컴퓨터 공학 기초 실험2 보고서

실험제목: Counter & RF

실험일자: 2020년 10월 12일 (월)

제출일자: 2020년 10월 19일 (월)

학 과: 컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2019202052

성 명: 김 호 성

1. 제목 및 목적

A. 제목

- Counter
- RF

B. 목적

- cntr5를 구현한다.
- shifer8을 구현한다.
- cntr8을 구현한다.
- Register_file을 구현한다.

2. 원리(배경지식)

- Counter

펄스신호에 따라 어떤 정해진 순서대로 상태의 변이가 진행되는 레지스터를 counter라고 한다. Counter는 어떤 사건이 발생할 때마다 펄스신호를 만들어 그 사건의 발생횟수를 세는 등에 사용된다.

- shifter

Register에 저장되어 있는 정보를 단방향이나 양방향으로 이동시킬 수 있는 하드웨어이다. flip-flop을 포함하고 있어 회로에 저장능력이 있다.
즉, 순차회로의 출력은 현재의 입력 값과 현재 flip-flop에 저장되어 있는 값에 의해 결정된다.

- Register File

Write는 write enable(we)에 의해 활성화된다.
Write operation: Decoder를 통해 address를 해석하여 해당 register enable
Read operation: MUX를 통해 n개의 register 중 한 개를 선택한다.
Write logic의 경우 user로부터 wAddr를 받아 n개의 register 중 한 개를 선택한 뒤 we이 1일 때 register에 값을 쓴다.
n개의 m-bit register 중 write logic에 의해 선택되어진 register는 user로부터 받은 data를 저장한다.
Read logic은 user로부터 rAddr를 받아 n개의 register 중 한 개를 선택하여 값을 출력한다.

- Moore FSM과 Mealy FSM의 장단점

Mealy state machine에서 Decoder는 조합회로, Memory Elements는 순차회로로 FSM을 동작한다. FSM의 출력이 Output Decoder와 Next State Decoder로 입력되는데, 이 때의 출력값이 Next State Decoder로 입력되어 다음 상태를 결정한다. (이 부분은 Moore와 동일)
그러나 Output Decoder에서 출력되는 Outside World Outputs의 값을 결정할 때가 다른데, Mealy state machine은 Output Decoder에 외부의 어떤 입력 값과 FSM의 출력 값 모두가 영향을 주지만, Moore state machine은 그렇지 않고, 오로지 FSM의 출력 값에 의해서만 Outside World Outputs가 결정된다.
Mealy의 경우 flip-flop을 사용하지 않는 신호가 계속 전달되는 단점이 있어 회로의 동작 속도에 제한을 가하기 때문에 여러가지 문제가 발생한다.

- ring counter

마지막 flip-flop의 출력이 첫 번째 입력에 공급되어 원형 또는 링 구조를 만드는 shift register에 연결된 flip-flop으로 구성된 counter유형이다.
Ring counter에는 두 가지 유형이 있다.

- straight ring counter

마지막 shift register의 출력을 첫 번째 shift register input에 연결하고 ring 주위에 단일 bit를 순환시킨다.

- twisted ring counter(= johnson counter)

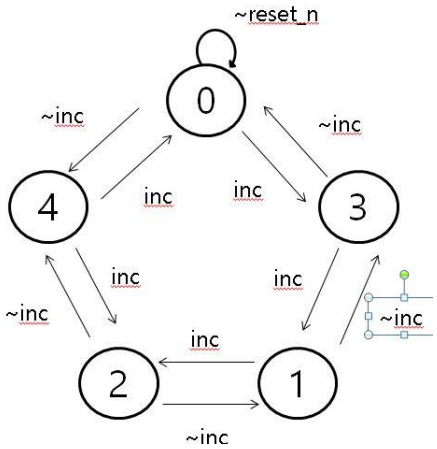
마지막 shift register output의 보수를 첫 번째 register의 input에 연결하고 1의 스트림을 순환시킨다.

- stack과 queue

Stack은 쌓아 올린다는 것을 의미하며 차곡차곡 쌓아 올린 형태의 자료구조를 말한다.
Stack의 경우 같은 구조와 크기의 자료를 정해진 방향으로만 쌓을 수 있고, top으로 정한 곳을 통해서만 접근할 수 있다. top에는 가장 최근에 들어온 자료를 가리키고 있으며, 삽입되는 새 자료는 top이 가리키는 자료의 위에 쌓이게 된다.
Last in first out방식의 자료구조이다.
Queue는 줄을 서서 기다리는 것을 의미하며, First in first out방식의 자료구조이다.
정해진 한 곳을 통해서 삽입, 삭제가 이루어지는 stack과는 달리 queue는 한쪽 끝에서 삽입 작업이, 다른 쪽 끝에서 삭제 작업이 양쪽으로 이루어진다.
이때 삭제 연산만 수행되는 곳을 front 삽입 연산만 이루어지는 곳을 rear로 정하여 각각의 연산작업만 수행된다. 이때 큐의 rear에서 이루이지는 삽입 연산을 enQueue, front에서 이루어지는 삭제 연산을 dnQueue라고 부른다.

