고급디지털회로설계

Design Project

PCFG 설계 보고서

2023.06.01(목)

12조

20161454 김규현

20181536 엄석훈

목차

[1. 제목 - 4 -](#_Toc136522832)

[2. 설계 목적 - 4 -](#_Toc136522833)

[3. 설계 내용 및 과정 - 4 -](#_Toc136522834)

[1) 전체 Block Diagram - 4 -](#_Toc136522835)

[(1) 사용되는 counter들 - 5 -](#_Toc136522836)

[(2) PC latch & Address Decoder - 6 -](#_Toc136522837)

[(3) Signal Controller overview - 7 -](#_Toc136522838)

[2) System Clock Setting & 8254 Reset - 8 -](#_Toc136522839)

[3) Software Reset - 8 -](#_Toc136522840)

[4) PC mode - 8 -](#_Toc136522841)

[(1) PC Write - 9 -](#_Toc136522842)

[(2) PC Read - 11 -](#_Toc136522843)

[5) DA mode - 12 -](#_Toc136522844)

[6) AD mode - 15 -](#_Toc136522845)

[(1) AD RAM write 시작 ~ 끝 - 16 -](#_Toc136522846)

[(2) PC RAM write 시작 ~ 끝 - 17 -](#_Toc136522847)

[7) ADR mode - 19 -](#_Toc136522848)

[8) Option Mode - 21 -](#_Toc136522849)

[(1) Step 1 - 21 -](#_Toc136522850)

[(2) Step 2 - 24 -](#_Toc136522851)

[(3) Step 3 - 25 -](#_Toc136522852)

[4. 평가 및 분석 (Simulation 및 시연 분석) - 28 -](#_Toc136522853)

[1) PC latch & Address Decoder - 28 -](#_Toc136522854)

[2) System Clock Setting & 8254 Reset - 28 -](#_Toc136522855)

[3) PC mode - 29 -](#_Toc136522856)

[(1) PC Write - 29 -](#_Toc136522857)

[(2) PC Read - 30 -](#_Toc136522858)

[4) DA mode - 31 -](#_Toc136522859)

[(1) DA Start - 31 -](#_Toc136522860)

[(2) DA Stop - 32 -](#_Toc136522861)

[5) AD mode - 32 -](#_Toc136522862)

[(1) AD RAM write 시작 ~ 끝 - 32 -](#_Toc136522863)

[(2) PC RAM write 시작 ~ 끝 - 33 -](#_Toc136522864)

[6) ADR mode - 34 -](#_Toc136522865)

[7) Option Mode - 34 -](#_Toc136522866)

[(1) Step 1 - 34 -](#_Toc136522867)

[(2) Step 2 - 36 -](#_Toc136522868)

[(3) Step 3 - 36 -](#_Toc136522869)

[8) 시연 분석 - 37 -](#_Toc136522870)

[5. 토의 - 39 -](#_Toc136522871)

[1) 제시된 주요 설계 요소와 제한 요소들을 만족하는 설계를 하였는지 논의 - 39 -](#_Toc136522872)

[2) Top-down 방식으로 설계하였는지 논의 - 39 -](#_Toc136522873)

[3) 임의의 입력에 대해서도 안정적으로 동작하는 논의 - 40 -](#_Toc136522874)

[4) logic량은 최적화되었는지 논의 - 40 -](#_Toc136522875)

[5) 설계 결과를 실제 제품개발에 사용될 수 있는지를 검토 - 40 -](#_Toc136522876)

[6) 설명서 구조의 단점 및 보완사항 논의 - 41 -](#_Toc136522877)

[6. 참조 문헌 - 41 -](#_Toc136522878)

# 제목

* PC기반 Function Generator (PCFG)

# 설계 목적

* 설계에서 주어진 기능의 구현을 통해 디지털 회로 설계 능력을 배양한다.
* Block diagram으로 회로를 도식화하고 Timing diagram을 통해 회로의 동작을 설계하는 능력을 배양한다.
* Xilinx Vivado를 사용하여 회로를 설계하고 검증하는 능력을 배양한다.

# 설계 내용 및 과정

## 전체 Block Diagram

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 PCFG 전체 block diagram

이번 프로젝트에서 설계한 PC based Function Generator (PCFG)는 위와 같은 block diagram을 가진다. PC에서 USB interface을 통해 PC에서 Parallel signal로 FPGA에 m\_cmd\_data, m\_wen, m\_ren, m\_oe\_b, m\_address, m\_data 등의 데이터를 보내게 된다. 이들을 이용하여 다양한 모드에 대한 기능을 수행하게 되고, 출력 데이터는 각 모드에 따라 inout 포트인 m\_data 포트를 이용해 PC로 출력하거나, m\_dac\_d 포트를 이용해 오실로스코프로 출력할 수 있다.

### 사용되는 counter들

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

그림 전체 block diagram에 표기되지 않은 counter들

위의 전체 block diagram에서는 표기되지 않은 counter들을 이용하여 synchronous SRAM에 데이터를 쓰게 된다. PC RAM에는 A, B port 공용으로 쓰이는 카운터 1개, AD RAM은 A port와 B port에 각각 이용되는 카운터 2개, Option RAM 역시 A port와 B port에 각각 이용되는 카운터 2개가 있다. RAM에 이용되는 카운터는 5개이고, 이후 Option mode step 1에서 이용되는 별도의 delayed counter를 두어 총 6개의 카운터를 이용하였다. 위의 카운터들은 active low인 reset\_b와 count enable (ce) 신호가 있으며, load enable 기능은 없다.

### PC latch & Address Decoder

이번 프로젝트에서는 FPGA의 setup time을 충분히 보장하기 위해 PC로 들어오는 data를 PC latch를 이용하게 된다. PC는 48MHz 클락을, FPGA는 40MHz 클락을 이용하기 때문에 PC에서 들어오는 신호가 FPGA에서는 setup time을 충분히 보장하지 못할 수 있으므로, PC latch를 이용한 신호들을 이용하게 된다. PC latch의 출력 신호로는 s\_cmd\_data, s\_wen, s\_ren, s\_oe\_b, s\_address가 있다. PC latch가 필요한 이유와 그의 기능에 대해서는 시뮬레이션 결과를 통해 더 자세히 확인할 수 있다.

Address decoder에서는 s\_address, s\_cmd\_data 신호를 입력 신호로 이용하여 어떠한 모드를 수행할 것인지를 알려주는 출력 신호를 내놓게 된다. 예를 들어, s\_address = 150H이고, s\_cmd\_data = ‘1’이면, da\_start\_addr 신호가 ‘1’로 출력되는 형태이다. 이들 출력 신호는 signal controller에 입력되어, 어떠한 모드를 수행할 것인지를 알려주는 기준이 된다. Pcfg\_top.vhd 파일에서는 top 파일의 architecture body 내의 신호임을 표기하기 위해, address decoder 출력 신호 앞에 ‘s\_’을 붙이게 되었다. Address decoder의 출력 신호와 이에 해당하는 모드에 대한 설명을 표로 정리하면 아래와 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mode** | **ADDRESS (9bits)** | | **Description** |
| **Address Decoder**  **Output Signal** | **Hex** |
| **System Clock Setting** | s\_pcs\_addr | 180H~183H | 8254 Setting |
| **Software Reset** | s\_reset\_addr | 170H | Software Reset (모드 동작 중에도 reset) |
| **8254 Reset** | s\_reset8254\_addr | 171H | 8254 Reset |
| **PC Mode** | s\_pc\_RAM\_addr | 160H | wen, ren signal에 따라 PC\_RAM에 쓰기, 읽기 |
| **DA Mode** | s\_da\_start\_addr | 150H | PC\_RAM의 데이터를 DAC로 출력함  (DA mode start) |
| s\_da\_stop\_addr | 151H | 위의 DA mode DAC출력을 멈춤 |
| **AD Mode** | s\_ad\_RAM\_addr | 140H | PC에서 요청한 데이터 개수만큼 ADC하여, AD\_RAM을 통하여 PC\_RAM로 옮기기 |
| **ADR Mode** | s\_adr\_RAM\_addr | 141H | ren signal에 따라 AD\_RAM에 있는 데이터 읽기 |
| **Option Mode (step1)** | s\_opt\_step1\_addr | 130H | PC\_RAM의 데이터에 LPF(Low Pass Filter)를 취해 4배 decimation 한 결과를 Option\_RAM에 저장 |
| **Option Mode (step2)** | s\_opt\_step2\_addr | 131H | Option\_RAM의 데이터를 PC에서 읽기 |
| **Option Mode (step3)** | s\_opt\_step3\_addr | 132H | Option\_RAM의 데이터를 DAC로 출력함 |

표 Address Memory Map

### Signal Controller overview

디지털 회로의 핵심이자 다양한 control 신호들을 제어하는 signal controller는 입력 신호로 s\_wen, s\_ren, s\_oe\_b, 그리고 address decoder의 출력 신호들을 입력 받는다. 주로 이 신호들을 이용하여 state machine이 작동하게 된다. 또한 각 모드의 세부 동작을 위하여 각종 RAM에 이용되는 5개의 address counter의 신호, AD 모드에서 필요한 m\_data, s\_AD\_write\_finish 신호, Option mode Step 1에 이용되는 delayed counter의 address 신호도 입력 받게 되는데, 이들 신호에 대한 설명은 각 모드를 설명하면서 더 자세히 설명하겠다.

아래의 그림은 signal controller의 중심부를 나타낸 것이다. st\_reset 상태에서는 바로 st\_idle 상태로 이동하게 된다. st\_idle에서는 address decoder의 출력 신호를 기다리게 된다. 이후 특정한 신호를 감지하면 각 모드를 동작하기 전의 ready state에 머무르게 되며, 이후에는 각 모드에 대한 동작을 수행하는 형식으로 진행된다.

텍스트, 그림, 스케치, 도표이(가) 표시된 사진

자동 생성된 설명

그림 signal controller의 중심부

## System Clock Setting & 8254 Reset

FPGA에서 이용되는 40MHz 신호인 s\_clk 신호를 8254를 이용하여 분주하여 sys\_clk을 출력한다. 8254의 0,1,2번 포트 중 0번 포트를 이용하게 되며, 183H -> 180H -> 180H 주소를 순서대로 입력함으로써 분주된 sys\_clk을 출력하게 된다. 몇 분주를 하고 싶은지는 m\_data의 LSB를 조정하여 설정할 수 있다. 또한 171H 주소를 입력하면 8254가 reset되어 sys\_clk 분주가 멈추게 된다. 8254 및 sys\_clk 분주는 프로젝트 시작 시 미리 제공된 파일을 이용하였다.

