ZynqLab04

학과	전기정보공학과
학번	20101418
이름	장민진
과목명	SoC 설계 입문
제출일자	2024/11/10

목차

I. 개요 및 구조

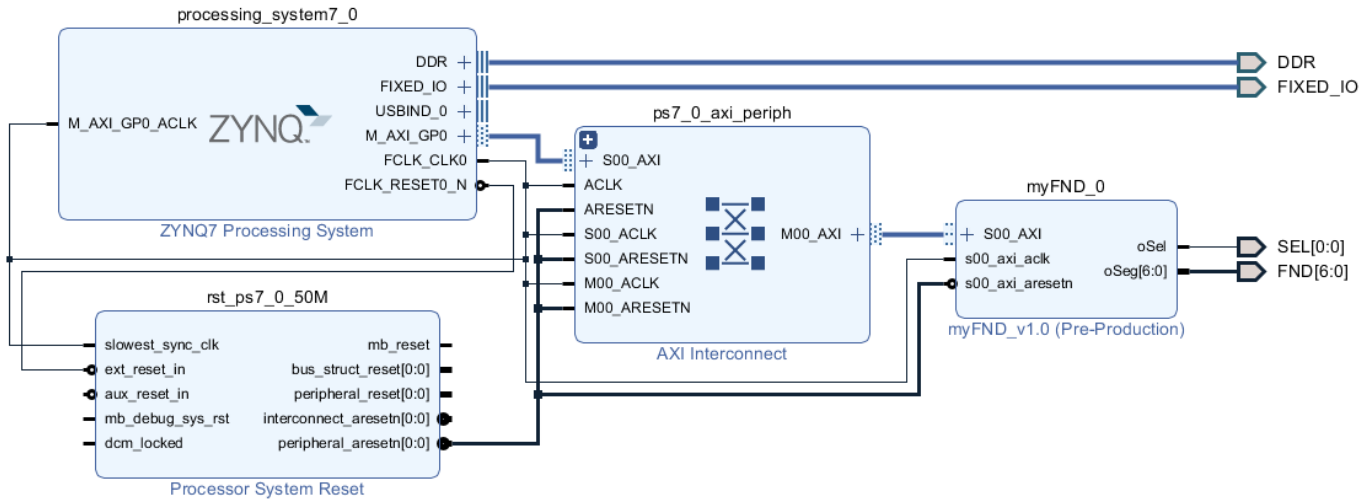
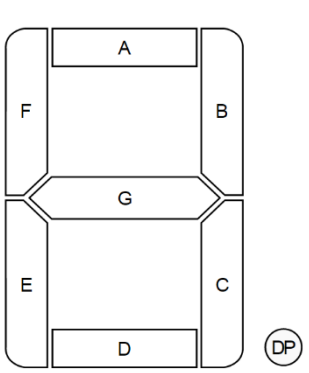
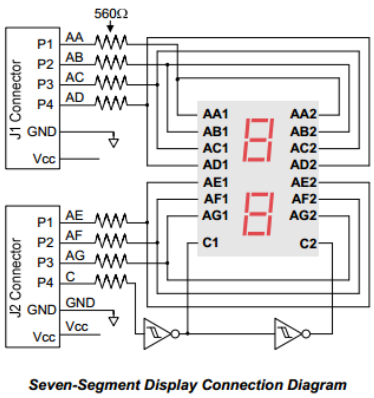


Figure 1 . Block Design Diagram

이번 과제는 최종적으로 FND 의 기능을 HW 코드로 구현하여 Board Test 까지 완료하는 것이다. 위는 AXI 와 Custom IP 인 myFND, 그리고 출력 포트인 FND[6:0], SEL 신호들 사이의 구성을 보여준다.





