 프로그램으로 Verilator 에서 시뮬레이션을 수행하십시오. 다음 신호를 추가할 수 있습니다. 시스템 클럭, 8 자리 7 세그먼트 디스플레이에 표시할 값을 저장하는 프로세서 레지스터, 타이머 레지스터 RPTC_CNTR, RPTC_LRC, RPTC_HRC 및 RPTC_CTRL.
- 이제 C 로 프로그램을 작성하고 Nexys A7 보드에서 실행합니다.
- RISC-V 어셈블리 프로그램의 파트 (b)와 같이 Verilator 에서 C 프로그램을 시뮬레이션 하십시오.

4. 타이머 초급 실습

이 섹션에서는 먼저 RVfpga 시스템에서 타이머 모듈의 초급 실습을 설명하고 먼저 모듈을 수정한 다음, Nexys A7 보드에서 사용할 수 있는 3 색 LED 를 제어하는 프로그램에서 사용하는 몇 가지 연습을 제안합니다. .

A. 타이머의 초급 실습

이전 LAB 에서 따랐던 방식과 유사하게 타이머 모듈의 분석을 여러 단계로 나눕니다.

- SweRVolFX SoC 에 새 모듈 통합 (그림 1 의 왼쪽 그림자 영역)
- 새 모듈과 SweRV EH1 Core 간의 연결 (그림 1 의 오른쪽 그림자 영역).

이전 LAB 과 달리 이 주변 장치 (타이머)는 Nexys A7 보드에 물리적으로 연결되지 않습니다. 타이머는 SweRVolFX 내부에 있습니다.