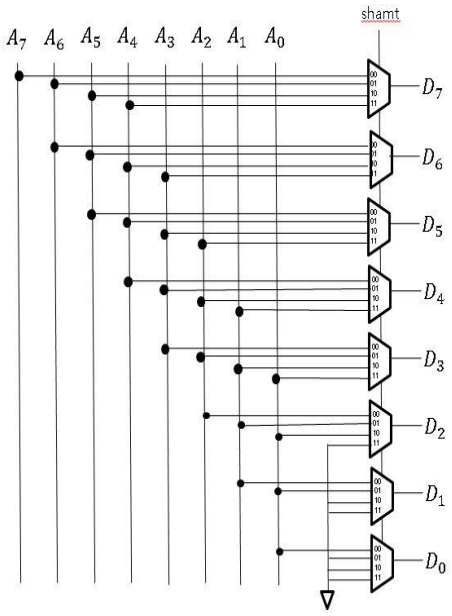
3. 설계 세부사항

- cnt5



위의 사진은 cnt5에 대한 회로도다. Inc의 값에 따라서 계속적으로 좌우로 0~4까지 오름차순 내림차순으로 움직이게 된다. 이 회로를 만들 때는 어떤 module도 instance를 하지 않았으며, parameter를 통해서 값을 집어넣는 방식을 이용하였다. 이 회로도는 clock pulse가 rising edge일 때만, 움직이게 설정되어 있으며, 모든 케이스에 대해서 설정해줘야 한다.

- shifer8



Shifter는 register에 저장된 data를 좌측 또는 우측으로 이동시키는 하드웨어이다. 어셈블리 시간에 배운 설명으로는 왼쪽으로 한 칸 shift할 경우 2배가 되고, 오른쪽으로 한 칸 shift할 경우에는 값이 절반이 된다.

(단, shift를 하기 전의 숫자의 구성과 모양이 shift를 했을 때 변하지 않고, 값이 사라지지

않는다는 전제하에 이루어진다.)

Signal

Reset_n: low에서 작동하는 reset signal로 register 내부의 값을 0으로 초기화 시킨다.

op

1. NOP: Not operation 현재 register에 저장되어 있는 값을 그대로 출력한다.
2. LOAD: 입력된 data 출력
3. LSL: Logical Shift Left를 실행
4. LSR: Logical Shift Right를 실행
5. ASR: Arithmetic Shift Right를 실행

Shamt: Shift Amount의 약자로 몇 칸을 움직일지 설정

- cntr8

8-bit CLA: 이전 실습에서 구현하였던 4-bit CLA를 instance하여 8-bit CLA를 구현

=> counter에서 값을 증가시키거나 감소하는데 사용한다.

Define states

1. IDLE: reset되었을 때, count 값을 0으로 하는 state
2. LOAD: 입력 data를 count 값에 할당하는 state
3. INC, INC2: count 값을 증가하기 위한 state

=> inc가 1인 동안 두 state로 서로 이동하며 값을 증가시킨다.

4. DEC, DEC2: count 값을 감소시키기 위한 state

=> inc가 0인 동안 두 state로 서로 이동하며 값을 감소시킨다.

IDLE: 3'b000

LOAD: 3'b001

INC: 3'b010

INC2: 3'b011

DEC: 3'b100

DEC2: 3'b101

Next_state_logic -> register3 -> Output_logic 순서대로

- Register file

Write: write enable(we)에 의해 활성화된다.