Pinout Description Table

Header J1			Header J2		
Pin	Signal	Description	Pin	Signal	Description
1	AA	Segment A	1	AE	Segment E
2	AB	Segment B	2	AF	Segment F
3	AC	Segment C	3	AG	Segment G
4	AD	Segment D	4	C	Digit Selection pin
5	GND	Power Supply Ground	5	GND	Power Supply Ground
6	VCC	Positive Power Supply	6	VCC	Positive Power Supply

Figure 2 . Seven Segment

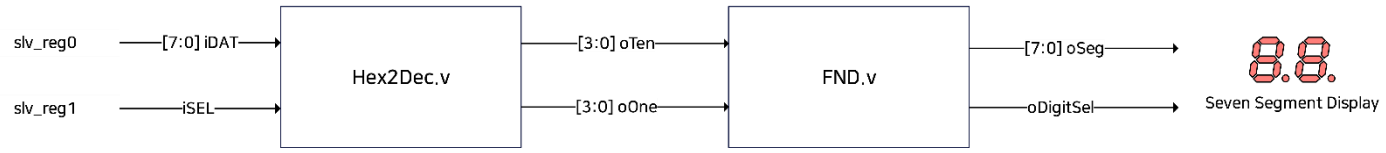


Figure 3 . FND Architecture

다음은 HW 동작을 설계한 Hex2Dec, FND 모듈의 구조이다.

먼저 Processor에서 slv_reg0에 8비트 값을 Write 한다. 이는 8비트 데이터 iDAT로, Hex2Dec 모듈에 입력된다. 또한 slv_reg1의 하위 1bit는 Hex형식으로 데이터를 표현할지, Dec 형식으로 데이터를 표현할지 결정하는 Select 신호로 마찬가지로 Processor에서 Write한다.

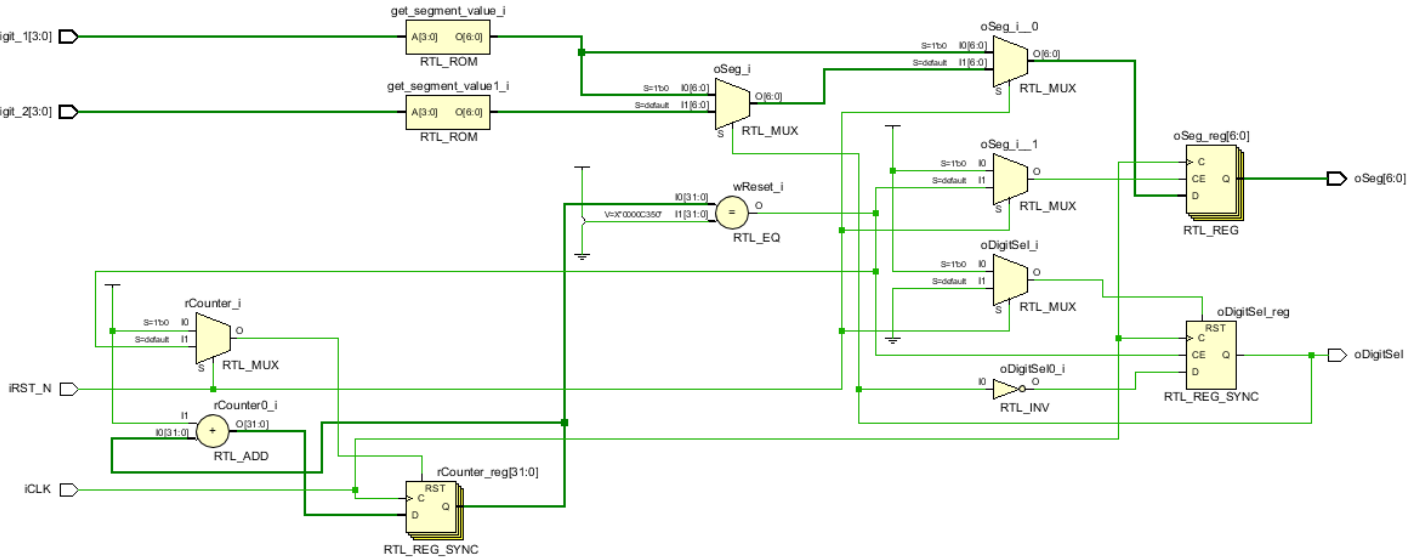


Figure 4 . FND RTL Schematic

FND 모듈은 Clk 기반으로 동작한다. 먼저 내부에 카운터를 구성하여 카운터가 설정한 값에 도달하면 wReset이 High가 되며 oDigitSel의 toggle 동작이 수행된다. oDigitSel은 두 세그먼트 중 어떤 세그먼트를 출력할지 결정하는 출력 제어 신호이다. 고속으로 스위칭하여 1개 digit씩 번갈아 표시하면 두 개의 세그먼트가 동시에 켜진 것으로 보인다.