## Software Reset

170H 주소를 입력하면 RAM에 저장된 정보와 8254를 제외한 FPGA의 모든 부분을 software reset하게 된다. 어떠한 모드를 수행 중이거나 수행한 이후에 170H를 입력하면 st\_reset이나 st\_idle로 돌아가 software reset를 할 수 있게 설계하였다.

## PC mode

텍스트, 그림, 도표, 스케치이(가) 표시된 사진

자동 생성된 설명

그림 PC mode state diagram

PC mode를 state diagram으로 나타내었다. s\_PC\_RAM\_addr을 입력으로 받으면 st\_idle에서 st\_pc\_ready로 넘어가게 된다. 여기서 s\_PC\_RAM\_addr = ‘1’이고 s\_oe\_b = ‘1’이면 PC write을 수행하게 되고, s\_PC\_RAM\_addr = ‘1’이고 s\_oe\_b = ‘0’이면 PC read을 수행한다. St\_pc\_ready에서 s\_PC\_RAM\_addr = ‘0’이면 이를 에러로 판단하여 st\_reset으로 가게 하였다.

### PC Write

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 PC write data flow path

텍스트, 도표, 번호, 평행이(가) 표시된 사진

자동 생성된 설명

그림 PC write timing diagram

PC write의 timing diagram에서는 s\_wen, s\_oe\_b, s\_pc\_RAM\_addr이 state를 넘어가는 기준 신호들이 된다. 이들을 연두색 파형으로 표기하였다. 그 아래의 신호들은 data flow path 흐름에 따라 signal controller에서 제어하는 control 신호 및 data들을 나타내었다. 이 중 signal controller에서 제어해야 하는 신호들은 연두색 표시로 따로 체크를 해두었다. S\_pc\_RAM\_addr와 s\_oe\_b를 이용하여 St\_pc\_w1로 이동하여 PC RAM ena를 ‘1’로 만든다. 이후 5 clock cycle 동안 1이 되는 s\_wen을 이용하여 PC RAM wea = “1”로 만들어 PC RAM에 data를 쓰게 된다. st\_pc\_w2에서는 PC RAM address의 count를 1 증가시키고, st\_pc\_w3에서 다음 address를 기다리게 된다.

State diagram의 PC write 부분을 살펴보면 st\_pc\_w1에서 software reset 시 st\_idle로 이동하게 하였는데, 그 이유는 st\_pc\_w1에서 만약 s\_wen = ‘1’을 감지 못할 경우, 무한루프에 빠질 수 있기 때문에 그렇게 조치하였다.

### PC Read

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 PC read data flow path

텍스트, 번호, 폰트, 평행이(가) 표시된 사진

자동 생성된 설명

PC read는 sys\_clk을 1분주한 후 진행된다. PC read는 s\_oe\_b, s\_ren, s\_pc\_RAM\_addr이 state를 넘어가는 기준 신호가 되며, PC write와 마찬가지로 이들을 연두색 파형으로 표기하였다. 또한 data flow path 흐름에 따라 신호와 데이터를 표기하였고, control 신호들은 연두색 체크 표시로 따로 표시를 하였다. Vivado로 Block RAM(BRAM)을 만들 때, read 시에는 2 clock cycle이 지난 후에 해당 데이터가 나오도록 설정하였다. s\_PC\_RAM\_enb가 ‘1’이 된 시점으로부터 2 clock cycle 후에 데이터가 s\_PC\_RAM\_Dout으로 출력된다. 이 데이터는 OUT\_MUX와 OUT\_Latch를 거치게 되고, s\_tri\_en\_b = ‘0’으로 active시켜 m\_data로 PC RAM의 데이터를 PC로 보내게 된다. 이때 PC에서 m\_ren이 ‘0’으로 떨어진 이후에도 3 clock cycle 정도 더 읽을 수 있게 st\_pc\_r13, st\_pc\_r2 등의 state를 더 부여하여 s\_tri\_en\_b를 컨트롤하였다. PC RAM address counter의 ce(count enable)은 st\_pc\_r2에서 ‘1’이 되어 다음 address로 넘어가게 하였다.

PC mode state diagram에서 PC read 쪽을 살펴보면, PC write와 유사하게 동작함을 확인할 수 있다. PC read 시에는 PC write와는 달리 s\_oe\_b = ‘0’이므로 이를 이용하여 st\_pc\_r1으로 이동한다. 이후에는 5 clock cycle동안 ‘1’이 되는 s\_ren을 이용하여 PC read 동작을 수행하게 되고, st\_pc\_r2에서 PC RAM counter를 +1시킨다. St\_pc\_r3에서는 다음 address를 기다리며 머물게 된다. PC write와 마찬가지로 무한루프를 대비하여 st\_pc\_r1에서 software reset를 하면 바로 st\_idle로 넘어가게 하였다.

## DA mode

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 DA start data flow path

텍스트, 번호, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

그림 DA mode timing diagram

DA mode는 sys\_clk을 1분주한 후 진행된다. 150H 입력 시 DA start, 151H 입력 시 DA stop mode가 된다. Signal controller에서는 PC write 시 data 개수에 대한 정보를 저장했던 s\_PC\_RAM\_len 신호와 address decoder 출력 신호인 s\_da\_start\_addr, s\_da\_stop\_addr 신호를 이용해 DA mode를 수행하게 된다. 위의 timing diagram에는 PC RAM의 0~10번지까지 11개의 데이터가 저장되어 있어 s\_PC\_RAM\_len = 10이라고 가정하였다. s\_da\_start\_addr가 ‘1’이 되면 st\_da\_ready 상태로 이동하고, s\_da\_start\_addr이 ‘0’이 되면 st\_da\_ready2, st\_da\_ready3를 거쳐 st\_da\_start 상태로 이동하여 DA start를 수행한다. St\_da\_ready2와 st\_da\_ready3는 dummy state로써, s\_da\_latch\_en 신호를 2 clock cycle 정도 늦게 active 시켜 안정적인 동작을 보장하기 위해 추가하였다. St\_da\_start에서 PC RAM의 0~10번지 데이터를 순차적으로 읽어 m\_dac\_d 포트로 내보내 오실로스코프에서 무한 반복하여 읽을 수 있게 하였다. PC RAM counter를 10번지에서 reset 시켜 0번지로 읽게 해야 하는 과정에서 s\_PC\_RAM\_len을 이용한다. PC RAM counter address와 s\_PC\_RAM\_len(이 경우는 10)과 일치하면 counter의 reset\_b를 ‘0’으로 만들어 0번지로 counter를 reset 시킨다. 이렇게 0~10번지를 읽고 다시 0~10번지를 읽는 식으로 무한 반복하여 읽게 된다.

St\_da\_start에서 151H을 입력하여 s\_da\_stop\_addr가 ‘1’이 되면 st\_da\_stop 상태로 이동하여 PC RAM의 counter를 reset하고 ce를 ‘0’으로 하고 DA latch의 en 신호를 ‘0’으로 하여 마지막으로 출력한 데이터가 m\_dac\_d에서 계속 유지된다. 오실로스코프에서는 데이터 출력이 멈추게 된다.

텍스트, 도표, 그림, 친필이(가) 표시된 사진

자동 생성된 설명

그림 DA mode state diagram

DA start와 DA stop mode를 state diagram으로 나타내었다. St\_da\_ready에서 software reset 시 st\_idle로 돌아가게 하여 오류를 방지하였다. St\_da start 상태에서는 151H 주소가 입력되면 st\_da\_Stop으로 이동하고, 주소가 입력되지 않은 상태이면 st\_da\_start 상태에 머물며, 151H를 제외한 다른 주소가 입력되면 st\_idle로 이동하게 하였다. St\_da\_Stop 상태에서는 150H가 입력되면 st\_da\_ready 상태로 이동하여 DA start를 다시 수행하고, 150H와 151H를 제외한 다른 주소가 입력되지 않으면 st\_da\_stop에서 머물며, 그 둘을 제외한 다른 주소가 입력되면 st\_idle로 이동하게 하였다.

## AD mode

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 AD mode data flow path

AD mode에서는 m\_data에 sampling할 데이터의 개수를 받은 후 function generator로부터 오는 데이터를 ADC와 AD\_latch를 거쳐 AD RAM에 write하게 된다. Timing diagram에서는 sys\_clk을 2분주하였다고 가정하고 작성하였다. AD mode는 AD RAM write, PC RAM write 2가지 동작으로 크게 나뉘는데, AD RAM write 시에는 2개의 state machine을 이용하게 된다. 하나는 다른 모드에서도 이용하는 signal controller이고, 다른 하나는 sys\_clk으로 작동하는 별도의 state machine인 sys\_state\_machine이다. Sys\_state\_machine에서는 AD RAM write에 이용되는 카운터의 reset\_b, ce 신호와 ena, wea 신호 및 추후에 설명할 s\_AD\_write\_finish 신호까지 총 5개의 신호를 컨트롤한다.

### AD RAM write 시작 ~ 끝

텍스트, 번호, 폰트, 평행이(가) 표시된 사진

자동 생성된 설명

그림 AD mode timing diagram (AD RAM write 시작 ~ 끝)

AD RAM write 시, signal controller에서는 m\_data로 sampling할 data 수, 즉 AD RAM에 저장될 data 수를 받는다. 이를 s\_data에 저장하고, s\_s\_AD\_RAM\_max\_address라는 signal controller 내부 신호에 s\_data에서 1을 뺀 수를 저장한다. s\_s\_AD\_RAM\_max\_address는 AD RAM의 최대 번지가 된다. 예를 들어 m\_data로 11을 받으면, AD RAM의 0~10번지에 저장되어야 하므로 s\_s\_AD\_RAM\_max\_address에는 10이 저장되는 식이다. 또한 이 신호를 s\_AD\_RAM\_max\_address라는 signal controller의 출력 신호로 내보내게 된다. 또한 st\_ad\_AD\_RAM\_writing 상태에 머물면서 s\_sys\_state\_machine\_on이라는 출력 신호 역시 같이 내보내게 된다. 앞서 언급한 두 신호는 sys\_clk으로 동작하는 sys\_state\_machine의 입력으로 받게 된다.