Write operation: Decoder를 통해 address를 해석하여 해당 register enable

Read operation: MUX를 통해 8개의 register 중 한 개 선택

n-to-2^n decoder: 입력 값을 이용해 알맞은 출력으로 변환

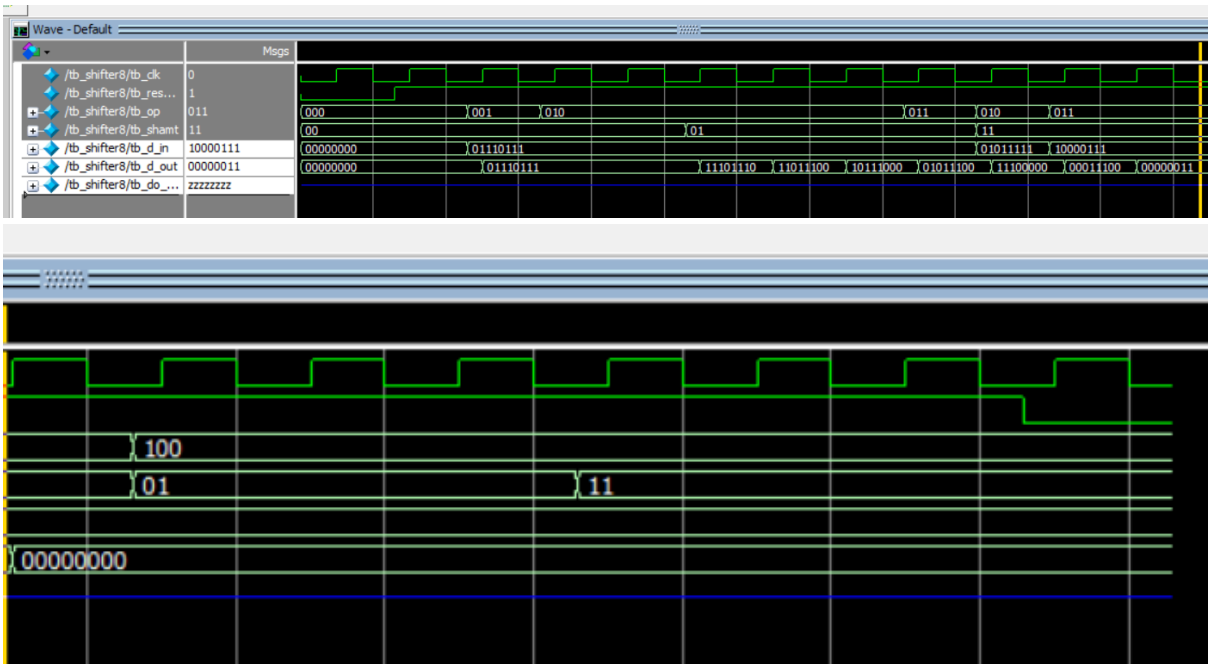
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

- cnt5

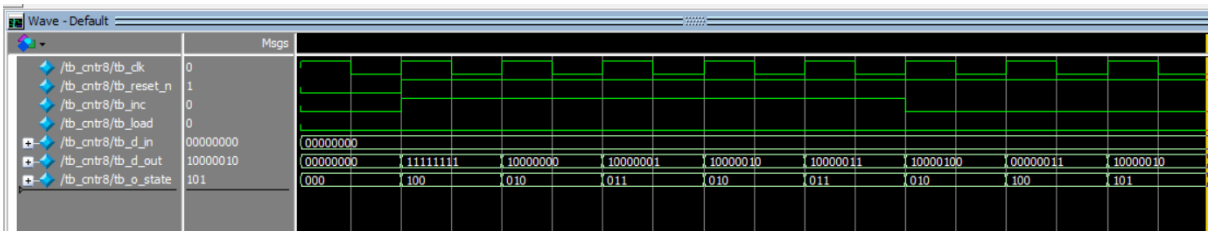
Reset이 0일 때는 값이 바뀌지 않고, reset이 1일 때에만 바뀐다. Inc가 0과 1을 번갈아 가면서 count의 값이 증가하고 감소하는 것을 확인했다.