Segment 디코딩은 Function으로 선언하여 4비트 값을 받아 해당 값을 적합한 패턴으로 변환하여 oSeg에 출력되는 구조로 설계하였다.

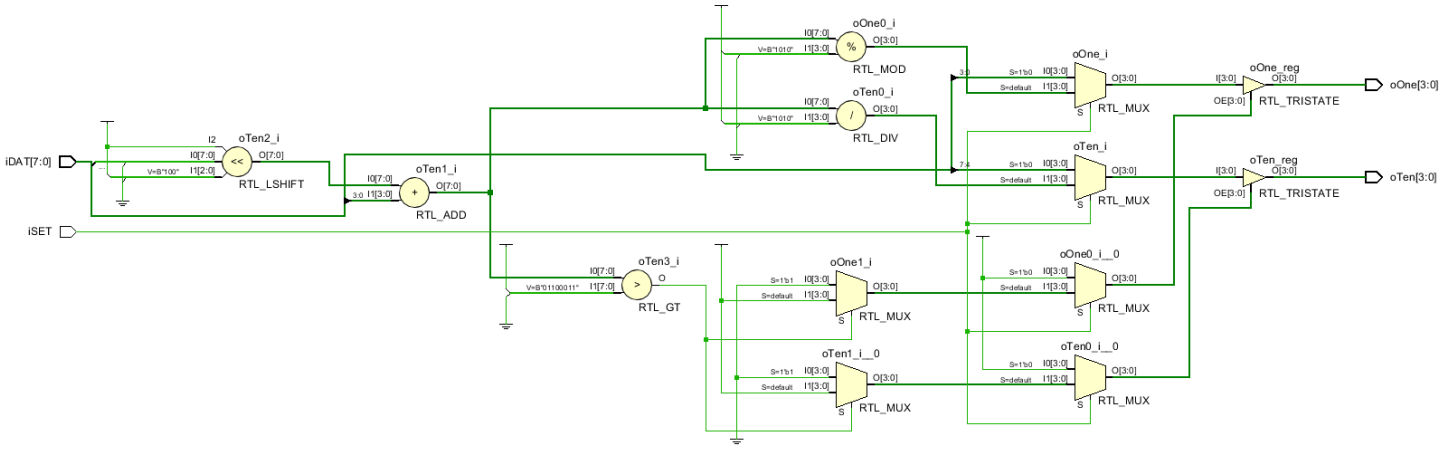


Figure 5. Hex2Dec RTL Schematic

다음은 Hex2Dec 모듈의 동작 설명이다.

Hex2Dec 모듈은 Combinational Logic으로 구성하여 asynchronous로 작동하도록 설계하였다.

iSET은 출력 모드를 설정하는 제어 신호로 0이면 16진수 출력 모드, 1이면 10진수 출력모드로 설정한다.

먼저 16 진수 모드이다 (iSET == 0). 이 모드는 입력 값을 그대로 16진수로 설정하여 상위 4비트와 하위 4비트를 각각 출력하게 된다. 따라서 iDAT의 상위 4비트인 iDAT[7:4]는 oTen으로 출력되고, iDAT[3:0]은 oOne으로 출력된다.

다음은 10 진수 모드이다 (iSET == 1). 이 모드는 입력 값 iDAT을 10진수로 변환하여 출력하여야 한다.

먼저 모듈 내부의 8비트 레지스터인 dec_value를 계산한다.

상위 4비트인 iDAT[7:4]를 왼쪽으로 4번 Shift 하여 iDAT[7:4]가 4 일 경우 다음과 같은 연산과 동일하다.

$$4 \times 16 = 64$$

Shift 연산은 곱셈과 나눗셈을 대체할 수 있기 때문에 곱셈과 나눗셈의 매우 긴 시간 지연을 해소하기 위하여 Shift 연산으로 대체하였다.

