컴퓨터 공학 기초 실험2 보고서

실험제목: Counter & Shifter

실험일자: 2023년 10월 13일 (금)

제출일자: 2023년 10월 24일 (월)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 금요일 0, 1, 2

학 번: 2020202037

성 명: 염정호

1. 제목 및 목적

A. 제목

Counter & Shifter

B. 목적

Shifter와 counter의 원리를 이해하고 이를 베릴로그를 이용해 구현해 본다. 이를 통해 레지스터와 순차회로의 동작방법을 이해하고, shifter와 counter의 동작 과정을 이해한다.

2. 원리(배경지식)

Mealy machine : 현재 상태와 입력에 의해 출력이 결정되는 유한상태 기계이다.

즉, 각 상태와 입력에 대해 최대 한번의 전환이 가능하다.

Moore machine: 현재 상태에 의해서 출력값이 결정되는 유한상태 기계이다. 입력으로 인해 상태가 변해 간접적으로는 출력에 영향을 미치긴 하지만 현재 출력 또는 즉각적인 출력에 영향을 미치지 않는다.

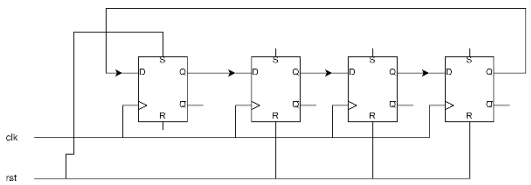
차이점

1. Mealy머신의 상태(state)수가 더 적다.
2. Moore machine은 더 안전하다.
   -출력이 클럭엣지에서만 변경된다.(항상 한 사이클 이후)
   -mealy머신의 경우 입력 변경으로 인해 로직이 완료되자마자 출력 변경이 발생할 수 있다. 두 머신이 연결될 경우 비동기 피드백이 발생하게 될 수 있다.
3. Mealy머신의 경우 기계에 사용하기 좋다.
   - 동일한 주기로 반응한다. Clock을 기다릴 필요가 없다.
   - 무어 머신의 경우 상태 출력을 디코딩하기 위해 더 많은 로직을 요구한다. Clk edge이후에 gate delay가 발생한다.

Ring counter: shift register에 연결된 filp -flop으로 구성된 카운터 유형이다.

마지막 flip flop의 출력이 첫 번째 flip flop의 입력으로 공급되어 원형 구조를 만들게 된다.

하드웨어 설계에서 유한상태기계를 구성할 때 많이 사용된다. 링 카운터의 경우 delay가 코드의 비트 수와 관계 없이 일정하고 N개의 상태만 나타낼 수 있다.



링카운터 구조

Counter: 2개 이상의 플립플롭으로 구성되며 매 입력 펄스마다. 정해진 순서대로 주기적으로 변하는 순서논리회로 또는 레지스터.

- 발생 횟수를 기록하거나, 동작 순서를 제어하기 위한, 타이밍 신호를 생성하기에 적합하다.
- 클럭 펄스 인가에 따라 비동기식, 동기식 카운터로 구분된다.
  - ■ 비동기식 카운터 : 클록 펄스에 동기화 되지 않으면서 동작한다
  - ■ 동기식 카운터: 클록 펄스 발생시 모든 플립플롭이 병렬로 동기화 되어 동작한다.

Shift register : 플립플롭의 출력이 다음 플립플롭에 연결되는 회로이다. 단일 클럭 신호로 이용됨으로 시스템에서 저장된 데이터가 한 위치에서 다음 위치로 이동 할 수 있다.

3. 설계 세부사항

Shifter8

Shtift8모듈의 경우 op값에 따라 다른 쉬프트 연산을 수행하게 된다.

- NOP : No operation(현재 register 의 값을 그대로 출력)
- Load : 입력된 data를 출력
- LSL : Logical shift left 를 수행
- LSR : Logical shift right 를 수행
- ASR : Arithmetic shift right 를 수행

Nop와 load의 경우 각자 output또는 처음input값을 출력시키며 나머지 연산의 경우 shitft를 통해 출력을 갱신해 준다

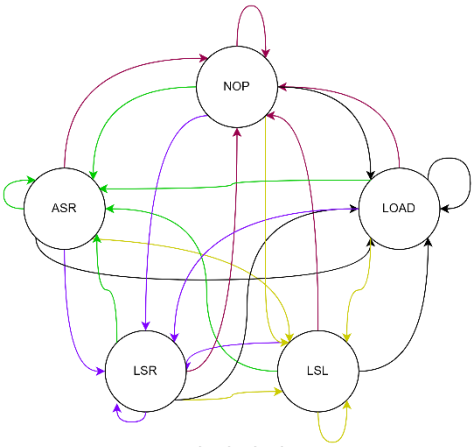
LSL : 입력받은 8bit를 왼쪽으로 shift연산을 진행시켜 준다. 옮겨진 오른쪼 bit들은 0으로 채워지게 된다.'