Sys\_state\_machine에서는 s\_sys\_state\_machine\_on이 ‘1’일 때부터 동작을 시작하며, 본격적으로 AD RAM에 데이터를 쓰게 된다. Sys\_st\_idle에서 sys\_st\_AD\_ready, sys\_st\_AD\_ready2라는 dummy state를 거치며 sys\_st\_AD\_write 상태에서 본격적으로 write를 시작한다. Sys\_state\_machine은 AD RAM의 A port address counter 신호(s\_AD\_RAM\_addra10)도 입력 받는데, 만약 s\_AD\_RAM\_addra10과 s\_AD\_RAM\_max\_address가 일치하면 (이 경우 10), sys\_st\_ADW\_done0으로 이동한다. 이후 Sys\_st\_ADW\_done1과 sys\_st\_ADW\_done2를 거쳐 sys\_st\_idle로 이동하게 된다. Sys\_st\_ADW\_done0~2까지 총 3 cycle 동안 AD RAM의 write 동작이 종료되었다는 s\_AD\_write\_finish 신호를 ‘1’로 만들게 되며, 이 신호를 signal controller에서 감지하면 그때부터 AD RAM 정보를 PC RAM으로 쓰는 과정을 시작한다.

### PC RAM write 시작 ~ 끝

텍스트, 번호, 평행, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 AD mode timing diagram (PC RAM write 시작 ~ 끝)

Signal controller에서 st\_ad\_AD\_RAM\_writing 상태에 머물다가 s\_AD\_write\_finish가 ‘1’이 되면 st\_ad\_AD\_RAM\_write\_done으로 이동하고, s\_AD\_write\_finish가 ‘0’이 되면 st\_ad\_PC0로 이동하며 이때부터 AD RAM 정보를 PC RAM으로 쓰는 과정이 시작된다. 이후 과정은 온전히 signal controller에 의해서만 진행하게 된다. AD RAM의 B port를 이용하여 데이터를 읽으면 2 clock cycle 후에 유효하므로, AD RAM B port counter보다 2 cycle 후에 PC RAM A port counter를 동작하게 하였다. 이를 위해 st\_ad\_PC0~st\_ad\_PC4까지 dummy state를 두었다. St\_ad\_PC\_RAM\_writing에서 본격적으로 PC RAM에 write를 시작한다. 그러다 AD RAM의 B port 주소가 AD RAM의 최대 번지(s\_s\_AD\_RAM\_max\_address)에 도달하면 st\_ad\_PC5로 넘어가며, 여전히 PC RAM write를 수행한다. 그러다가 PC RAM A port 주소마저 최대 번지에 도달하면 st\_ad\_PC6, st\_ad\_PC7을 거쳐 st\_ad\_wait에 머물게 된다. AD 모드에서 PC RAM write가 모두 끝나면 PC write 모드에서처럼 s\_PC\_RAM\_len에 최대 번지를 저장하게 된다.

텍스트, 친필, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 AD mode state diagram

AD mode의 전 과정을 state diagram을 이용하여 나타내었다. Signal controller에서는 st\_ad\_AD\_RAM\_writing에 머무는 동안 sys\_state\_machine이 작동하며 AD RAM에 write를 수행하고, 이를 마치고 나면 s\_AD\_write\_finish 신호를 signal controller로 보내어 이후에는 PC RAM write 동작을 수행한다. AD mode 전 과정을 마치면 두 state machine은 각각 st\_ad\_wait와 sys\_st\_idle에 머물게 된다.

## ADR mode

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 ADR mode data flow path

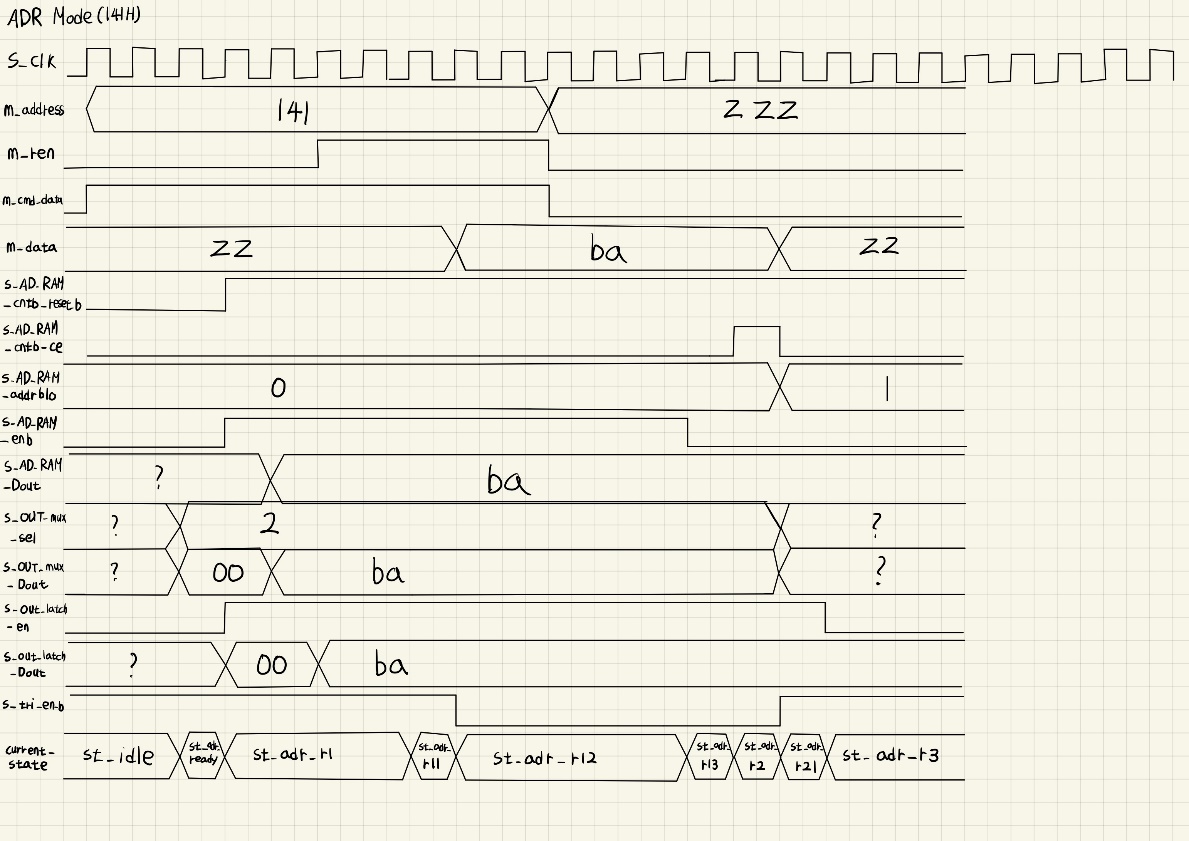


그림 ADR timing diagram

ADR mode에서는 PC mode에서 read할 때와 비슷하게 AD모드를 통해서 저장한 AD RAM의 데이터를 OUT\_mux와 OUT\_latch를 지나 m\_data로 출력하는 동작을 수행한다. ADR mode에서는 address decoder에서 s\_address가 141H이고 s\_cmd\_data가 1로 s\_address가 유효할 때 s\_adr\_RAM\_addr가 1로 set되고 ADR mode가 시작된다. AD RAM은 2 clock cycle이 지난 후에 해당 데이터 나오도록 설정하였기 때문에 s\_AD\_RAM\_enb가 ‘1’이 된 시점으로부터 2 clock cycle 후에 데이터가 s\_AD\_RAM\_Dout으로 출력된다. 그리고 이 데이터는 OUT\_mux와 OUT\_latch를 거치고 s\_tri\_en\_b 를 0으로 active시켜 m\_data로 해당 데이터가 PC로 나갈 수 있도록 해주었다. 그리고 PC에서 m\_ren이 0이된 이후에도 3 cycle동안 더 읽을 수 있도록 st\_adr\_r13, st\_adr\_r2 등의 조건 없이 지나가는 state를 추가하였다. 또한 AD RAM address counter의 ce는 st\_adr\_r2의 한 사이클동안 1이 되도록 하여 AD RAM의 address를 증가시켰다.

텍스트, 친필, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 ADR 모드 state diagram

ADR 모드의 전체적인 state diagram은 위와 같다. 141H가 들어오면 st\_adr\_ready가 되어 기다리다가 s\_oe\_b가 0이 되면 st\_adr\_r1으로 넘어간다. 그 상태에서 s\_ren 신호가 들어와 PC가 read 신호를 주면 st\_adr\_r11, st\_adr\_r12까지 넘어가며 pc에 AD RAM의 데이터를 출력해준다. 그리고 s\_ren 신호가 꺼지고 나서도 3사이클 동안 더 읽을 수 있도록 st\_adr\_r13, st\_adr\_r2, st\_adr\_r21동안 계속해서 m\_data를 출력해준다. 그리고 st\_adr\_r2에서는 한 사이클 동안 AD RAM 카운터를 증가시킨다. 그리고 st\_adr\_r3가 되면 다음 주소를 기다리다 s\_adr\_RAM\_addr이 들어오면 다시 st\_adr\_ready로 가고, 다른 주소가 들어오면 st\_idle로 이동시킨다.

## Option Mode

### Step 1

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 Option1 mode data flow path

텍스트, 번호, 평행, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 Option1 mode timing diagram

Option 1 mode는 PC RAM에 있는 데이터를 B port로 읽고, 그 데이터들을 15 tap filter를 통과시킨 후, filter의 출력 데이터를 4배 decimation하여 Option RAM A port를 이용해 저장한다. s\_opt\_step1\_addr와 s\_wen 신호를 이용하여 st\_op1\_ready -> st\_op1\_ready2 state로 이동한다. St\_op1\_ready에서는 option1 mode에 쓰이는 3가지 카운터인 PC RAM counter, Option RAM A port counter, Delayed counter를 모두 reset시킨다. 또한 filter도 reset시켜 filter 내의 모든 데이터를 0으로 만든다. St\_op1\_ready2에서는 Option RAM ena를 ‘1’로 만들게 된다. 이후 s\_wen이 ‘0’이 되면 trans0~12 state까지 1 cycle마다 이동한다. PC RAM 데이터가 2 cycle 이후에 출력되고, 필터에서 데이터를 처리하는 데에도 몇 cycle이 소모된다. 또한 필터에서 출력된 총 7개의 transient 데이터는 Option RAM에 저장하지 않기 때문에 이들을 모두 고려하여 총 13개의 trans state를 두었다.