최종적으로 dec_value = iDAT[7:4] * 16 + iDAT[3:0]으로 계산되는 것과 동일하다.

10의 자리 출력인 oTen은 dec_value / 10으로, 1의 자리 출력인 oOne은 dec_value % 10으로 연산한다.

만약 dec_value가 99보다 크다면, oTen과 oOne을 Hi-Z 상태로 출력한다. (표현 가능 범위 초과)

II. Simulation Waveform

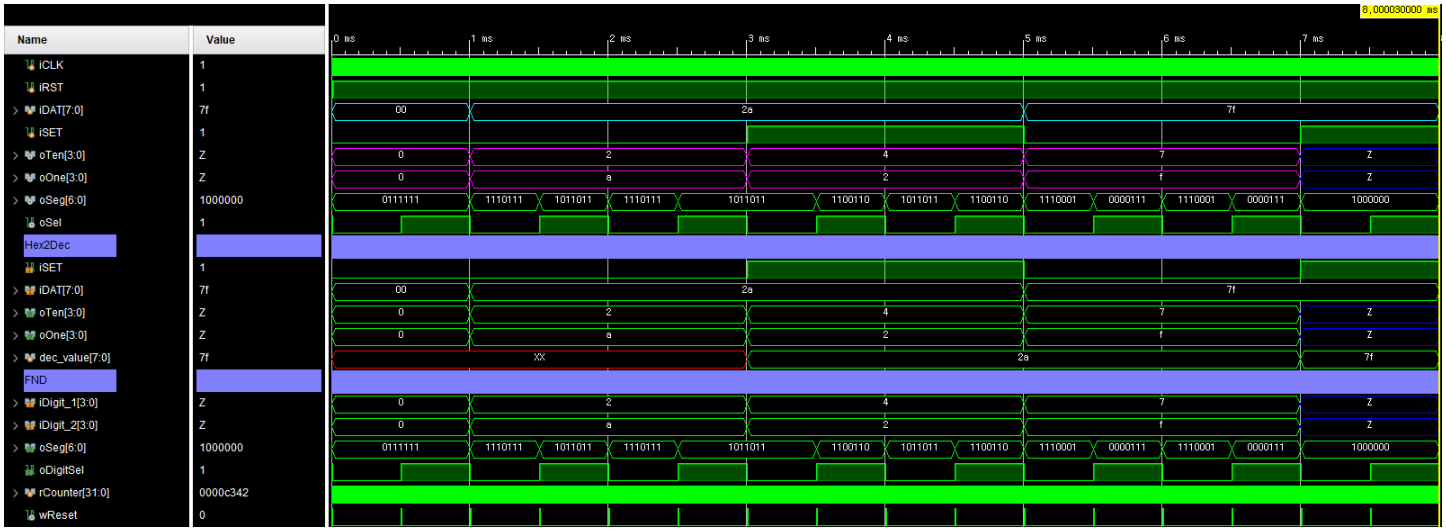


Figure 6 . Functional Simulation

먼저 Functional Simulation의 결과는 Figure 6에 나타난 바와 같다.

iSEL 신호가 1이면 iDAT을 10진수로 표현한 각 자릿수의 값이 oTen, oOne으로 정상적으로 나타나는 모습을 확인할 수 있다.

iSEL 신호가 0일때도 iDAT을 16진수로 표현한 각 자릿수의 값이 oTen, oOne으로 정상적으로 나타나는 모습을 확인할 수 있다.

또한 iCLK의 주기는 10ns로, 50μs 마다 oSel 신호가 Toggle되면서 두 개의 Segment에 각 자릿수의 값이 한 번씩 번갈아 가며 oSeg로 출력되는 파형을 관찰할 수 있다.

이로써 기능적인 동작은 정상적으로 수행되고 있음을 알 수 있다.