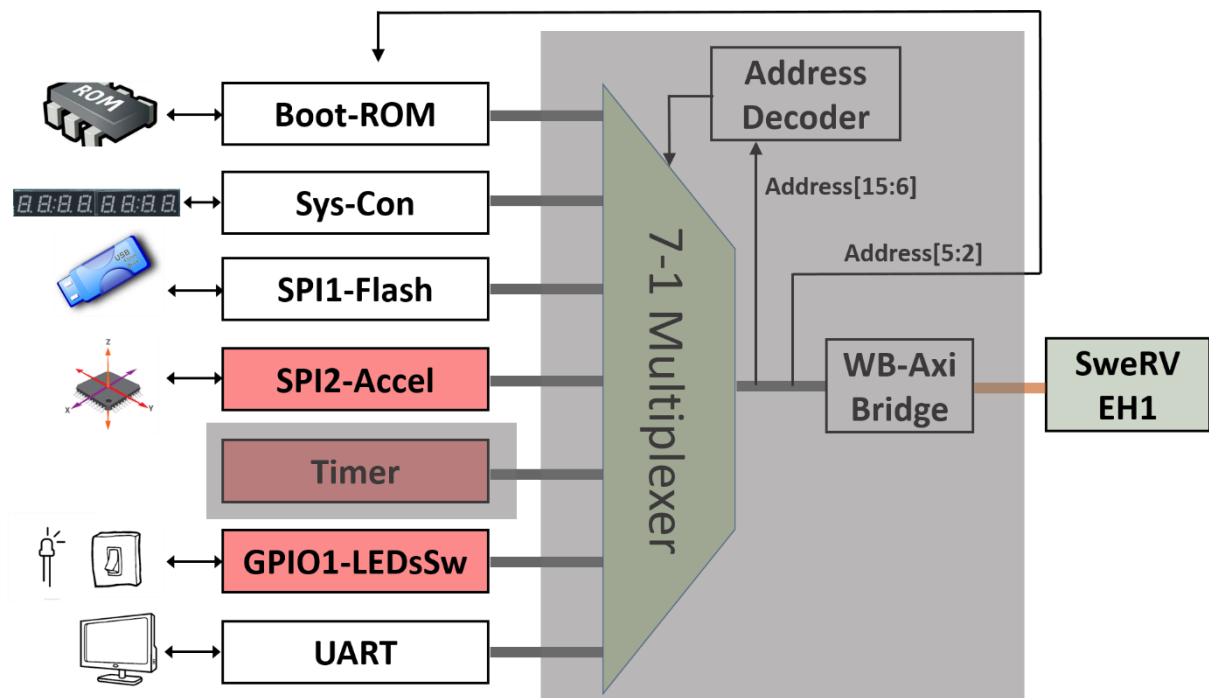


Figure 1. Timer module analysis in 2 phases

i. SoC 에 타이머 모듈 통합

swervolf_core 모듈 (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v*)의 361-379 행에서 타이머 모듈이 인스턴스화 됩니다 (그림 2 참조).

```

358 // PTC
359 wire      ptc_irq;
360
361 ptc_top timer_ptc(
362     .wb_clk_i      (clk),
363     .wb_rst_i      (wb_rst),
364     .wb_cyc_i      (wb_m2s_ptc_cyc),
365     .wb_adr_i      ({2'b0,wb_m2s_ptc_adr[5:2],2'b0}),
366     .wb_dat_i      (wb_m2s_ptc_dat),
367     .wb_sel_i      (4'b1111),
368     .wb_we_i      (wb_m2s_ptc_we),
369     .wb_stb_i      (wb_m2s_ptc_stb),
370     .wb_dat_o      (wb_s2m_ptc_dat),
371     .wb_ack_o      (wb_s2m_ptc_ack),
372     .wb_err_o      (wb_s2m_ptc_err),
373     .wb_inta_o     (ptc_irq),
374     // External PTC Interface
375     .gate_clk_pad_i (),
376     .capt_pad_i    (),
377     .pwm_pad_o     (),
378     .oen_padoen_o  ()
379 );

```

그림 2. 타이머 모듈 통합 (파일 *swervolf_core.v*).

평소와 같이 모듈의 인터페이스는 Wishbone 신호 (표 3)와 외부 I/O 신호 (표 4)의 두 블록으로 나눌 수 있습니다. Wishbone 신호를 통해 SweRV EH1 Core 는 컨트롤러/주변 장치 모델을 사용하여 타이머와 통신할 수 있습니다. 외부 I/O 신호는 타이머 모듈을 외부 장치와 연결합니다. 예를 들어, *pwm_pad_o*는 위에서 설명한

PWM 모드에서 작동할 때 PWM 출력 신호를 제공합니다 (타이머 모듈을 3 색 LED 와 연결하려면 연습 2 에서이 신호를 사용해야 합니다).

Table 1. Wishbone Signals

Port	Width	Direction	Description
wb_cyc_i	1	Inputs	유효한 버스 사이클 표시 (코어 선택)
wb_adr_i	15	Inputs	주소 입력
wb_dat_i	32	Inputs	데이터 입력
wb_dat_o	32	Outputs	데이터 출력
wb_sel_i	4	Inputs	데이터 버스의 유효한 바이트를 나타냅니다 (유효 사이클 동안, 이 신호는 0xf 여야 함).
wb_ack_o	1	Output	승인 출력 (정상적인 트랜잭션 종료를 나타냄)
wb_err_o	1	Output	오류 확인 출력 (비정상적인 트랜잭션 종료를 나타냄)
wb_rty_o	1	Output	미사용
wb_we_i	1	Input	High 인 경우, 트랜잭션 쓰기
wb_stb_i	1	Input	유효한 데이터 전송 사이클을 나타냅니다.
wb_inta_o	1	Output	인터럽트 출력

Table 2. External I/O Signals

Port	Width	Direction	Description
gate_clk_pad_i	1	Input	외부 클럭/게이트 입력
capt_pad_i	1	Input	입력 캡처
pwm_pad_o	1	Output	PWM 출력
oen_padoen_o	1	Output	PWM 출력 드라이버 활성화 (tri-state 또는 오픈 드레인 드라이버 용)

그림 2 의 365 행에서 볼 수 있듯이 Wishbone 버스 신호 (*wb_m2s_ptc_adr [5:2]*)에서 코어가 제공하는 주소의 비트 [5:2]는 사용 가능한 4 개의 레지스터 (Memory Mapped I/O) 중에 하나를 선택하는데 사용됩니다. 따라서 주소 0x80001200 에서 RPTC_CNTR 레지스터에 액세스하고 0x80001204 주소에서 RPTC_HRC 를 등록하고, 0x80001208 주소에서 RPTC_LRC 를 등록하고, 0x8000120C 주소에서 RPTC_CTRL 을 등록할 수 있습니다.