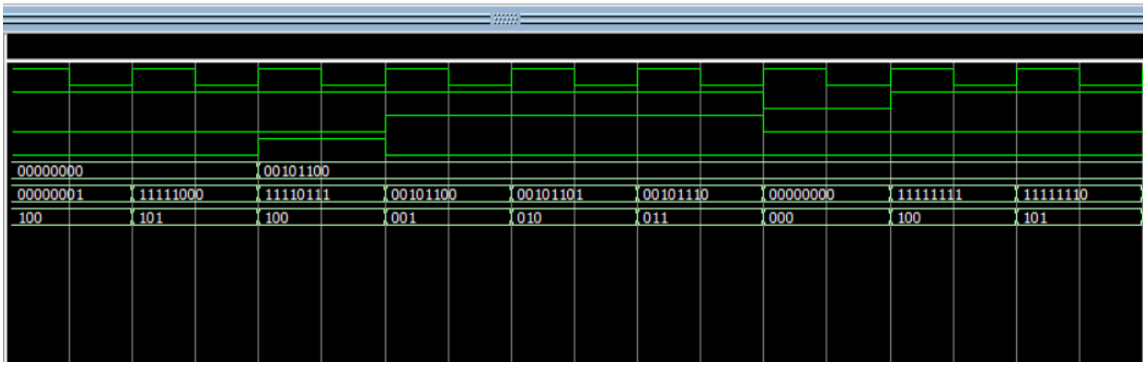
- shifter8

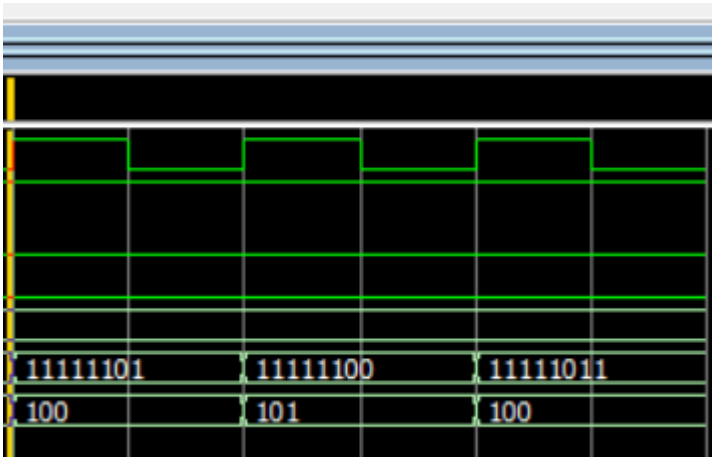


각각의 operation이 제대로 동작함을 확인했다.

- cntr8

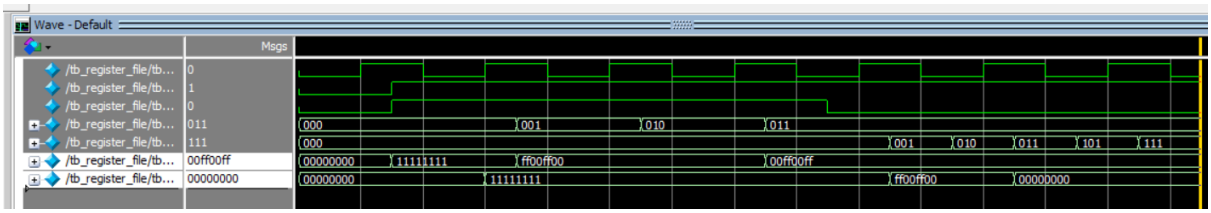






각각의 operation이 제대로 실행됨을 확인하였다.

- Register file

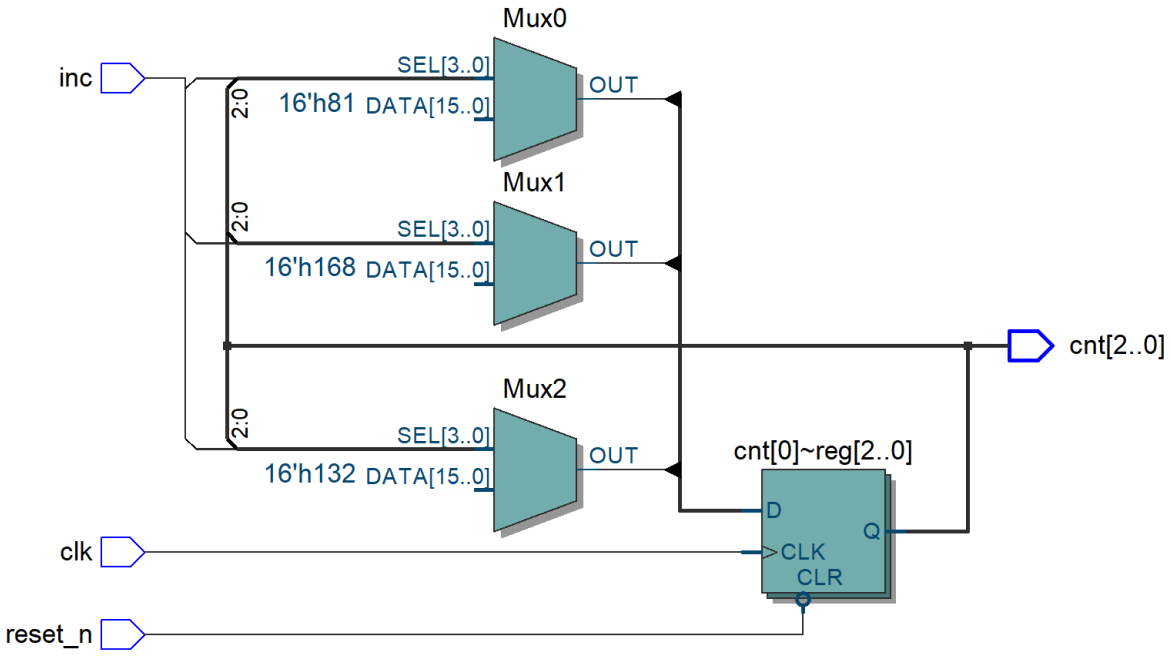


각각의 register에 값을 쓰고 출력하는 것이 정상적으로 나오는 것을 확인하였다.