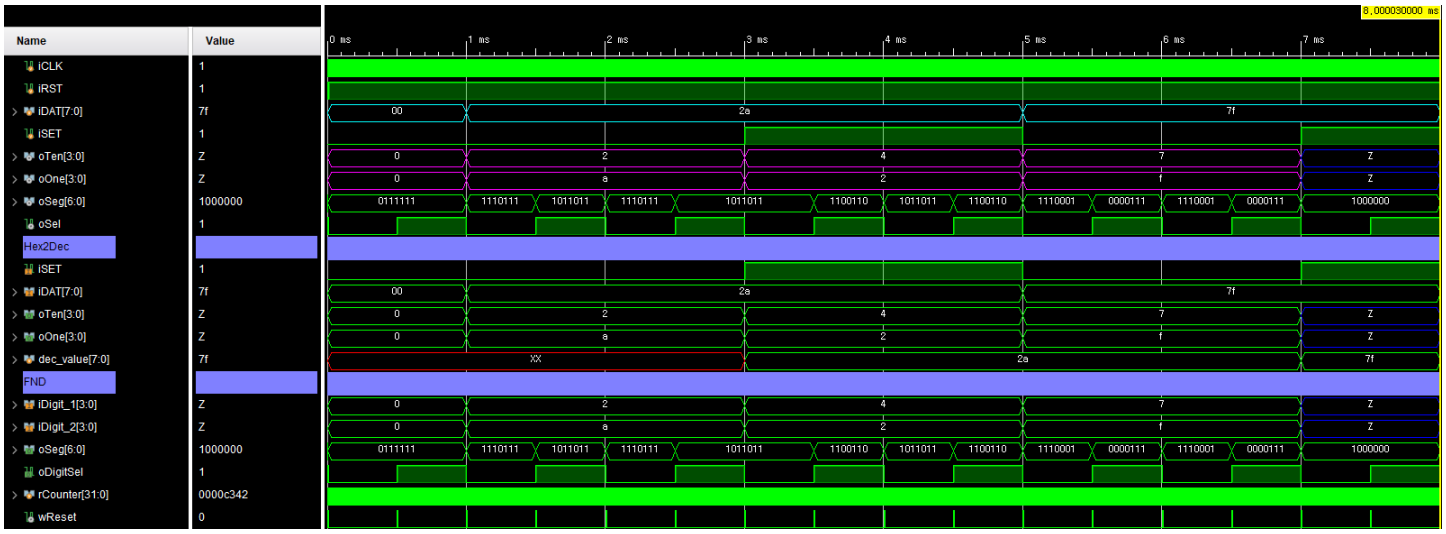


Figure 7 . Post-Synthesis Simulation

Synthesis 이후 생성된 Gate Netlist 파일로 Post-Synthesis Simulation을 수행하였다.

Functional Simulation과 같은 동작을 하는 모습을 확인할 수 있었으므로, 실제 HW 동작에서도 정상적인 기능을 할 것이라고 예측할 수 있었다.