Trans0에서 PC RAM enb를 ‘1’로 만들고, PC RAM counter도 count enable시켜 0번지 데이터부터 순차적으로 출력되게 한다. 2 cycle 이후인 trans2부터 PC RAM의 0번지 데이터가 출력되는데, 이때부터 필터의 en를 ‘1’로 만들어 필터 내로 데이터가 입력된다. 필터에서 처리된 데이터는 trans9 state부터 출력된다. S\_filter\_Dout에서 -1, -1, -1, -1, 0, 3, 9까지의 7개의 데이터는 transient data이므로 이들은 Option RAM에 저장하지 않고, 이후 데이터인 13부터 Option RAM에 저장한다. Trans0~12까지 모두 지나면 st\_op\_write0~3까지의 state를 Option RAM에 데이터를 모두 쓸 때까지 반복한다.

St\_op\_write0부터는 Delayed counter가 0부터 카운트한다. St\_op\_write1 -> st\_op\_write2 -> st\_op\_write3 -> st\_op\_write0 ->…을 반복할 때 하나의 state를 움직일 때마다 +1씩 카운트하게 된다. 이 카운터는 PC RAM의 데이터의 최대 번지인 799까지 카운트하게 된다. Delayed counter는 7개의 transient data를 모두 보내고, 그 이후부터 800개의 데이터를 세기 위해 만들었다. Option RAM A port counter는 st\_op\_write3에서만 count enable되고, wea도 “1”이 된다. 이러한 방식으로 4배 decimation을 할 수 있다. Timing diagram에서 보면 Option RAM 0번지에 13, 1번지에 19, 199번지에 3을 저장하고, st\_op1\_wait으로 이동한다. 그리고 그 때의 Option RAM A counter 값(199)을 s\_Option\_RAM\_len에 저장한다. Option3 mode를 위해 Option RAM 최대 번지 데이터가 필요하다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 15 tap filter

이번 설계에서 제작한 15 tap filter이다. 이 필터는 LPF의 기능을 수행한다. 총 15개의 D flipflop과 15개의 multiplier IP를 이용하였고, add 연산은 sign extension을 이용하였다. 출력된 21 비트 신호 중 20번째와 15~9번째 신호만 이용하여 최종 출력은 8비트가 된다. 또한 D flipflop에서 1 cycle, multiply 연산 시 1 cycle, add 연산에 총 4 cycle이 소요되므로, 최초 입력된 데이터는 6 cycle 이후에 출력된다. 또한 필터 입력은 PC RAM의 데이터 출력 부분과 연결되는데, Vivado Block RAM(BRAM) 특성 상 PC RAM enb가 ‘0’이어도 그 이전에 출력한 데이터가 계속 출력된다. 그러면 불필요한 데이터가 D flipflop을 통해 필터 내로 입력될 수 있는데, 불필요한 신호가 필터에 입력되는 것을 방지하기 위해 D flipflop에 enable 신호(s\_filter\_en)를 달아서 꼭 필요한 데이터만 필터에 입력되게 하였다.

### Step 2

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 Option2 mode data flow path

텍스트, 번호, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 Option2 timing diagram

Option mode step 2는 Option RAM의 데이터를 PC read와 동일하게 m\_data로 출력하는 동작을 수행한다. PC read와 다른 점은 state의 이름이나 사용하는 signal만 PC RAM에서 Option RAM으로 바뀌고 OUT\_mux에서 출력하는 값만 다르고 나머지는 동일하게 동작하는 모드이다.

### Step 3

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

그림 Option3 mode data flow path

텍스트, 번호, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

그림 Option3 timing diagram

Option mode step 2는 Option RAM의 데이터를 DAC로 반복적으로 출력하는 모드로 DA 모드와 RAM만 다르고 나머지는 동일하게 동작한다. DA 모드와 다른 점은 state의 이름이나 사용하는 signal이 다르고 s\_DA\_mux\_sel이 1이 아니라 0이 되도록 해서 Option RAM에서 나온 출력이 DA latch를 거쳐 ADC로 나가게 된다.

텍스트, 그림, 도표, 친필이(가) 표시된 사진

자동 생성된 설명

그림 Option mode state diagram

Option 1,2,3 mode의 state diagram이다. Option1은 ready, ready2 state에서 trans0~12를 거치고st\_op\_write0~3까지 Option RAM에 데이터를 저장하고, 모두 저장하면 st\_op1\_wait에 머물어 다음 주소를 기다리게 된다. Option2, 3은 PC read와 DA mode와 유사하게 동작한다. 특정한 상황에서 무한루프가 생기는 것을 방지하기 위해 특정 state에서는 software reset을 하면 st\_idle로 돌아가게 하였다.

# 평가 및 분석 (Simulation 및 시연 분석)

## PC latch & Address Decoder

스크린샷, 다채로움, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

그림 PC latch, Address decoder 시뮬레이션

PC write를 할 때의 시뮬레이션 결과이다. PC latch를 이용하여 ‘m\_’ 신호들을 s\_clk의 rising edge에 동기화되는 ‘s\_’신호들로 출력한다. 이렇게 하면 PC에서 임의의 시간에 신호가 들어와도, PC latch를 거침으로써 FPGA 내에서는 setup time을 충분히 확보할 수 있게 된다. 또한 160H 주소가 들어왔으므로 address decoder 출력 신호에는 s\_PC\_RAM\_addr가 ‘1’이 됨을 확인할 수 있다. Signal controller에서는 이 신호를 감지하여 PC mode 동작을 수행하게 된다.

## System Clock Setting & 8254 Reset

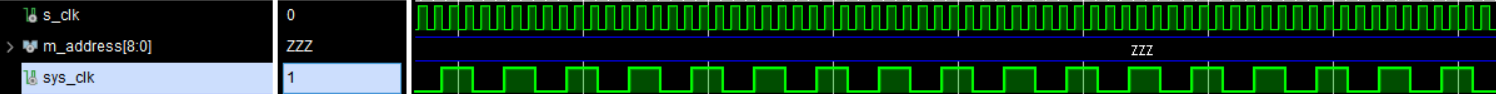


그림 8254 4분주 시뮬레이션

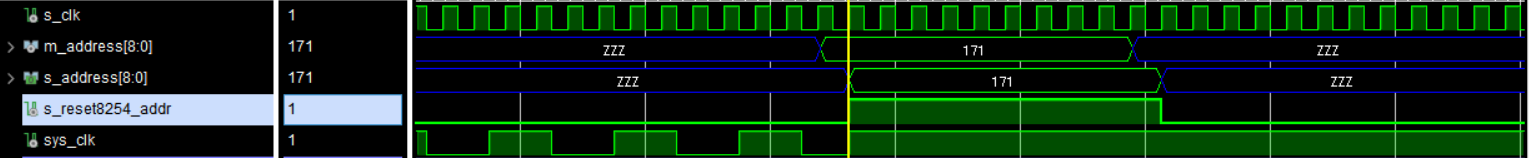


그림 8254 reset 시뮬레이션

Figure 3은 8254에 183H -> 180H -> 180H 순으로 주소를 입력하고, 4분주할 수 있게 m\_data를 입력하여 나타낸 시뮬레이션 결과이다. s\_clk에서 4분주되어 sys\_clk이 잘 출력되는 모습을 볼 수 있다. 또한 Figure 4에서는 171H 주소를 입력해 address decoder에서는 s\_reset8254\_addr이 ‘1’이 되어 이 신호가 8254 device을 reset시켜 sys\_clk의 분주가 멈추는 것을 확인할 수 있다.