B. 합성(synthesis) 결과

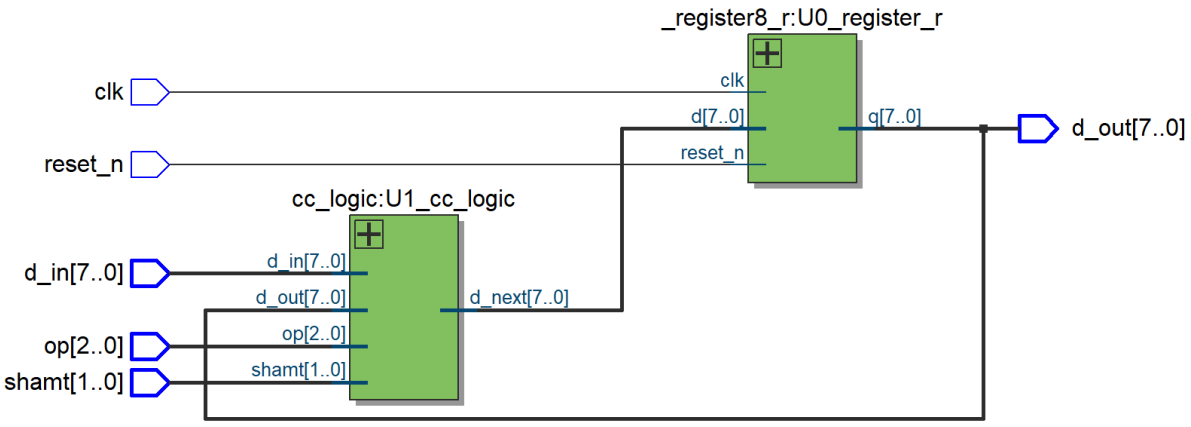
- RTL viewer

- cnt5



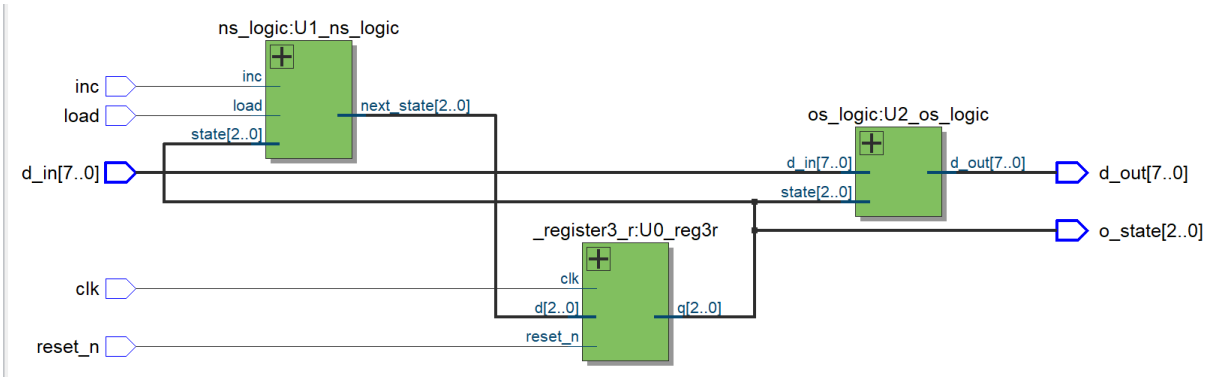
Parameter로 0~4까지 지정한 후 조건문을 통해 분기가 넘어간다.