LSR : 입력받은 8bit데이터를 오른쪽으로 shift연산을 진행한다. 옮겨진 데이터 들은 0으로 채워지게 되며 이 경우 음수가 양수로 변하게 됨을 주의해야 한다.

ASR : LSR과 하는 일은 유사 하지만 왼쪽 bit를 msb로 채운 다는 점에서 차이가 있다.

NOP	000
LOAD	001
LSL	010
LSR	011
ASR	100

state



상태전이

8 bit Loadable Up/Down Counter

입력에 따라 상태를 변화시키고 감산, 리셋, 가산, 출력 등의 연산을 진행한다.

IDLE : reset이 되었을 때, count값을 0으로 하는 state

LOAD : 입력 data를 count 값에 할당하는 state

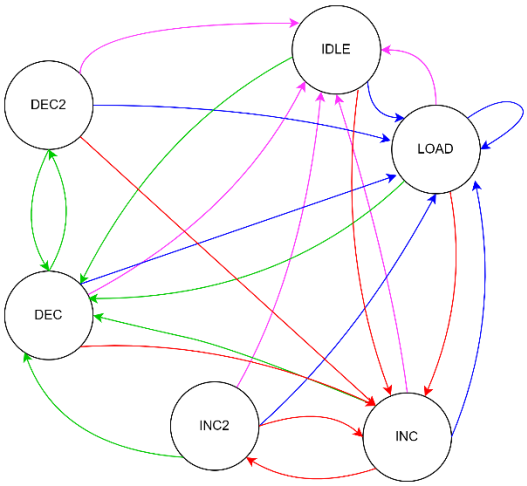
INC, INC2 : count 값을 증가하기 위한 state

inc 가 1인 동안 두 state로 서로 이동하며 값을 증가시킨다.

DEC, DEC2 : – count 값을 증가하기 위한 state

inc 가 0인 동안 두 state로 서로 이동하며 값을 감소시킨다.

각 입력이 동시에 활성화 되었을 때 reset , load, inc순서대로 우선순위를 갖는다.



상태전이

Ns_logic : 현재 상태와 활성화 된 bit들을 통해 다음 상태를 출력한다.

Os_logic : 입력받은 데이터를 현재 상태에 맞게 가공한다. Inc -> 가산 dec ->감산, load -> 입력 데이터 출력, reset 데이터 초기화

Register : ns_logic로부터 받는 현재 상태를 ns_logic의 출력 값으로 바꿔

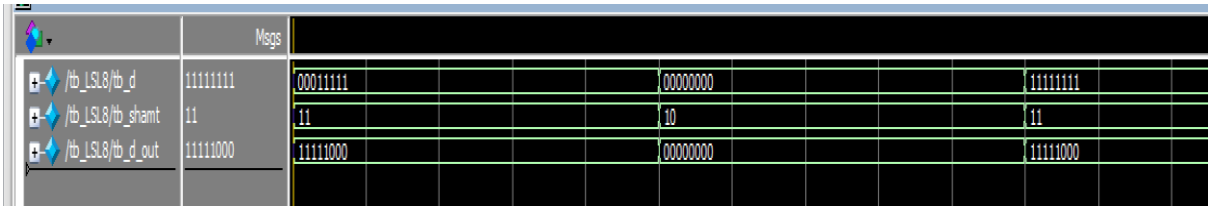
상태를 전이 시켜주는 역할을 한다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

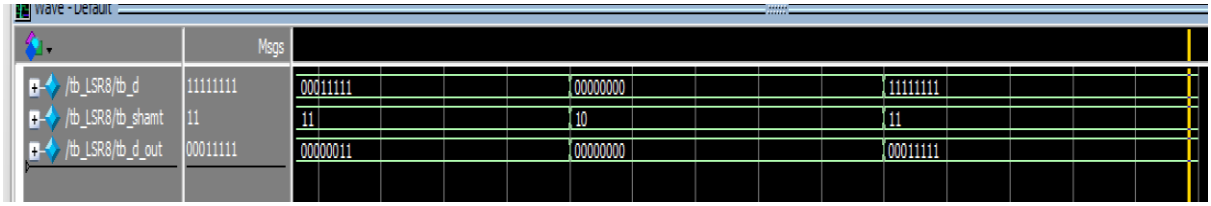
Shifter8-Submodule

LSL8



시뮬레이션 결과 shamt값 만큼 왼쪽으로 비트가 이동했으며, 오른쪽 비트들은 0으로 채워졌다. 연산 결과 8bit가 넘어가는 경우 해당 bit는 사라진다 즉 8bit를 이동 할 경우 0으로 초기화가 가능하다.