## PC mode

### PC Write

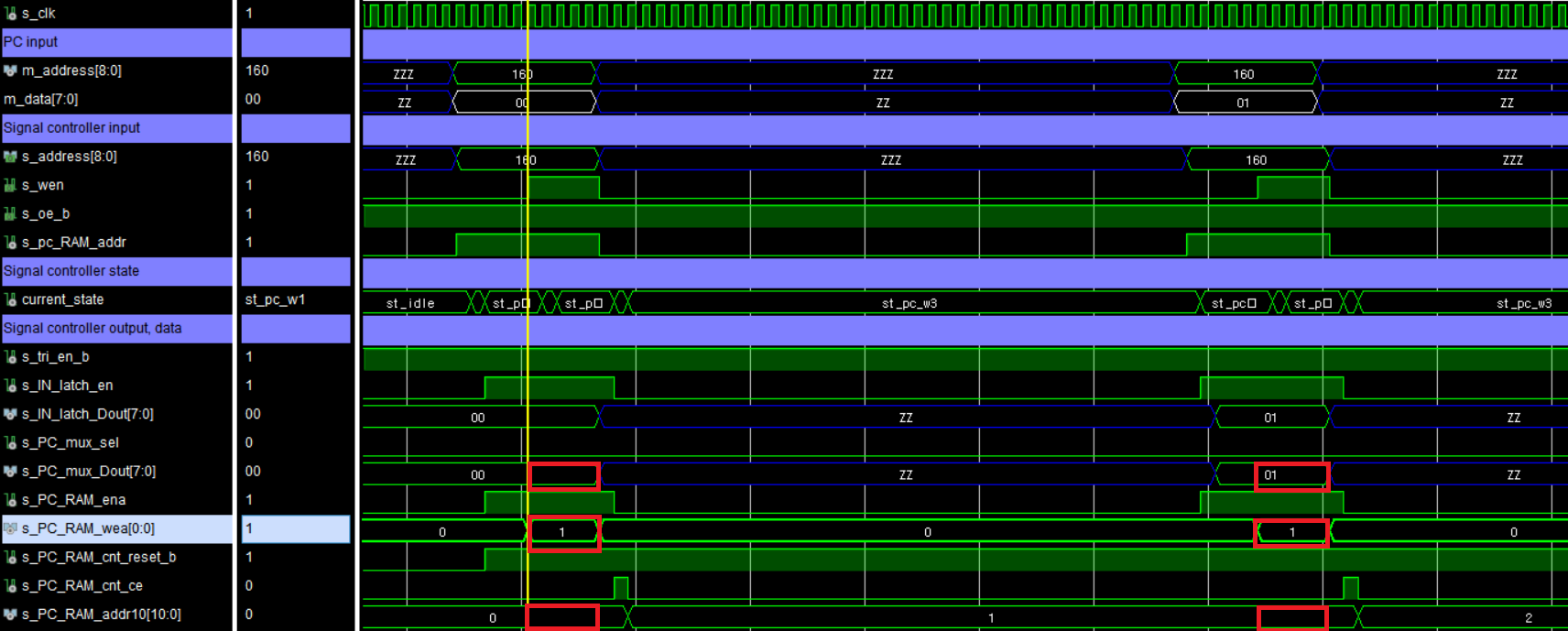


그림 PC write 시뮬레이션

PC write 시의 PC input 신호, signal controller input 신호(s\_address는 입력 받지 않음), signal controller state, 그리고 control되는 신호 및 data들을 나타내었다. 160H를 입력하면, s\_PC\_RAM\_addr와 s\_oe\_b 신호를 이용하여 st\_pc\_ready -> st\_pc\_w1 state로 이동한다. 이후에 5 clock 동안 들어오는 s\_wen 신호의 변화를 감지하면서 st\_pc\_w11 -> st\_pc\_w12 -> st\_pc\_w2 -> st\_pc\_w3 state로 이동한다. PC RAM counter의 출력 신호인 s\_PC\_RAM\_addr10이 0번지일 때, m\_data 00을 쓰고, 1번지일 때 m\_data 01을 쓰는 것을 ena와 wea 신호를 이용하여 확인할 수 있다. 그리고 st\_pc\_w2 상태에서 PC RAM counter의 count enable (ce)를 1 clock 동안 active 시켜 다음 번지 수로 넘어가는 것 역시 확인할 수 있다. 이런 식으로 다음 번지로 계속 넘어가며 PC에서 넘어오는 data들을 차례로 저장하게 된다.

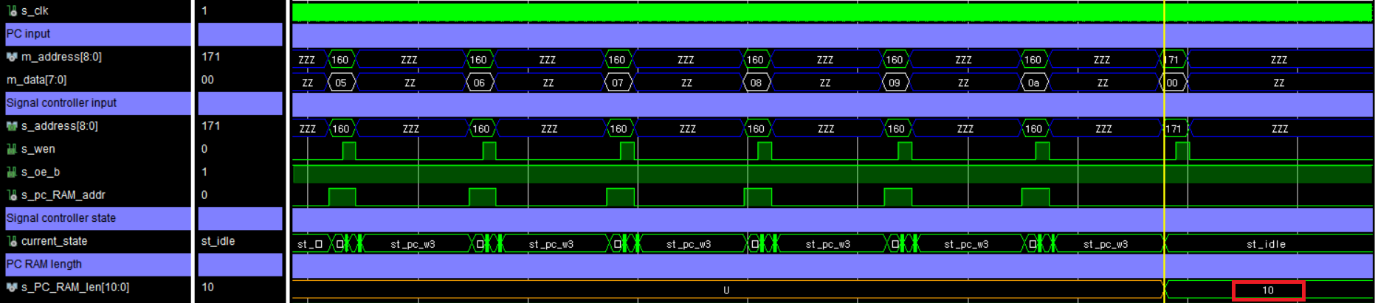


그림 PC write 후 data 수 저장

PC write를 모두 마치고 다른 address를 받아들여 st\_idle 상태로 돌아갈 때, s\_PC\_RAM\_len에 PC RAM에 쓰여진 최대 번지 수를 저장하게 된다. 만약 11개의 데이터를 저장하면, 0~10번지까지 저장되어 있으므로 s\_PC\_RAM\_len에는 10이 저장이 된다. 이 값은 이후 DA mode에서 이용하게 된다.

### PC Read

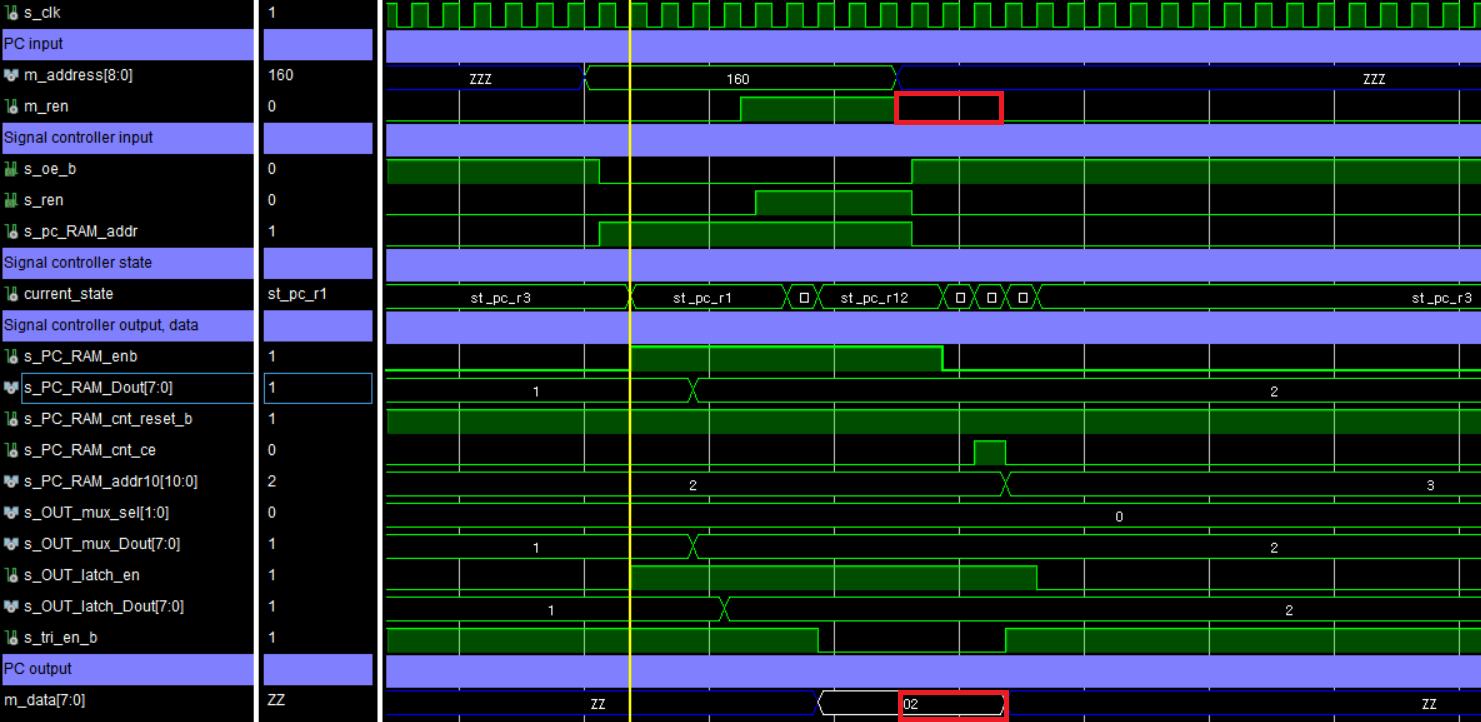


그림 PC read 시뮬레이션

PC write와 마찬가지로, st\_idle 상태에서 s\_PC\_RAM\_addr와 s\_oe\_b 신호를 이용하여 st\_pc\_ready, st\_pc\_r1 상태로 이동한다. 위의 결과는 2번지의 데이터를 읽을 때의 시뮬레이션 결과이므로 st\_pc\_r3에서 st\_pc\_r1으로 이동한다. PC RAM의 enb = ‘1’이 된 시점부터 2 clock cycle 후에 PC RAM Dout이 바뀌어 2번지의 데이터를 읽는 것을 확인할 수 있다. 또한 s\_ren이 ‘0’이 된 이후에도 dummy state를 몇 개 두어 m\_ren이 ‘0’이 되고 나서도 3 clock cycle 정도는 넉넉하게 PC에서 데이터를 읽을 수 있게 하였다. m\_data의 02 값이 m\_ren이 ‘0’이 되고 나서도 3 clock cycle은 유지되는 것으로 이를 확인할 수 있다.