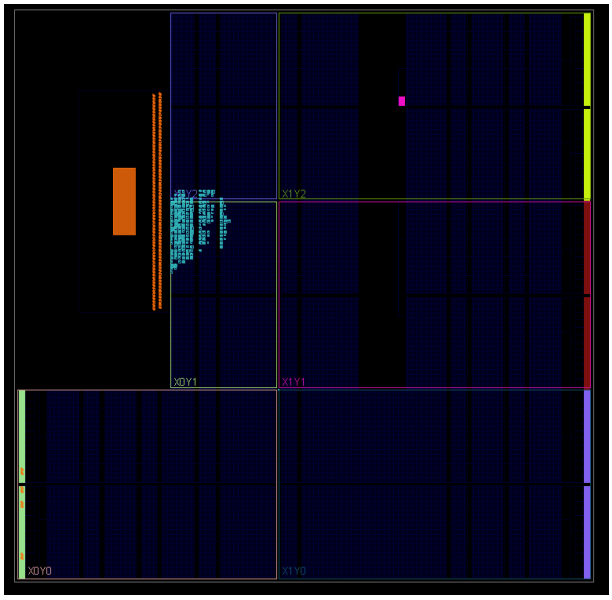
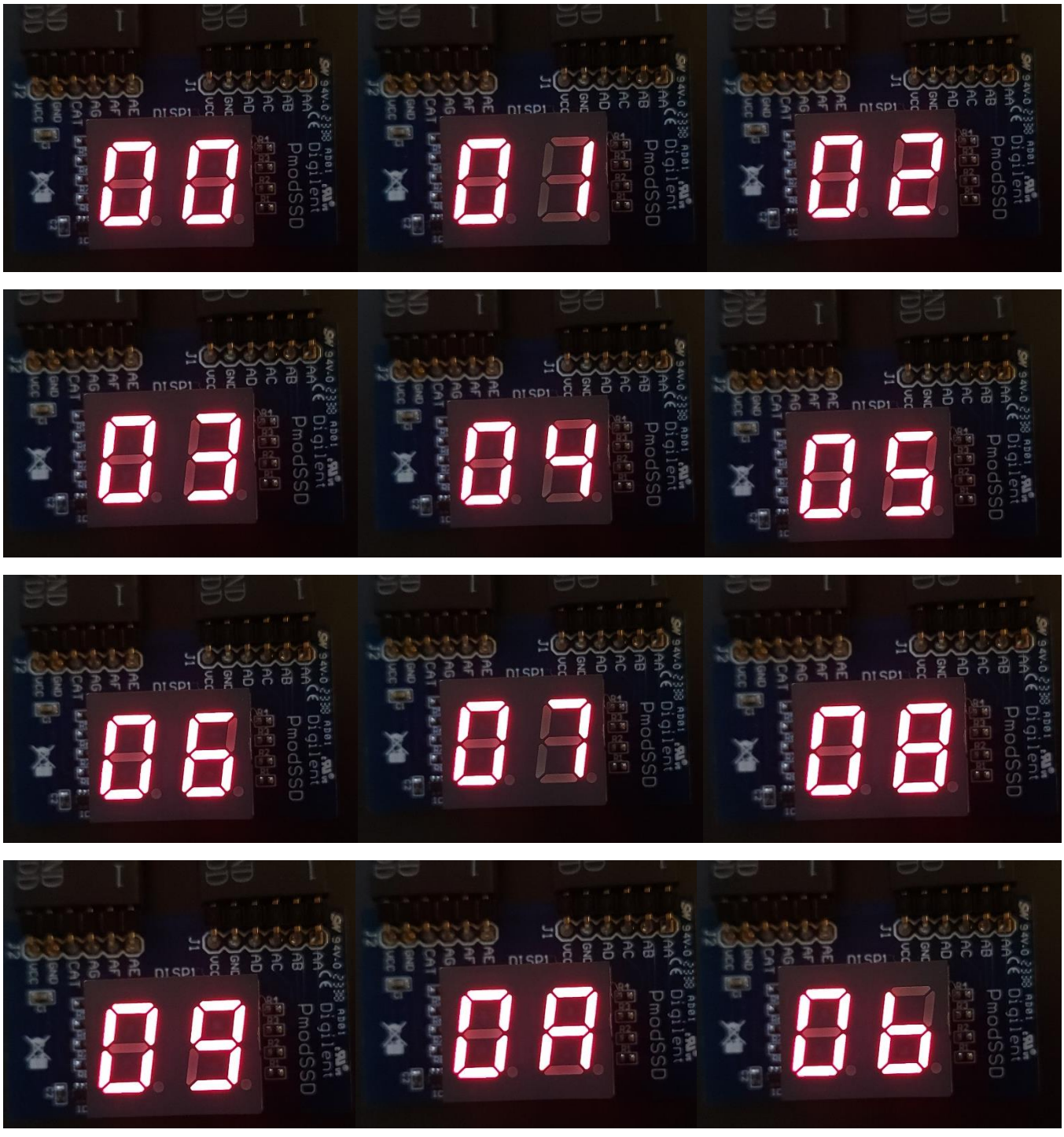
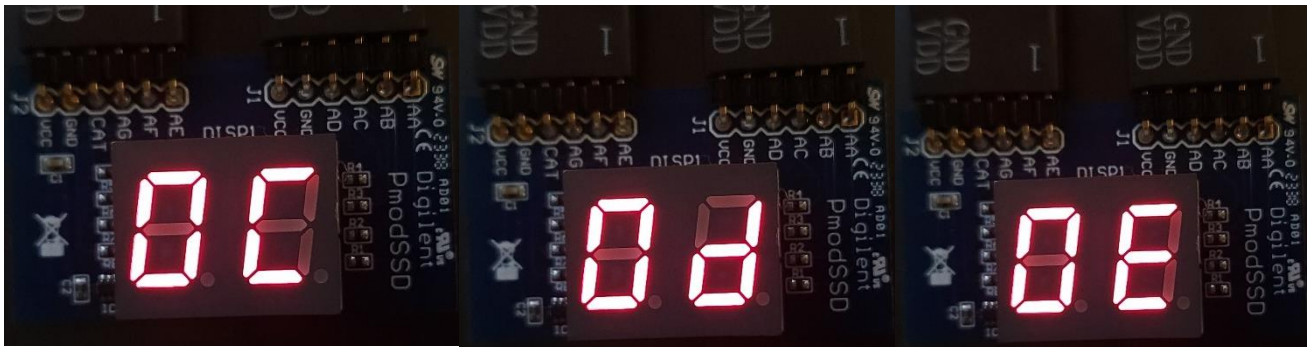