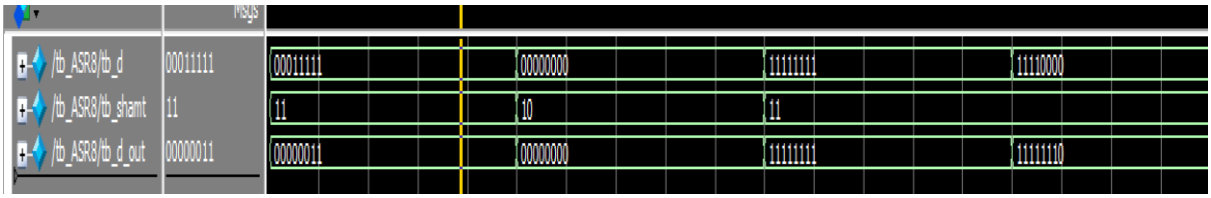
LSR8



시뮬레이션 결과 shamt값 만큼 오른쪽으로 비트가 이동했으며, 왼쪽 비트들은 0으로 채워졌다. 즉 8bit를 이동 할 경우 0으로 초기화가 가능하다.

주의해야 할 점은 오른쪽으로 당겨진 bit는 0으로 채워지기 때문에 음수의 경우 양수로 바뀌게 된다.

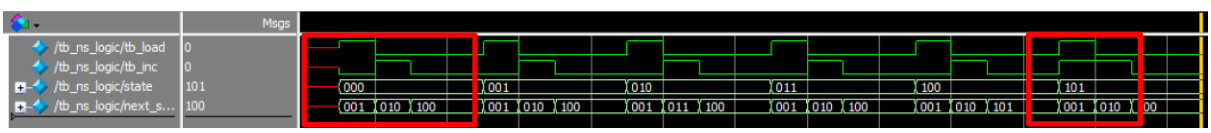
ASR8



시뮬레이션 결과 shamt값 만큼 오른쪽으로 비트가 이동했으며, 왼쪽 비트들은 MSB로 채워졌다. 즉 LSR에서 생기는 부호가 바뀌게 되는 문제가 발생하지 않는다.

cntr8-Submodule

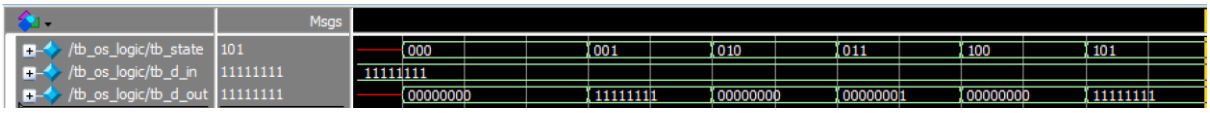
ns_logic



활성화된 bit에 따라 load. Inc dec모드로 이동하는 것을 확인 할 수 있고

비교 순서는 reset ->load->inc순서이다. Dec의 경우 inc가 0 이면 dec가 활성화 된 것으로 간주한다.

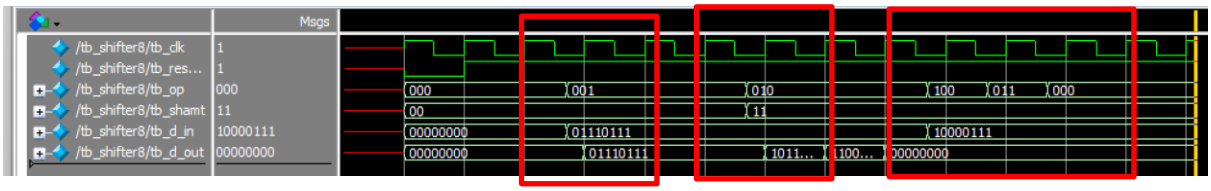
Os_logic



Inc가 활성화 됐을 땐 가산이 시작 되고 inc가 비활성화 되었을 때 가산이 시작되는 것을 확인했다.