- shifter8



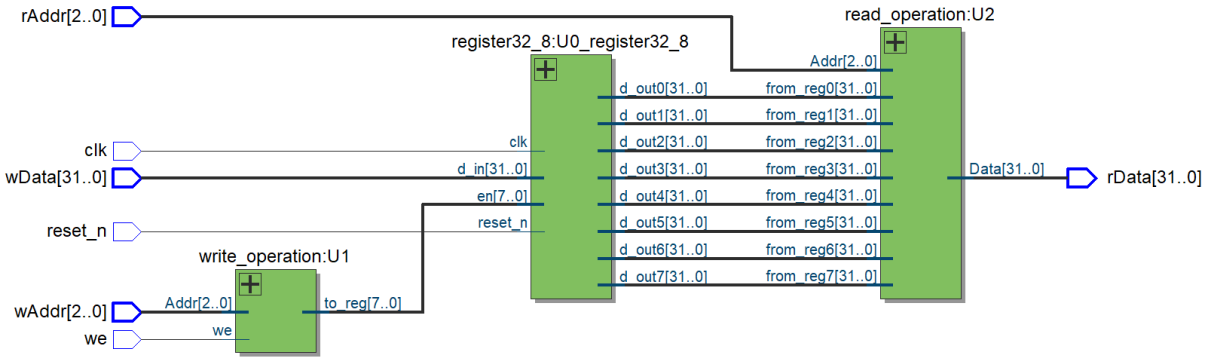
Combinational circuit C + _register8

- cntr8



Next state Logic + 3bit register + Output Logic

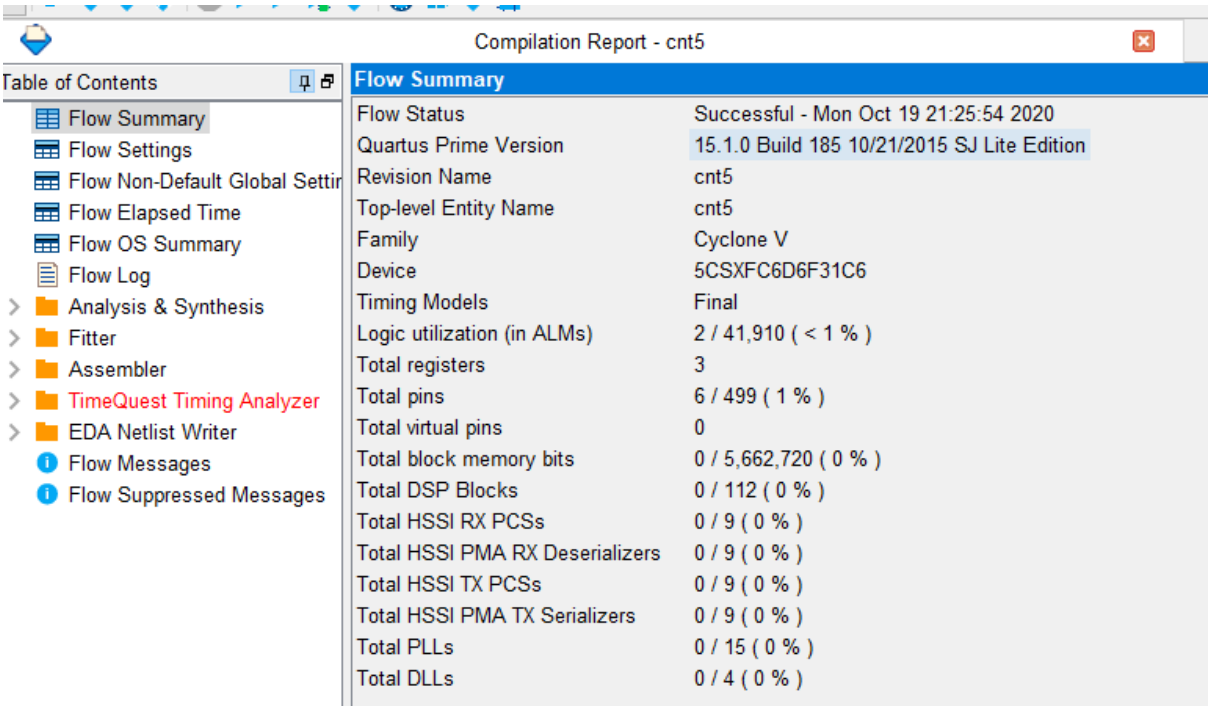
- Register file



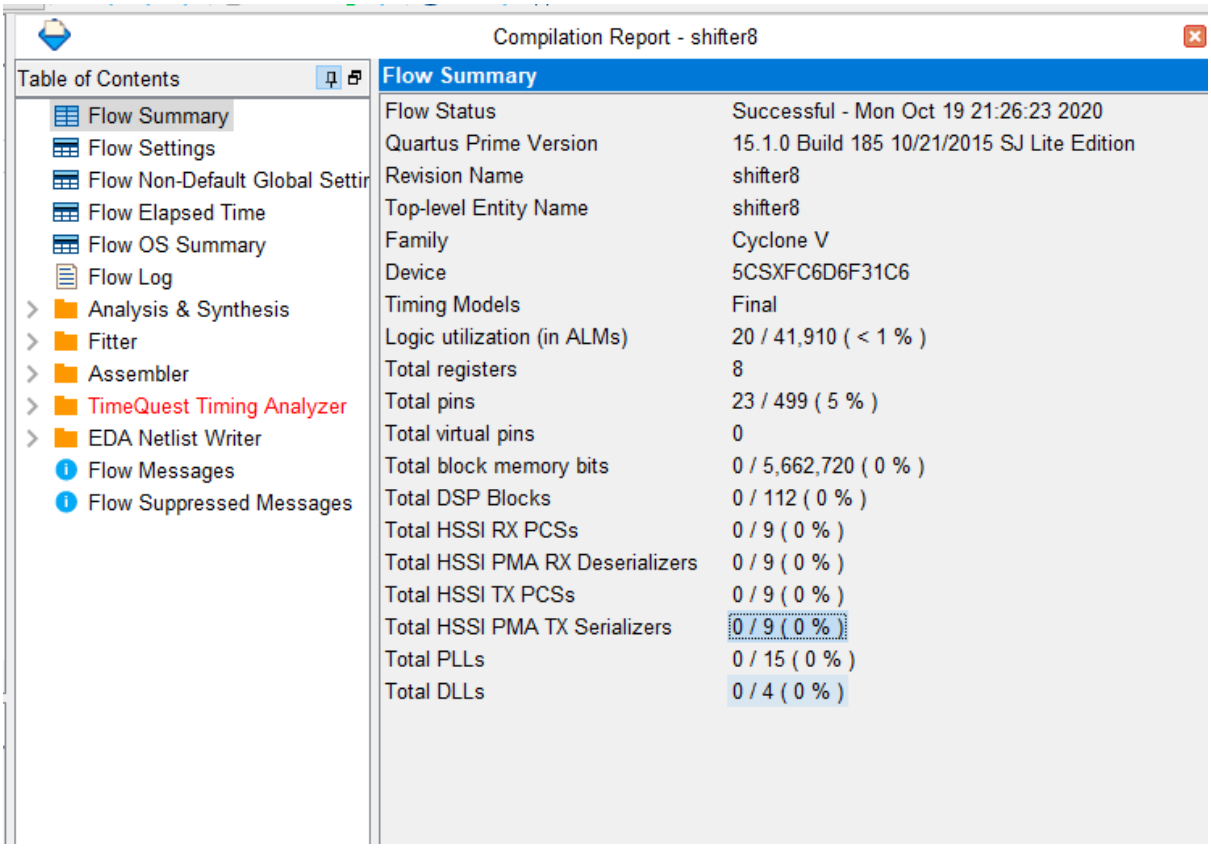
Write operation + 32*8bit register + read operation