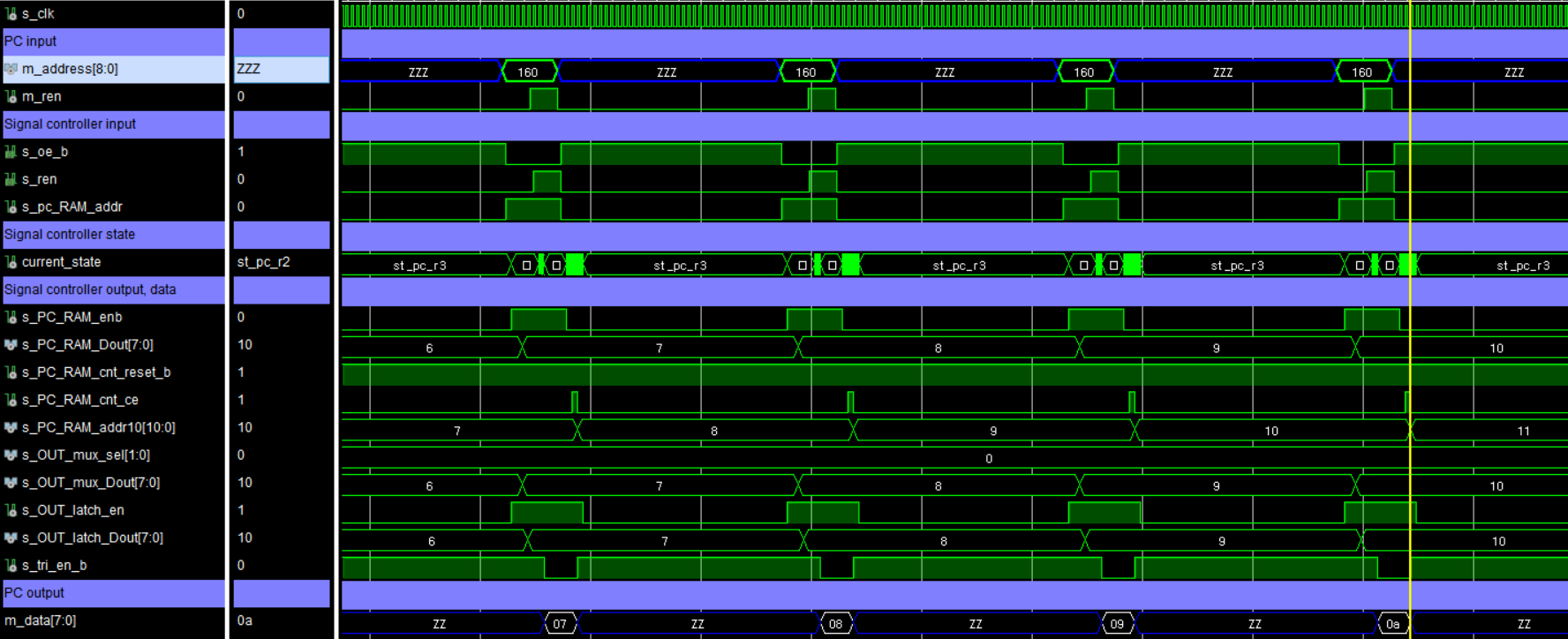


그림 PC read 시뮬레이션2

PC에서 address를 n번 반복하여 FPGA에 보내면 PC RAM의 데이터 n개를 PC read할 수 있다. 위의 시뮬레이션은 00~0a까지 총 11개의 데이터를 PC RAM의 0번지~10번지에서 불러와 읽는 결과이다. m\_data가 0a까지 온전하게 출력되는 것을 시뮬레이션으로 확인할 수 있다.

## DA mode

### DA Start

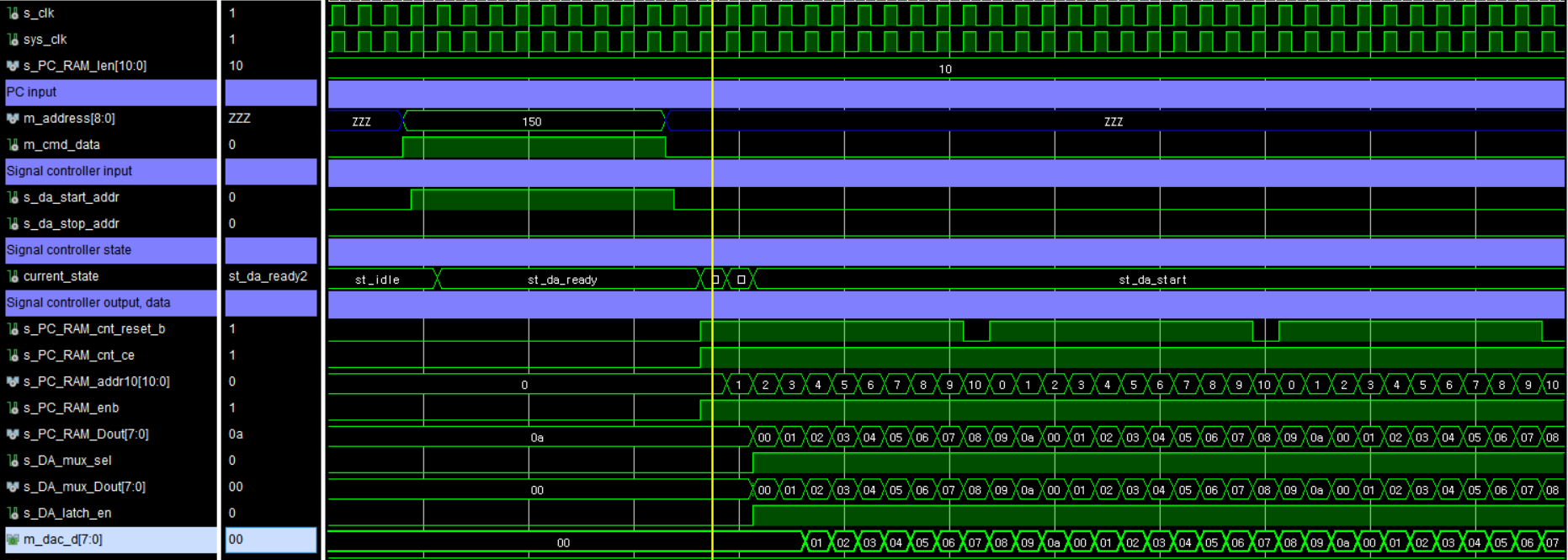


그림 DA start 시뮬레이션

DA mode는 일단 sys\_clk을 1분주한 후에 진행된다. s\_da\_start\_addr가 ‘1’이 되면 st\_da\_ready로 넘어가고, 이후 st\_da\_start에서 m\_dac\_d로 데이터를 무한 반복하여 내보내게 된다. 이러한 과정이 시뮬레이션에서는 모두 잘 표현되어 있으며, s\_PC\_RAM\_addr10 (PC RAM의 주소)와 s\_PC\_RAM\_len가 일치할 경우에 counter를 reset시킨다. 이 시뮬레이션에서는 PC RAM 주소가 10일 때 reset시켜 0~10번지 데이터를 무한 반복하여 출력하게 된다.

### DA Stop

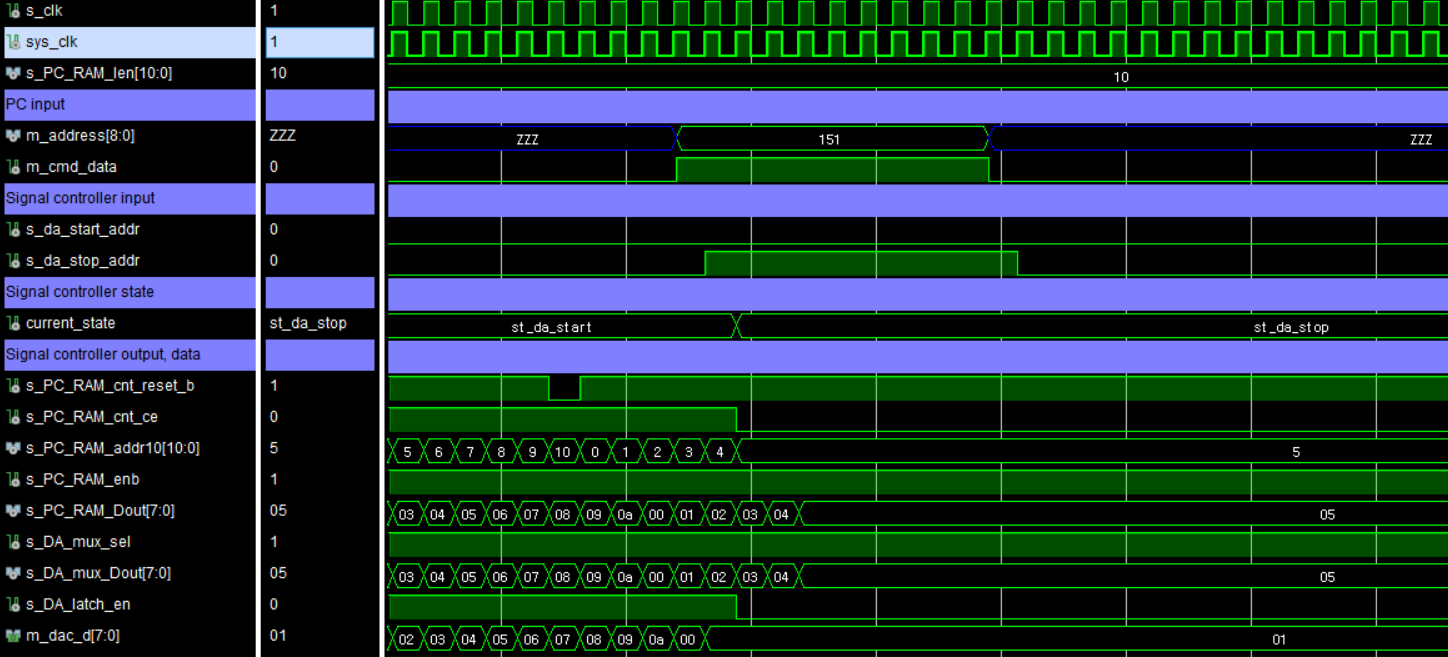


그림 DA stop 시뮬레이션

DA start를 진행하다가 151H 주소를 입력 받으면 s\_da\_stop\_addr이 ‘1’이 되고 이를 감지하여 무한 반복하는 출력이 멈추게 된다. m\_dac\_d는 이전에 출력한 데이터인 01로 계속 유지된다.

## AD mode

### AD RAM write 시작 ~ 끝



그림 AD mode AD RAM write 시뮬레이션

Sys\_clk을 2분주 한 후, PC에서 140H 주소와 sampling할 데이터 수를 11개로 입력하였다. 일단 signal controller 내부 신호 s\_s\_AD\_RAM\_max\_address와 출력 신호 s\_AD\_RAM\_max\_address에 10이 저장됨을 확인할 수 있다. Signal controller는 s\_ad\_RAM\_addr가 ‘1’이 되면 st\_ad\_AD\_RAM\_writing 상태에 머물고, s\_sys\_state\_machine\_on 출력 신호를 내보낸다. 이를 sys\_state\_machine이 감지하면 sys\_state\_machine이 움직이며 AD RAM write 동작을 수행한다. 0번지부터 10번지까지 AD RAM wea가 “1”이 되면서 write가 잘 수행됨을 확인할 수 있다. AD RAM 번지 수와 s\_AD\_RAM\_max\_address가 일치하면 sys\_state\_machine은 sys\_clk의 3 clock 동안 s\_AD\_write\_finish 신호를 ‘1’로 만들고, 이를 signal controller에서 감지하여 st\_ad\_AD\_RAM\_write\_done 상태로 이동하면서 PC RAM에 데이터를 쓸 준비를 하게 된다.

### PC RAM write 시작 ~ 끝



그림 AD mode PC RAM write 시뮬레이션

AD RAM에 있는 정보를 읽은 후, PC RAM에 저장하는 과정이다. AD RAM enb가 ‘1’이 되고 2 clock cycle 후에 해당 번지의 데이터가 출력되므로 dummy state 몇 개를 거치게 되고, st\_ad\_PC\_RAM\_writing 상태에서 본격적으로 PC RAM에 데이터를 쓰게 된다. PC RAM의 0~10번지까지 AD RAM에서와 동일한 데이터가 쓰이는 것을 확인할 수 있다. 모든 과정이 마치면 st\_ad\_wait 상태에 머물며 다음 주소를 기다리게 된다. 시뮬레이션에서는 이전의 s\_PC\_RAM\_len가 10이어서 변화가 없지만 실제로는 st\_ad\_wait으로 넘어가면서 s\_PC\_RAM\_len을 새로운 값으로 갱신한다.