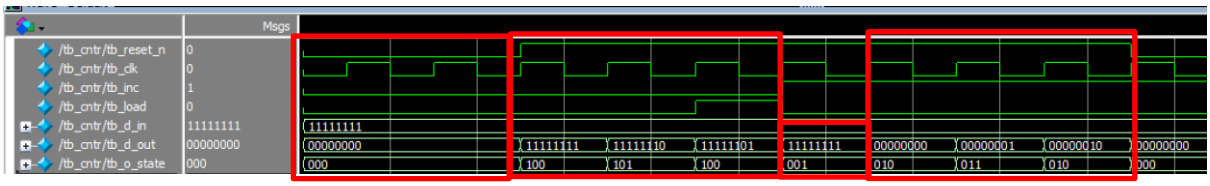
Main module

Shifter8



1. Load -처음 입력 값이 그대로 출력 되는 것을 확인.
2. LSL 3 – 입력된 값이 왼쪽으로 3bit 이동해서 출력
3. ASR 3 -입력 된 값이 오른쪽으로 3bit 이동 한 후 왼쪽 bit를 MSB로 채워졌다.
4. LSR 3 – 입력된 값이 오른쪽으로 3bit 이동 후 왼쪽 bit가 0으로 채워지는 모습
5. NOP – 아웃풋 값이 그대로 출력

cntr8

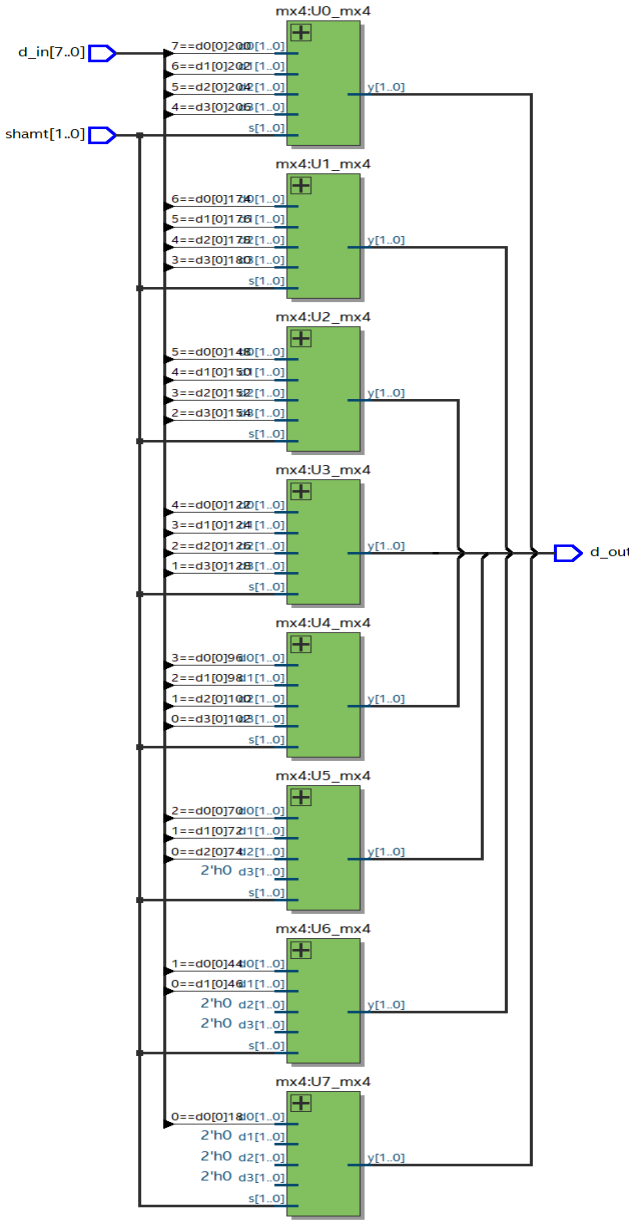


1. reset이 활성화 되어있을 땐 state와 output모두 0이된다.
2. Load와 inc모두 0 이 되었을 때 dec가 활성화 됨으로 감산이 시작 되는 것을 확인 할 수 있다.
3. load명령어 사용 시 input값을 출력한다.
4. Load가 0이고 inc활성화시 덧셈 연산을 실행한다.
5. Reset이 활성화 되면 다시 0으로 초기화 된다.