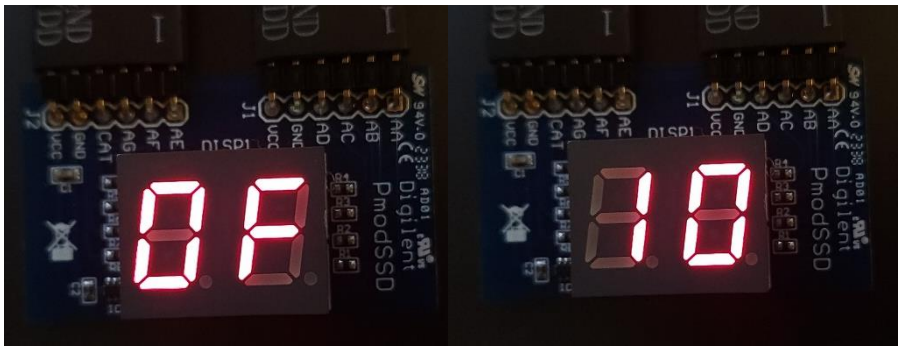
Figure 8 . Implement

III. Board Test 결과

첫 번째 Board Test - 16진수 output

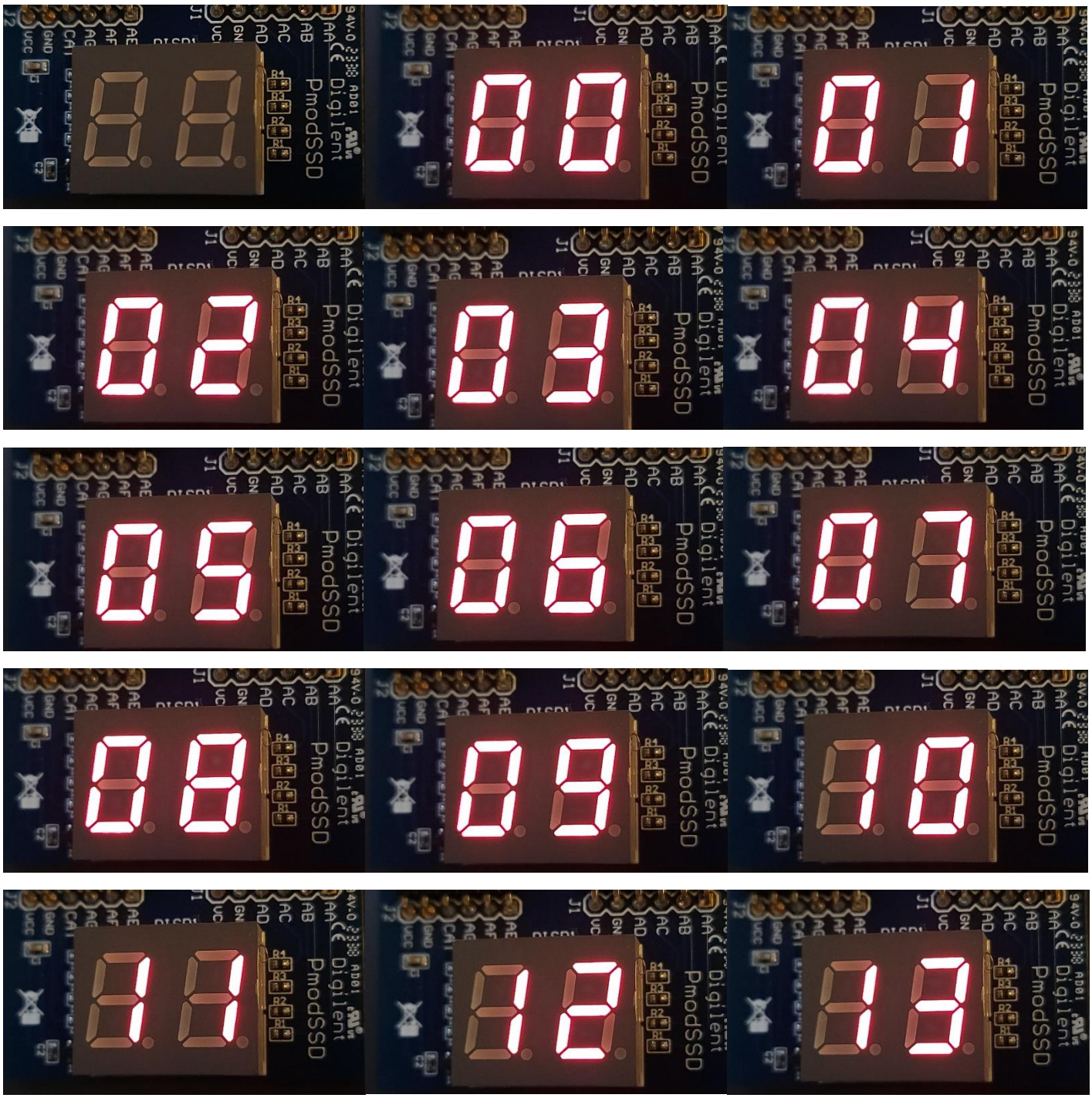






위 사진들을 확인하면 16진수로 순차적으로 Seven Segment에 정상적으로 표현되는 모습을 확인할 수 있다.

두 번째 Board Test - 10진수 output



위 사진들을 확인하면 10진수로 순차적으로 Seven Segment에 정상적으로 표현되는 모습을 확인할 수 있다.

IV. 결론

이번 과제를 통해 FND의 기능을 하드웨어 코드로 구현하고, 이를 기반으로 Board Test까지 성공적으로 수행하였다. AXI 인터페이스를 통해 프로세서와 Custom IP 간의 데이터 전송을 설정하고, FND 모듈과 Hex2Dec 모듈을 설계하여 16진수 및 10진수 값을 각각의 모드로 출력하는 동작을 확인할 수 있었다.

특히, 디지털 설계의 중요한 개념인 클럭 기반 카운터와 디지털 디코딩을 적용하여 안정적인 화면 출력이 가능하도록 구성하였으며, Shift 연산을 통해 곱셈 지연을 줄이는 등 최적화된 설계 기법도 적용할 수 있었다.

Functional Simulation과 Post-Synthesis Simulation을 통해 기본 동작 검증을 완료하였고, 실제 보드 테스트에서도 16진수와 10진수가 정상적으로 표시되는 것을 확인함으로써 설계의 신뢰성을 높일 수 있었다.

추가적으로 Constraints는 Pmod Header JB에 설정하였다. 만약 JB와 JC의 상위 part를 사용하고 싶다면 Constraints의 수정이 필요하다.

본 과제를 통해 AXI 인터페이스와 Custom IP를 직접 설계하여 하드웨어의 동작을 설계하는 일련의 과정을 경험할 수 있었다.