## ADR mode

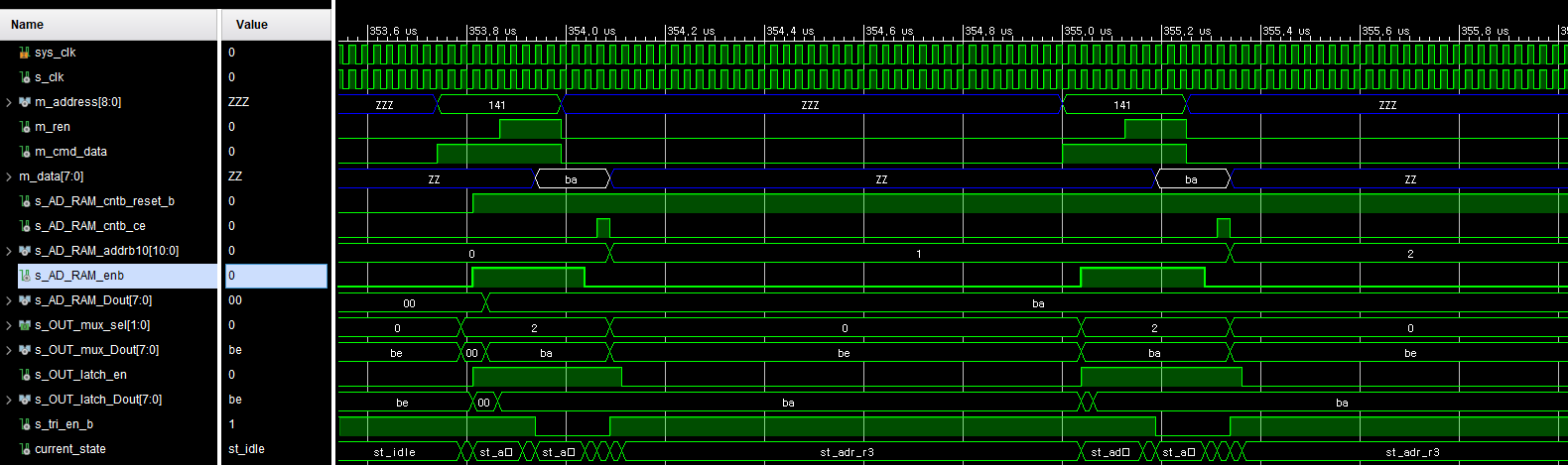


그림 ADR mode 시뮬레이션

ADR 모드를 통해서 AD RAM에 있는 데이터를 m\_data로 출력하는 과정이다. 먼저 m\_address로 1이 들어오면 state가 st\_adr\_ready, st\_adr\_r1이 되고 m\_ren신호로 데이터를 읽겠다고 알려주면 st\_adr\_r11에서 AD RAM에 enb신호가 들어가고 2 cycle뒤에 AD RAM Dout으로 나오는 것을 확인할 수 있다. 그리고 m\_ren이 0이 되고 나서도 3 cycle동안 dummy state를 통해서 m\_data를 출력해주었다. 마지막으로 st\_adr\_r2에서 AD RAM의 카운터를 증가시키고 ad\_str\_r3가 되어 다음 m\_address를 기다린다.

## Option Mode

### Step 1

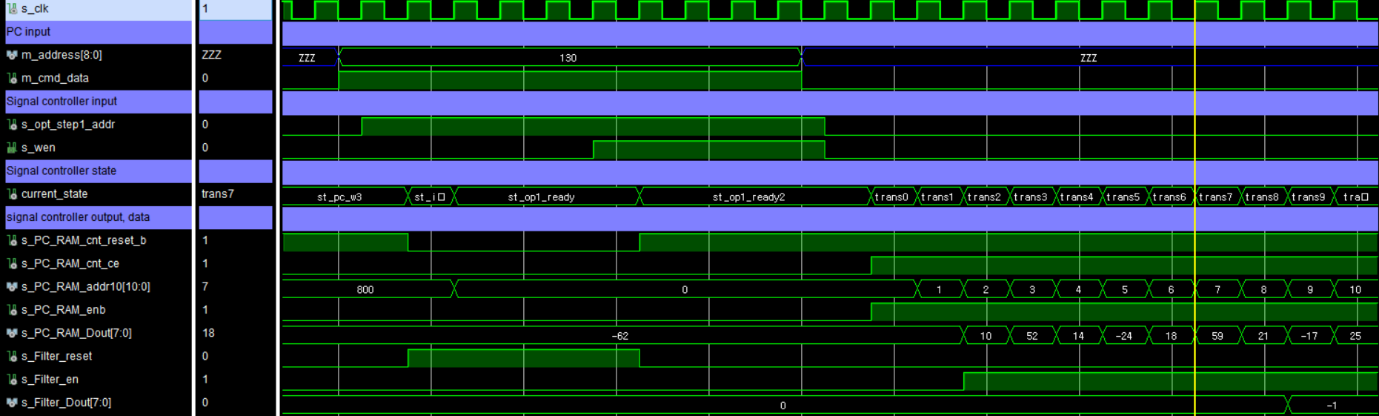


그림 Option1 시뮬레이션 – Filtering

Option1의 시작은 PC RAM enb를 ‘1’로 만들어 PC RAM 데이터를 순차적으로 출력시키고, 이 데이터를 필터를 거치게 하는 것이다. St\_op1\_ready에서는 filter를 reset시켜 내부의 데이터를 모두 0으로 만든다. Trans0 state에서 PC RAM enb를 ‘1’로 만들고, PC RAM address도 카운팅을 시작한다. 그러면 2 cycle 후인 trans2 state부터 0번지 데이터인 10이 출력이 되는데, 이때부터 필터를 enable시킨다. 그러면 trans9 state부터 필터링된 -1 데이터가 출력되기 시작한다.



그림 Option1 시뮬레이션 – Option RAM write

S\_filter\_Dout에서 -1~9까지의 데이터는 transient data이기 때문에 이들은 저장하지 않는다. Trans12이후로는 st\_op\_write0~3까지 계속 돌면서 1 cycle마다 delayed counter가 +1씩 카운트하는 것을 확인할 수 있다. st\_op\_write3에서만 Option RAM 주소가 +1되고, 데이터가 써지는 것을 확인할 수 있다. Option RAM 0번지에 13, 1번지에 17, 2번지에 24, 3번지에 32가 저장된다.

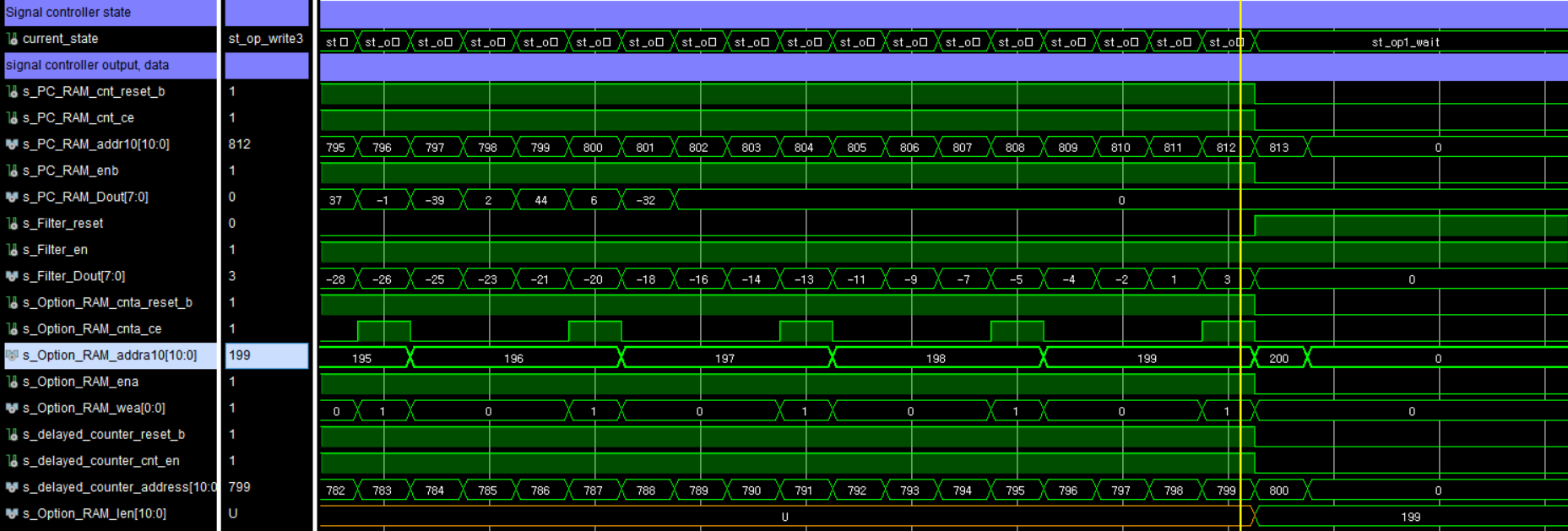


그림 Option1 시뮬레이션 – Option RAM write 종료

Delayed counter가 PC RAM 최대 번지 수인 799까지 도달하면 st\_op1\_wait으로 이동한다. 이 때 Option RAM은 199까지 써진 것을 확인할 수 있고, s\_Option\_RAM\_len에 최대 번지인 199을 저장한다. St\_op1\_wait에서 다음 주소를 기다리게 된다.

### Step 2

스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

그림 Option2 mode 시뮬레이션

Option mode step2를 통해서 Option RAM 저장되어 있는 데이터를 m\_data로 출력하는 과정이다. m\_address로 131이 들어와 s\_opt\_step2\_addr가 1로 set되고 st\_op2\_ready Option RAM의 카운터를 reset하고 m\_ren 신호가 들어오면 m\_data로 Option RAM의 데이터를 출력하기 시작한다. 그리고 몇 개의 dummy state를 지나서 st\_op2\_r2에서 Option RAM의 카운터를 증가시키고 m\_data의 출력도 멈춘다. 그리고 st\_op2\_r3에서 다음 m\_address를 기다리다가 다시 s\_opt\_step2\_addr가 1로 set되고 카운터가 하나 증가된 상태에서 동일하게 Option RAM의 데이터를 출력하는 동작을 수행한다.