ii. 타이머와 SweRV EH1 Core 간의 연결

이전 LAB 에서 설명한 것처럼 장치 컨트롤러는 멀티플렉서를 통해 SweRV EH1 Core 와 연결됩니다 (그림 1). 7:1 멀티플렉서 (그림 3)는 *[RVfpgaPath]/RVfpga/src* 파일의 104-205 행에서 (*/SweRVVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.vh*) 인스턴스화되는 *[RVfpgaPath]/RVfpga/src/SweRVVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v* 파일에서 구현됩니다. 이 마지막 파일은 **swervolf_core** 모듈의(*[RVfpgaPath]/RVfpga/src/SweRVVolfSoC/swervolf_core.v*) 168 행에 포함되어 있습니다.

```

108 wb_mux
109 #(.num_slaves (7),
110 .MATCH_ADDR ({32'h00000000, 32'h00001000, 32'h00001040, 32'h00001100, 32'h00001200, 32'h00001400, 32'h00002000}),
111 .MATCH_MASK ({32'hffffff00, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffff00}))
112 wb_mux_io
113 (.wb_clk_i (wb_clk_i),
114 .wb_rst_i (wb_rst_i),
115 .wbm_adr_i (wb_io_adr_i),
116 .wbm_dat_i (wb_io_dat_i),
117 .wbm_sel_i (wb_io_sel_i),
118 .wbm_we_i (wb_io_we_i),
119 .wbm_cyc_i (wb_io_cyc_i),
120 .wbm_stb_i (wb_io_stb_i),
121 .wbm_cti_i (wb_io_cti_i),
122 .wbm_bte_i (wb_io_bte_i),
123 .wbm_dat_o (wb_io_dat_o),
124 .wbm_ack_o (wb_io_ack_o),
125 .wbm_err_o (wb_io_err_o),
126 .wbm_rty_o (wb_io_rty_o),
127 .wbs_adr_o ({wb_rom_adr_o, wb_sys_adr_o, wb_spi_flash_adr_o, wb_spi_accel_adr_o, wb_ptc_adr_o, wb_gpio_adr_o, wb_uart_adr_o}),
128 .wbs_dat_o ({wb_rom_dat_o, wb_sys_dat_o, wb_spi_flash_dat_o, wb_spi_accel_dat_o, wb_ptc_dat_o, wb_gpio_dat_o, wb_uart_dat_o}),
129 .wbs_sel_o ({wb_rom_sel_o, wb_sys_sel_o, wb_spi_flash_sel_o, wb_spi_accel_sel_o, wb_ptc_sel_o, wb_gpio_sel_o, wb_uart_sel_o}),
130 .wbs_we_o ({wb_rom_we_o, wb_sys_we_o, wb_spi_flash_we_o, wb_spi_accel_we_o, wb_ptc_we_o, wb_gpio_we_o, wb_uart_we_o}),
131 .wbs_cyc_o ({wb_rom_cyc_o, wb_sys_cyc_o, wb_spi_flash_cyc_o, wb_spi_accel_cyc_o, wb_ptc_cyc_o, wb_gpio_cyc_o, wb_uart_cyc_o}),
132 .wbs_stb_o ({wb_rom_stb_o, wb_sys_stb_o, wb_spi_flash_stb_o, wb_spi_accel_stb_o, wb_ptc_stb_o, wb_gpio_stb_o, wb_uart_stb_o}),
133 .wbs_cti_o ({wb_rom_cti_o, wb_sys_cti_o, wb_spi_flash_cti_o, wb_spi_accel_cti_o, wb_ptc_cti_o, wb_gpio_cti_o, wb_uart_cti_o}),
134 .wbs_bte_o ({wb_rom_bte_o, wb_sys_bte_o, wb_spi_flash_bte_o, wb_spi_accel_bte_o, wb_ptc_bte_o, wb_gpio_bte_o, wb_uart_bte_o}),
135 .wbs_dat_i ({wb_rom_dat_i, wb_sys_dat_i, wb_spi_flash_dat_i, wb_spi_accel_dat_i, wb_ptc_dat_i, wb_gpio_dat_i, wb_uart_dat_i}),
136 .wbs_ack_i ({wb_rom_ack_i, wb_sys_ack_i, wb_spi_flash_ack_i, wb_spi_accel_ack_i, wb_ptc_ack_i, wb_gpio_ack_i, wb_uart_ack_i}),
137 .wbs_err_i ({wb_rom_err_i, wb_sys_err_i, wb_spi_flash_err_i, wb_spi_accel_err_i, wb_ptc_err_i, wb_gpio_err_i, wb_uart_err_i}),
138 .wbs_rty_i ({wb_rom_rty_i, wb_sys_rty_i, wb_spi_flash_rty_i, wb_spi_accel_rty_i, wb_ptc_rty_i, wb_gpio_rty_i, wb_uart_rty_i}));
139
140 endmodule