- Flow Summary

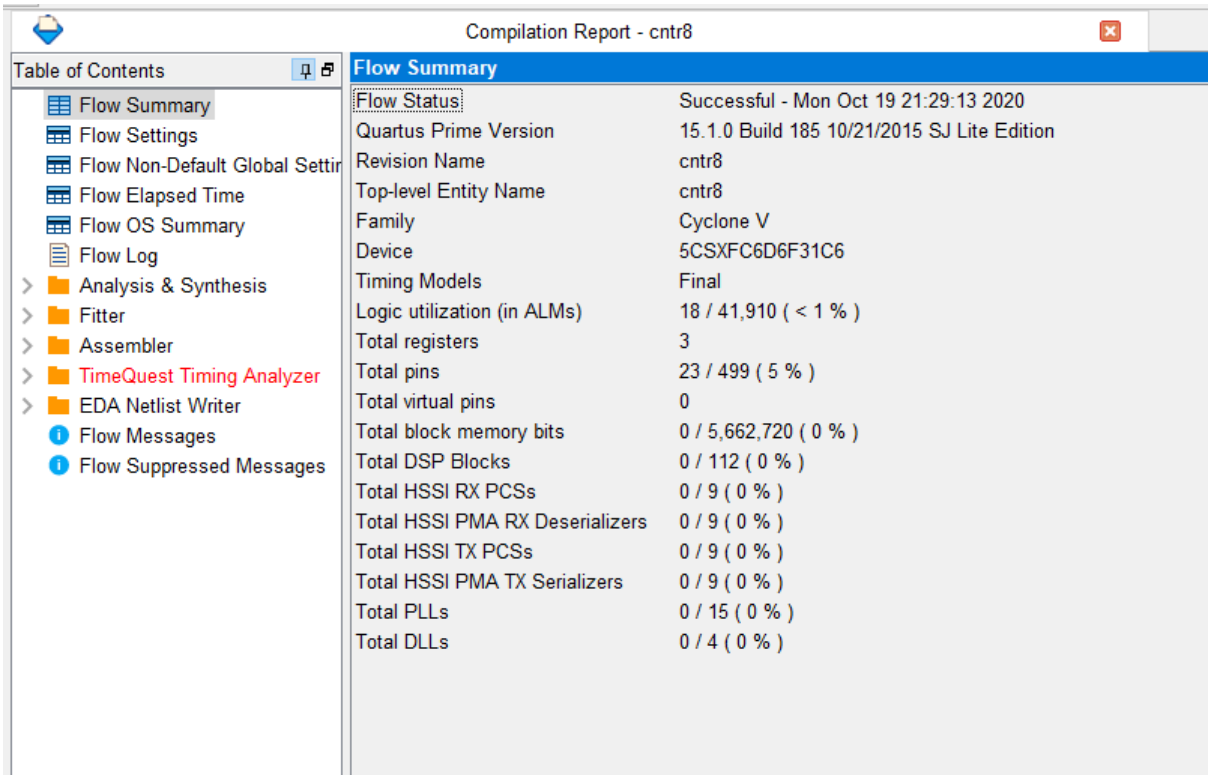
  - cnt5



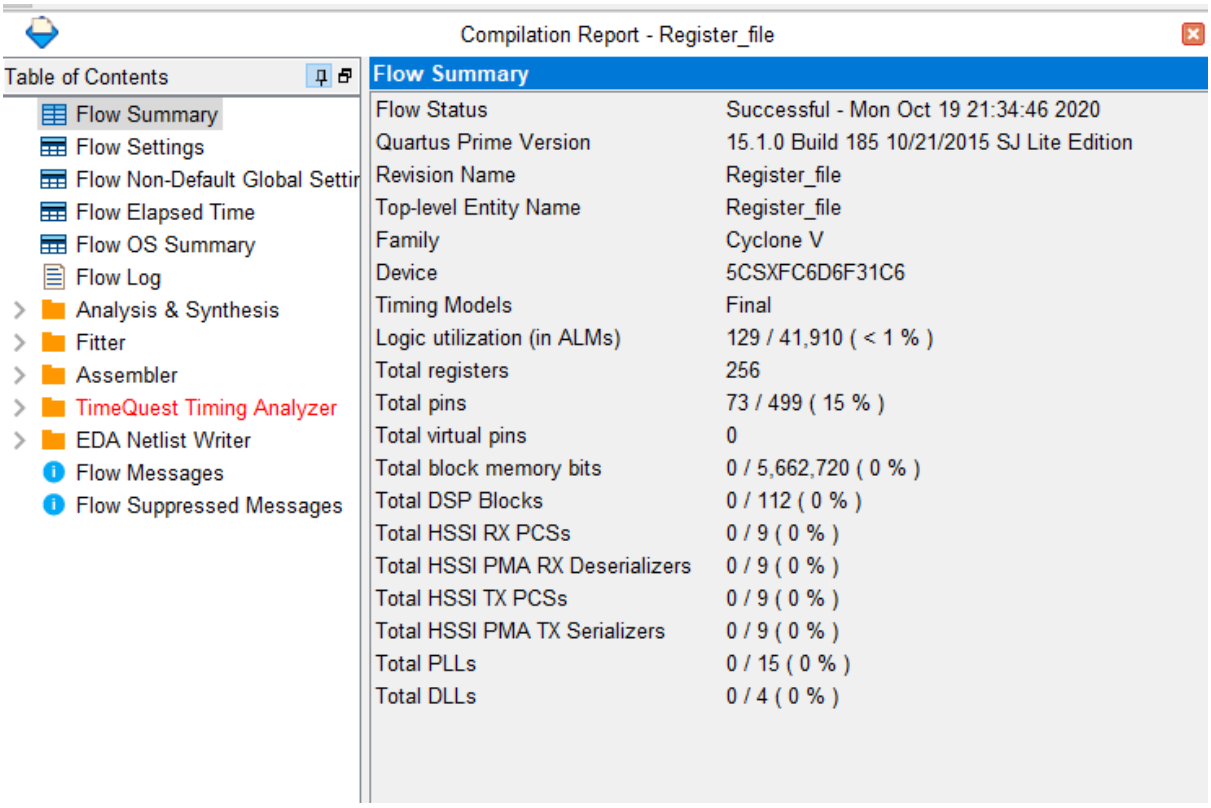
  - shifter8



- cntr8



- Register file



5. 고찰 및 결론

A. 고찰

- loadable counter와 ring counter의 장단점 및 응용분야

Ring counter의 경우 2진 카운터 또는 유사한 2개의 안정한 요소를 고리로 연결시켜 놓은 것을 의미한다. 주어진 시점에서 링 속의 하나의 요소만이 지정되어 있다고 했을 때, 입력에 값을 넣으면 그 횟수만큼 ring 안을 돈다고 생각하면 된다.

Loadable counter는 우리가 데이터를 고르는 과정이 필요할 때 사용하는 Multiplexer가 없어도 사용할 수 있다는 장점을 가진 회로이다. 그와 동시에 데이터의 증감을 위한 조합회로를 최적화가 가능해서, 회로의 면적을 줄일 수 있다는 장점을 가지고 있다. 이 회로는 많은 기능을 하게 되는 현 시대에서 회로가 복잡해지고 면적이 넓어지는 것을 막을 수 있는 회로일 것이다.

- barrel shifter란 무엇이며, register를 n-bit만큼 shift시키고자 할 때 필요한 multiplexer와 bandwidth와 수에 대하여

한 개의 연산자로 데이터 워드 내에 있는 다수의 비트를 이동시키거나 회전시킬 수 있는 하드웨어 장치이다.

Register가 n-bit를 가지고 있고 n-bit만큼 shift시킬 때 필요한 multiplexer의 bandwidth의 수는: 8-bit shifter -> 8 4-to-1 multiplexers, n-bit shifter -> $n \log⟦(2n)⟧$

B. 결론

이번 과제는 카운터의 동작원리에 관해서 익힐 수 있는 과제였다. If, else if, else문과 switch문을 쓰면서, 추가적인 문법을 익힐 수 있었다. 디지털 논리회로 1 시간에 배운 FSM Moore를 다시 한번 복습하게 되었으며, 많이 어려왔던 과제였다.

6. 참고문헌

이준환/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2020
공영호/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2020