컴퓨터 공학 기초 실험2 보고서

실험제목: Shifter & Counter, Register File

실험일자: 2023년 10월 16일 (월)

제출일자: 2023년 10월 20일 (금)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 월요일 0, 1, 2

학 번: 2022202065

성 명: 박나림

1. 제목 및 목적

A. 제목

Shifter & Counter, Register File

B. 목적

Bit를 shift 할 수 있는 연산 종류 모듈들을 설계하여 LSL, LSR, ASR을 만들 수 있도록 한다. 8-bit loadable up/down counter을 이해하고 FSM design 과정에 맞추어서 설계하도록 한다. Register File에 대해 공부하고 각 register마다 값을 저장하여 확인할 수 있도록 한다.

2. 원리(배경지식)

A. Shifter

Register에 저장되어 있는 정보를 단방향, 또는 양방향으로 이동시킬 수 있는 하드웨어를 shifter라고 한다. 이번 실습에서는 8-bits loadable shifter로 여러 control signal를 입력으로 받는 shifter로 설계하게 된다. Shifter의 종류로 3가지의 연산 수행 기능을 가지게 되며, 연산 종류는 다음과 같다.

LSL (Logical shift left): 지정한 bit수 만큼 왼쪽으로 shift 시킨다.

LSR (Logical shift right): 지정한 bit수 만큼 오른쪽으로 shift 시킨다.

ASR (Arithmetic shift right): 지정한 bit수 만큼 오른쪽으로 shift 시키면서 MSB이니 부호 비트는 유지한 채로 바꾼다.

B. Counter

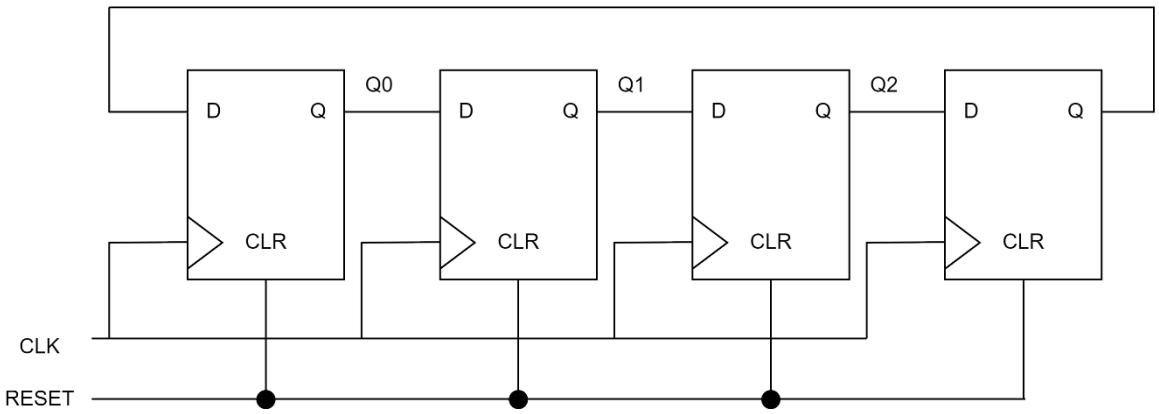
2개 이상의 flip-flop으로 구성되며, 펄스 신호에 따라 상태들이 차례대로 변이되는 register를 counter라고 한다. 해당 상태가 변이되는 횟수를 세는 데에 사용된다.

-Moore FSM & Mealy FSM의 장단점

Moore machin은 출력이 현재 상태에 의해서만 결정되는 것을 말하며, Mealy machine은 출력이 현재 상태와 입력 모두에 의해서 결정되는 것을 말한다. 이러한 Mealy machine은 Glitch 문제가 발생할 수 있으나 Moore machine은 안정적인 편이라 현재 대부분 시스템에서 주로 쓰이는 편이다. 또한 Moore FSM은 state 설계가 현재 상태에 의해서만 결정되기 때문에 이해도가 쉬운 편이라는 장점도 있다. Mealy FSM은 그보다는 어렵다는 단점

이 있지만 Moore FSM보다 더 적은 수의 state를 가진다는 점에서 장점이 있다. 따라서 Moore FSM을 기준으로 하여 Mealy FSM을 구성하면 서로의 장단점을 보완하여 최적화 시키기에 유용하다.

-ring count



Ring counter는 shift register에 연결된 flip-flop으로 구성된 counter의 종류 중 하나로써 마지막 flip-flop의 출력이 다시 첫번째 flip-flop의 입력으로 들어가는 원형 형태의 구조이다. 입력된 데이터는 clock의 펄스 신호마다 한 칸 씩 이동하게 되는 형식이다. 이러한 ring counter는 하드웨어 설계에서 FSM을 설계할 때 주로 사용된다.