### Step 3

스크린샷, 라인, 다채로움이(가) 표시된 사진

자동 생성된 설명

그림 Option3 mode 시뮬레이션

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 Option3 mode stop 시뮬레이션

Option step 3 mode에서는 Option RAM의 데이터를 m\_adc\_d로 반복해서 출력시킨다. m\_address에서 132가 들어와 s\_opt\_step\_addr이 1이되면 st\_op3\_ready상태에 있다가 st\_op3\_ready2와 st\_op3\_ready를 거쳐 데이터가 출력된다. 이때 Option RAM에 데이터가 2 cycle뒤에 나오기 때문에 st\_op3\_ready2와 st\_op3\_ready를 추가하여 대기하도록 하였다. 그리고 st\_op3\_start에서는 반복적으로 카운터를 증가시키며 Option RAM의 데이터를 ADC로 출력해준다. 그리고 m\_address로 151이 들어오면 s\_da\_stop\_addr가 1로 set되면서 st\_op3\_start에서 st\_da\_stop모드로 들어가 더 이상 카운터를 증가시키지 않고 출력도 하지 않는다.

## 시연 분석

1차 시연에서는 모든 기능이 작동하지 않았다. 그 이유를 schematic 분석을 통해 확인할 수 있었다. Schematic을 봤을 때, 특정 입력 포트인 m\_adc\_d, m\_oe\_b, m\_ren 포트가 회로에 연결되지 않았고, component들이 10개 이상 삭제되어 있음을 확인하였다. Behavioral 시뮬레이션으로는 문제없이 작동됐지만, 툴에서 회로를 합성할 때 필요 없는 component들을 삭제하고, 포트 연결을 제대로 하지 않은 것이었다. 그 원인을 알아내기 위해 많은 시간을 소비하였는데, 바로 signal controller의 process문과 연관이 있었다.

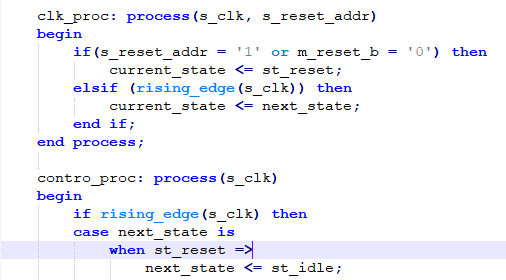


그림 1차 시연 때 사용한 process 문 코드

1차 시연에서는 process 문 2개를 이용하였다. 1개는 current\_state가 무조건 next\_state로 이동하라는 process 문이었고, 나머지 1개는 각 모드에 따라 next\_state를 지정해주는 process 문이었다. 하지만 이러한 코드는 툴이 회로를 합성할 때 이상 현상을 일으켜 우리가 원하는 형태의 회로를 합성하지 못하였다. 이를 기존 방식대로 1개의 process 문을 이용하는 코드로 변경하였을 때는 schematic에서 모든 입력 포트가 연결되고, top 파일에 선언한 component들이 모두 살아있음을 확인할 수 있었다.

|  |  |  |
| --- | --- | --- |
|  | Schematic 회로 | 입력 포트 |
| Process 문 2개 이용 |  |  |
| Process 문 1개 이용 |  |  |

2차 시연에서는 PC read, AD, ADR, Option 2 모드가 되지 않았다. AD 모드는 원래 s\_clk으로 돌아가는 process 문에서 sys\_clk의 rising edge를 감지하는 방식으로 코드를 구성했는데, 시연에서 sys\_clk을 1분주하여 데이터를 입력 받아 rising edge를 감지하지 못하였기 때문이었다. ADR 모드는 AD 모드가 되지 않아 확인할 수 없었다. PC read와 Option 2는 둘 다 RAM의 데이터를 PC로 읽는 과정에서 각 address counter 값이 중간에서 0으로 reset 되는 문제가 발생했었다.

텍스트, 스크린샷, 친필, 평행이(가) 표시된 사진

자동 생성된 설명

그림 2차 시연에서 PC로 read 시 문제점

AD 모드는 sys\_clk으로 돌아가는 state machine을 따로 설계하여 이를 해결하였다. PC read 및 option 2 모드는 wait 하는 상태에서 각 모드의 1단계로 돌아가지 않고 st\_idle로 돌아가 카운터를 reset 시키는 것이 문제였으므로 이 부분의 코드를 수정하여 해결하였다.

# 토의

## 제시된 주요 설계 요소와 제한 요소들을 만족하는 설계를 하였는지 논의

제시된 PCFG의 요구사항인 8254를 통한 clk 분주, Software reset, 8254 reset, PC write mode, PC read mode, DA start, DA stop, AD mode, ADR mode, Option1 – 15 tap LPF & decimation, Option2 – read, Option3 – DA까지 모든 설계 요구사항을 만족시키도록 회로를 설계하였다. 또한 이러한 여러 mode가 원활히 동작하기 위해서 latch와 tri state buffer, mux 등을 사용하여 데이터 bus에서 데이터가 이동할 때 충돌하지 않고 원하는 곳으로 데이터가 이동할 수 있도록 bus를 관리해주었다.

또한 실제 보드에서도 회로가 동작하도록 기본적인 state에 일부 여분의 state를 추가하여 timing에서 문제가 발생하지 않도록 설계하였다. 이를 통해 주어진 clock에 따라 각 모드에서 사용하는 RAM에 데이터가 잘 저장되고 출력되도록 설계하였다.

## Top-down 방식으로 설계하였는지 논의

주어진 top 파일을 기본 베이스로 하여 필요한 모듈을 먼저 선언하고 top 파일에 해당 component와 component간의 연결을 위한 signal을 추가하여 기본적인 data path를 설계하였다. 그리고 각 모듈의 동작을 하나씩 설계하고 정상적으로 동작하는지 확인하고 top 파일 상에서도 잘 연결되어 동작하는지 확인하며 top-down 방식으로 회로를 설계하였다.

## 임의의 입력에 대해서도 안정적으로 동작하는 논의

우리가 만든 보드에 주어줄 수 있는 임의의 입력은 크게 m\_address m\_oe\_b, m\_ren, m\_wen, m\_cmd\_data, m\_data, ADC\_din, m\_clk가 존재한다. 우리가 설계한 회로에서는 기본적으로 각 모드를 동작하기 위해 필요한 데이터가 모두 들어와야 다음 state로 넘어가서 동작을 수행하기 때문에 임의의 입력이 들어오더라도 안정적으로 동작한다.

예를 들어 m\_address로 잘못된 주소가 들어오면 address\_decoder에서 모든 신호에서 0이 나오고 따라서 어떠한 동작도 수행하지 않는다. 또한 ADC에서 잘못된 데이터가 들어오더라도 AD 모드가 아니면 ADC의 데이터가 회로 내부로 들어오지 못하거나 m\_ren, m\_wen신호 없이는 read나 write동작을 수행하지 않기 때문에 임의의 입력에 대해서도 안정적으로 동작한다. 또한 필요하다면 software reset신호를 통해서 회로를 reset해줄 수 있기 때문에 우리가 설계한 회로는 안정적인 회로이다.

## logic량은 최적화되었는지 논의

기본적으로 주어진 전체적인 block diagram에서 선언된 신호나 모듈만을 추가하여 설계를 시작하였다. 그리고 필요에 따라 최소한으로 signal이나 모듈의 input, output을 추가하며 회로를 설계하였다. 예를 들어 signal controller에서 DA모드나 Option 3를 위해서 PC RAM에 저장된 데이터의 개수를 저장하는 s\_PC\_RAM\_len과 같은 신호를 추가하거나 여러 모드에서 state 관리를 위한 signal도 current\_state만 선언하여 모든 모드를 관리하도록 하였다. 이처럼 signal을 최소한으로 선언하고 각 모듈을 동작시키기 위한 신호도 en, reset과 기본적인 동작을 위한 input들만 포함하도록 설계하여 logic량을 최적화하였다.

## 설계 결과를 실제 제품개발에 사용될 수 있는지를 검토

이번에 설계한 회로를 그대로 실제 제품개발에 사용하기는 어렵겠지만 실제 제품 설계 시 제공되는 신호를 바탕으로 이번 회로의 기본 동작 logic은 그대로 사용하면 실제 제품에서도 사용할 수 있다고 생각된다. 필요에 따라 동작을 수행하기 위한 address가 바뀌더라도 address decoder만 바꾸고 회로에서 모든 모듈에 control 신호를 제공하는 signal controller의 회로는 그대로 사용하면 우리가 설계한 회로와 동일한 동작을 하도록 만들어 줄 수 있다. 또한 추가적인 모드가 더 필요하다면 해당 모드를 위한 모듈만 따로 추가하고 signal controller에 data bus를 관리하기 위한 코드만 일부 추가하면 설계한 회로를 확장하여 실제 제품개발에도 사용할 수 있을 것이다.

## 설명서 구조의 단점 및 보완사항 논의

설명서의 구조에서는 clk을 s\_clk와 sys\_clk를 사용한다. 따라서 Option RAM과 AD RAM의 경우는 port A와 port B에서 사용하는 clk이 다르다. 따라서 램의 두 포트를 두개의 process를 통해서 관리해야 하는 단점이 있다. 또한 각 모드를 수행할 때 마다 RAM을 초기화 하는데 RAM에 주소를 이어서 저장할지 처음부터 저장할 지 선택할 수 없다는 단점이 있다. 이를 보완하기 위해 각 모드별로 address나 signal을 하나 더 추가하여 선택할 수 있도록 하면 좋을 것 같다. 그 외에도 8254 세팅을 위해 하나의 address만으로 가능하도록 내부 회로를 설계하는 등 몇 가지 더 보완하면 조금 더 안정적이고 추가기능이 많은 회로를 설계할 수 있을 것이다.

# 참조 문헌

* 2023 PCFG 보강 자료
* 2023 1학기 고급디지털회로설계 프로젝트 PCFG 상세설계설명서