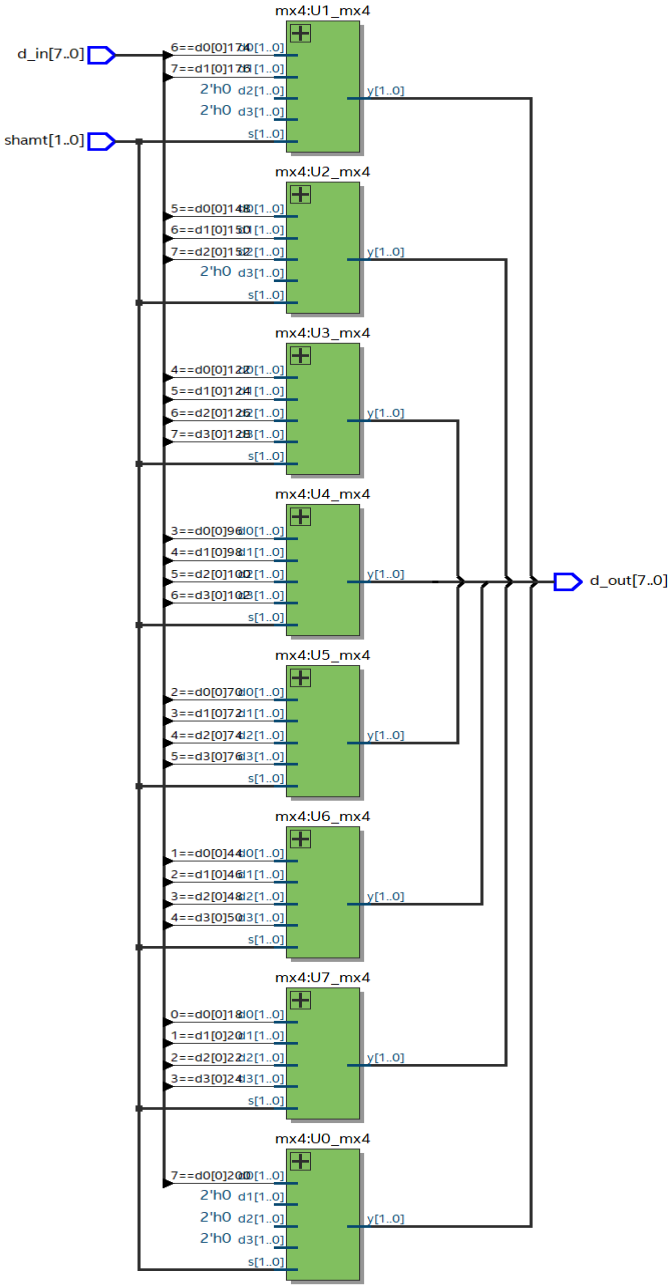
B. 합성(synthesis) 결과

Shifter8-Submodule

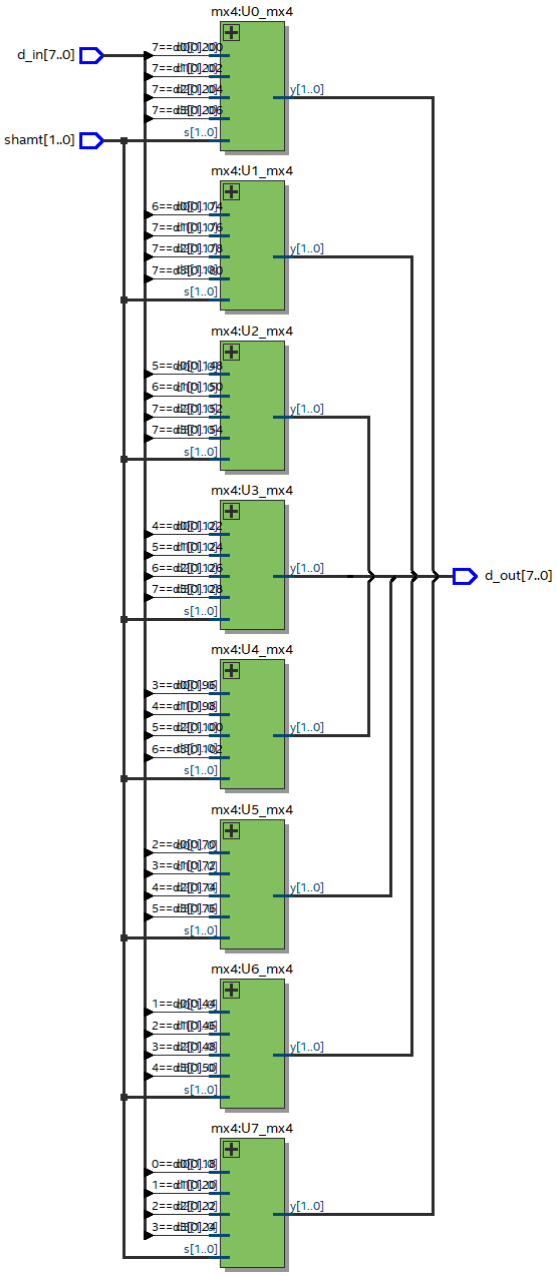
LSL8



LSR8

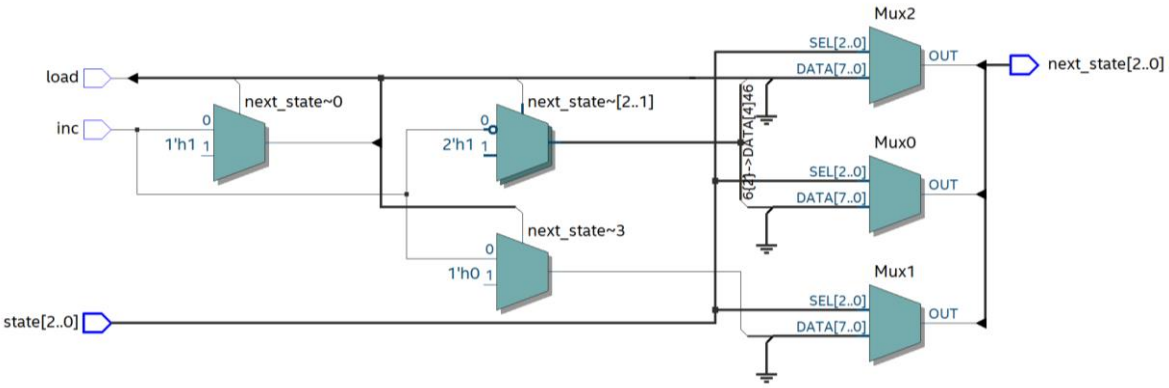

mx4:U1_mx4
mx4:U2_mx4
mx4:U3_mx4
mx4:U4_mx4
mx4:U5_mx4
mx4:U6_mx4
mx4:U7_mx4
mx4:U0_mx4
d_in[7..0]
shamt[1..0]
d_out[7..0]
y[1..0]
s[1..0]
2'h0 d2[1..0]
2'h0 d3[1..0]