```

CPU/Controller Signals

Peripheral Signals

그림 3. 7-1 CPU 와 연결된 주변 장치를 선택하는 멀티플렉서 (파일 wb_intercon.v)

멀티플렉서는 주소 (110-111 행)에 따라 CPU (wb_io_* 신호 – 그림 3의 115-126 행)를 하나의 주변 장치 (그림 3의 127-138 행)의 Wishbone 버스와 연결하여 읽거나 쓸 주변 장치를 선택합니다. 예를 들어 CPU에서 생성한 주소가 0x80001200-0x8000123F 범위이면 타이머 모듈이 선택되어 wb_io_* 신호는 wb_ptc_* 신호와 연결됩니다.

5. 고급 실습

연습 2. 타이머의 PWM 출력 신호 (pwm_pad_o)를 Nexys A7 보드에서 사용할 수 있는 2개의 3색 LED 중 하나에 연결하도록 RVfpgaNexys를 수정합니다. Labs 6 및 7에서 수정하여 업데이트된 RVfpgaNexys 시스템에 이 새로운 기능을 추가하는 것이 좋습니다.

- Digilent는 Nexys A7 보드에서 사용할 수 있는 3색 LED에 대한 다음 정보를 제공합니다:
<https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual>
- 위 문서를 요약하면 보드에는 2개의 3색 LED가 있습니다. 각 3색 LED에는 세 개의 작은 내부 LED (빨간색, 파란색, 녹색)의 음극을 구동하는 세 개의 입력 신호가 있습니다. 이 High 중 하나를 구동하면 각각의 내부 LED가 켜집니다. 3색 LED는 현재 조명중인 내부 LED의 조합에 따라 색상을 방출합니다. 예를 들어, 빨간색과 파란색을 높이면 보라색이 발산됩니다. Digilent는 3색 LED를 구동할 때 펄스 폭 변조 (PWM) 사용을 적극 권장합니다. 입력을 안정된 논리 '1'로 구동하면 LED가 불편할 정도로 밝은 수준으로 켜집니다. 3색 신호가 50% 이상의 듀티 사이클로 구동되지 않도록 함으로써 이를 방지할 수 있습니다. 또한 PWM을 사용하면 3색 LED의 잠재적인 색상 팔레트가 크게 확장됩니다. 각 색상의 듀티 사이클을 50%에서 0% 사이에서 개별적으로 조정하면 서로 다른 색상이 서로 다른 강도로 조명되어 사실상 모든 색상을 표시할 수 있습니다.
- 이미 SweRVofx에 포함된 것을 기반으로 세 개의 새로운 타이머 모듈을 만듭니다. 각 색상 (빨간색, 파란색 및 녹색)은 각기 다른 전압을 받을 수 있도록 다른 타이머 모듈로 구동되어야 합니다.
- 각 새 타이머의 레지스터를 메모리에 매핑하려면 다음 주소 범위를 사용합니다.
 - Timer-2: 0x80001240-0x8000127F
 - Timer-3: 0x80001280-0x800012BF
 - Timer-4: 0x800012C0-0x800012FF
- 이 경우 주변 장치를 선택하는 멀티플렉서에 3개의 새 항목을 추가해야 합니다 (그림 1).

- 3 가지 색상이 다음 보드 핀에 연결되어 있다는 점을 고려하여 constraint 파일을 수정해야 합니다.
 - iv. LED16_B \leftrightarrow PIN R12
 - v. LED16_G \leftrightarrow PIN M16
 - vi. LED16_R \leftrightarrow PIN N15

연습 3. 16 개의 스위치가 제공하는 값을 사용하여 3 색 LED 를 제어하기 위해 새로운 주변 장치를 사용하는 프로그램을 구현합니다. 가장 오른쪽에 있는 5 개의 스위치를 사용하여 파란색의 듀티 사이클을 조정하고, 다음 5 개의 스위치를 사용하여 녹색의 듀티 사이클을 조정하고, 다음 5 개의 스위치를 사용하여 빨간색의 듀티 사이클을 조정합니다. (가장 왼쪽 스위치는 사용되지 않습니다.)

- a. 먼저 RISC-V 어셈블리로 프로그램을 작성합니다.
- b. 다음으로 C 로 프로그램을 작성하십시오.