ASR8

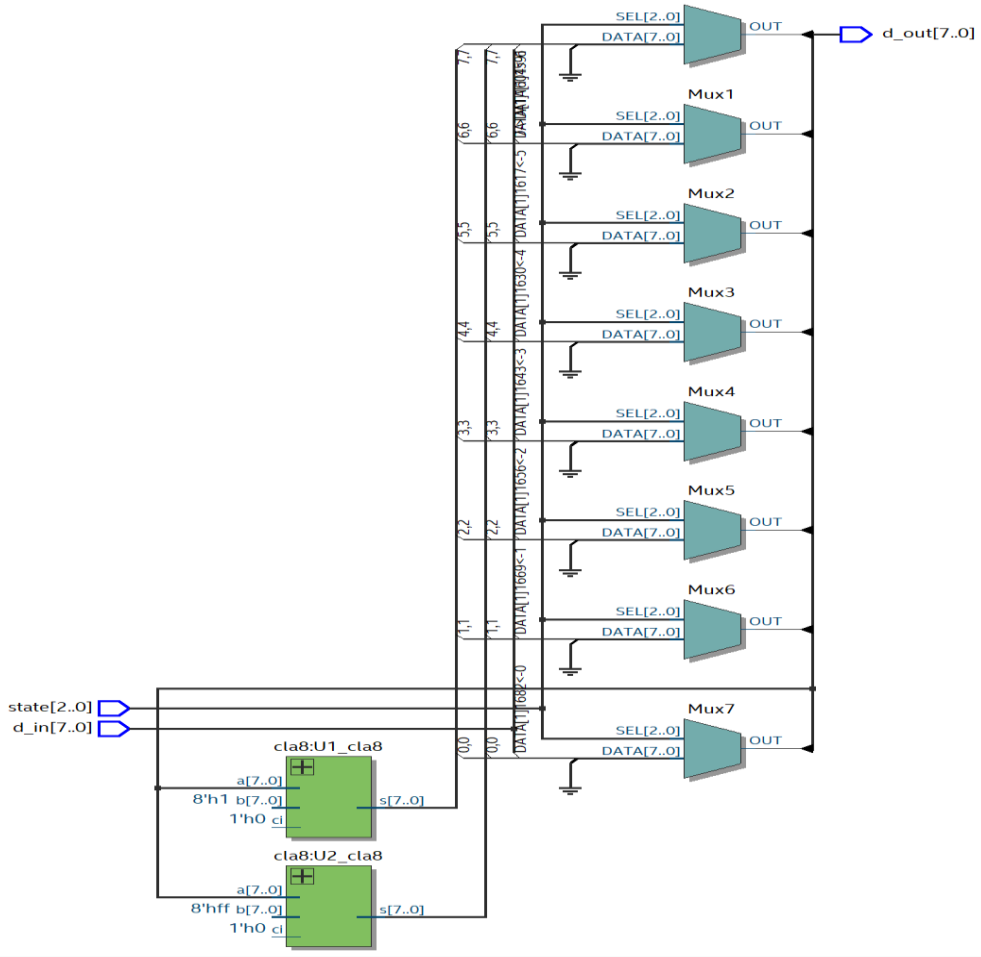


cntr8-Submodule

ns_logic

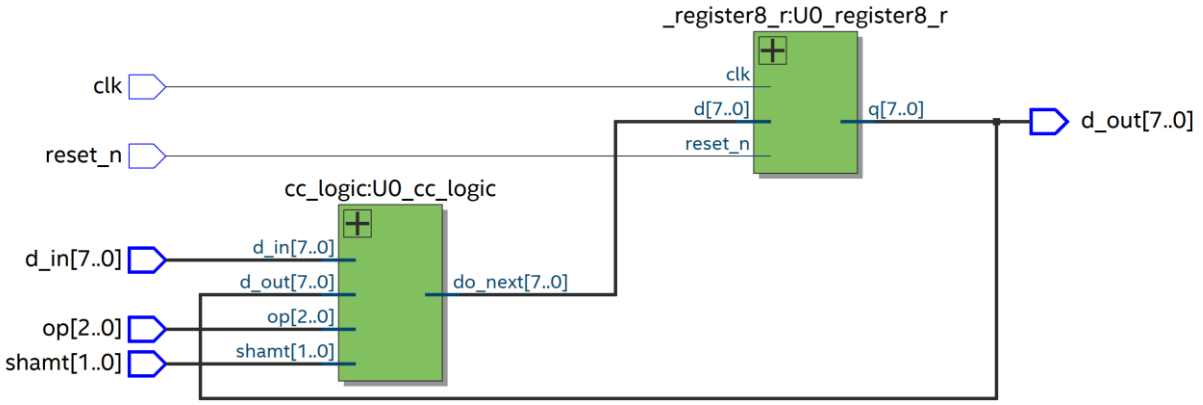

Mux2
SEL[2..0]
DATA[7..0]
OUT
next_state[2..0]
load
inc
next_state~0
1'h1
next_state~[2..1]
2'h1
next_state~3
1'h0
Mux0
Mux1
state[2..0]

os_logic


Mux7
Mux6
Mux5
Mux4
Mux3
Mux2
Mux1
SEL[2..0]
DATA[7..0]
OUT
d_out[7..0]
state[2..0]
d_in[7..0]
cla8:U1_cla8
cla8:U2_cla8
a[7..0]
b[7..0]
s[7..0]
8'h1
1'h0
8'hff

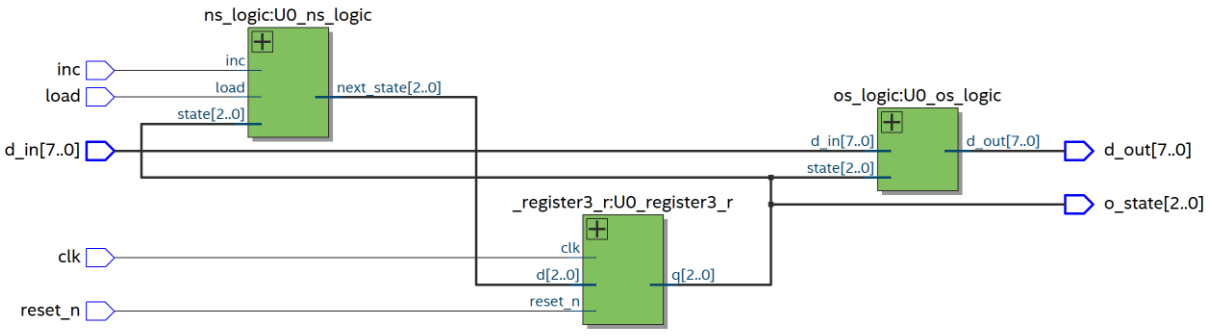
Main moduel

Shifter8



Flow Status	Successful - Sun Oct 22 08:24:39 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	shifter8
Top-level Entity Name	shifter8
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	8
Total pins	23

Cntr8



Flow Status	Successful - Sun Oct 22 08:23:04 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cntr8
Top-level Entity Name	cntr8
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	3
Total pins	23

5. 고찰 및 결론

A. 고찰

베릴로그의 tb작성시 오류가 많이 나와 고생했다. 서브 모듈이나 메인 모듈의 코드 에러의 경우 코드가 돌아가지 않지만 tb코드의 경우 일단 실행이 되기 때문에 잘 찾기 어려웠다. 그래서 warning부분을 확인했는데 reg의 bit수나 모듈이름이 잘못 된 경우를 표시해준다는 것을 알게 되어 warning부분도 컴파일시 확인해야 한다는 것을 배웠다.
counter구성 할 때 값이 누적 되어 감산, 가산 실행하지 않고 처음 input에 대해서만 연산을 실행하는 것을 확인했다. cla모듈 구성에서 처음 데이터 a값에 d_out이 아닌 d_in을 넣어 계속 초기 데이터에 대한 연산을 실행하는 것으로 확인 되었다.
counter에서 비교 우선순위를 지정할 때 처음에 case 문을 이용하려고 했으나. 연산 우선순위를 확실하게 표시하고 이용하기 위해 if else문을 이용했다.

loadable counter는 특정 횟수나 펄스를 세기 전에 특정한 값을 미리 설정할 수 있는 디지털 카운터 회로를 의미합니다. 특정한 수의 사건이나 펄스를 세어야 하는 응용 분야에서 유용합니다. 즉 마이크로 컨트롤기반의 시스템 및 디지털 신호처리 응용 분야에서 주로 사용된다.
Ex) 자동화 시스템 및 공정 제어, 데이터 통신 및 프로토콜 분석, 빈도수 및 주파수 측정, 시간 지연 및 타이밍 제어, 주파수, 합성 및 발생기, 이벤트 및 펄스 세기.

Ring counter 용도: 회로 동기화 및 타이밍 제어, 지원 다중화, 패턴 생성, 버퍼 레지스터, 시프트 레지스터 등.

barrel shifter란 순수한 조합 논리만을 이용하여 순차논리를 사용하지 않고 지정된 비트0수 만큼 데이터를 이동 할 수 있는 디지털 회로이다. 즉 이진 산술 연산을 제공하는 것이다. 하지만 고정된 비트를 사용하는 경우 shift와 같은 단항 연산을 구현하는 데에도 사용 할 수 있다. 구현 방법 중 하나는 멀티플렉서의 출력 시프트 거리에 따라 다음 멀티플렉서의 입력에 연결되는 일련의 멀티플렉서 이다.
n – bit의 길이를 가지는 register를 n-bit 만큼 shift연산을 진행하기 위해서는
N*log2(n)개의 2-1 mux가 필요하다
즉 128bit의 경우128 * 7 = 896개의 mux가 요구된다.

B. 결론

이번 시험을 통해 FSM의 동작 구성 방법을 이해할 수 있었다. 이전 traffic light실험에서도 FSM을 사용하였지만. 상태 전이가 다양하지 않았던 반면, 이번 실험에서는 한 상태에

서 여러가지 상태로 넘어가는 것이 가능했고, 각각의 상태마다 단순한 출력이 아닌 연산을 통한 출력을 이용해야 했기 때문에 더 많은 경험과 지식이 필요했다.

6. 참고문헌

이준환 / 디지털논리회로2/광운대학교/2023

유지현 / 디지털논리회로1/광운대학교/2023

이형근 / 컴퓨터공학기초실험2 / 광운대학교/ 2023

COUNTER/http://www.ktword.co.kr/test/view/view.php?m_temp1=4561

Mealy machine//https://en.wikipedia.org/wiki/Mealy_machine