C. Register File

Register file(RF)는 여러 개의 register을 붙여서 각각의 register에 값을 저장하고 이후 필요에 따라 해당 register의 주소를 불러와서 저장되어 있는 값을 읽을 수 있게 하는 것을 말한다. 이번 실습에서는 Write logic과 8개의 32-bits register, Read logic으로 구성하여 설계한다. 쓰기 모드는 write enable(we)로 활성화시키며, Decoder를 통해 주소를 해석하여 각각의 register를 enable시킨다. 읽기 모드는 MUX를 통해 8개의 register중 해당하는 register를 한 개 선택하여 저장되어 있는 값을 불러온다.

-Stack & Queue

Stack은 데이터를 쌓아 올리다는 의미로, 순서대로 위로 쌓이는 식으로 하여 가장 마지막에 삽입된 자료가 가장 먼저 삭제되는 후입 선출 구조(LIFO: Last In Firsr Out)를 가진다. 정해진 방향으로만 쌓을 수 있으며 top 포인터를 통해서만 접근이 가능하다. 새로 삽입

(Push)되는 자료는 top이 가리키는 가장 맨 위에 쌓이게 되며, 삭제 시(Pop)에도 top이 가리키는 곳을 삭제하게 된다. 이러한 스택은 주로 웹 방문 기록, 실행 취소, 후위표기법 등에서 쓰인다.

큐는 스택과 반대로 먼저 들어온 데이터가 먼저 나가는 선입 선출의 구조(FIFO: First In First Out)를 가진다. 첫번째 데이터를 front 포인터가, 마지막 데이터를 rear 포인터가 가리키고 있어서 삭제 연산(Dequeue)은 front를 통해 수행되고 삽입 연산(Enqueue)은 rear를 통해 수행된다. 이러한 큐는 대기열 순서, CPU 연산 처리시의 작업 대기, 버퍼링 시 등에 사용된다.

3. 설계 세부사항

A. Shifter

-I/O

Shifter8	Input	clk	1-bit, clock
		reset_n	1-bit, Active low
		op	3-bit, Operation
		shamt	2-bit, Shift amount
		d_in	8-bit, data in
	Output	d_out	8-bit, data out

LSL8	Input	d_in	8-bit, data in
LSR8		shamt	2-bit, shift amount
ASR8	Output	output	8-bit, data out

mx4	Input	d0	1-bit, data 1
		d1	1-bit, data 2
		d2	1-bit, data 3
		d3	1-bit, data 4
		s	1-bit, 선택 신호
	output	y	1-bit, 결과 값

B. Counter

-I/O

		clk	1-bit, clock
		reset_n	1-bit, Active low

Cntr8	Input	load	1-bit, cont값에 할당
		inc	1-bit, 증감 신호
		d_in	8-bit, data in
	output	d_out	8-bit, data out
		o_state	3-bit, 검증용

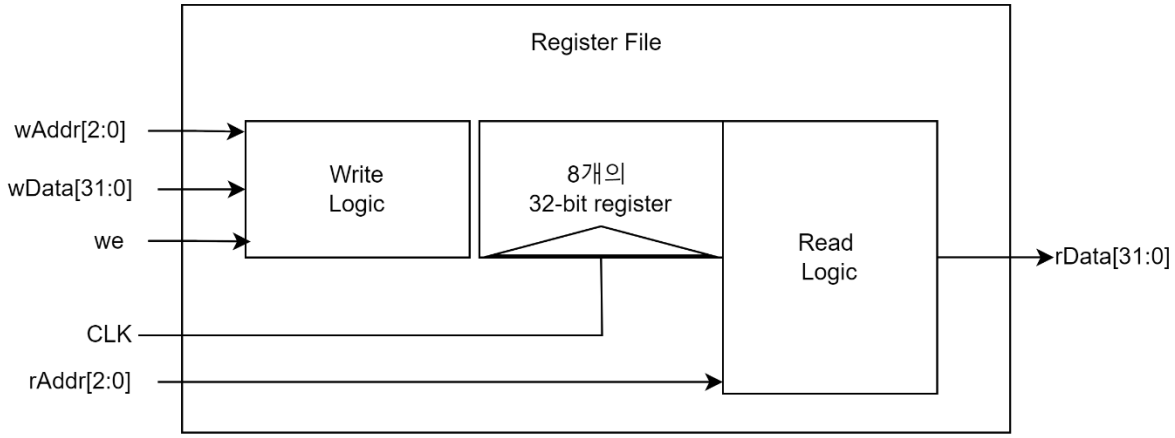
Cla8	Input	a	8-bit, data A
		b	8-bit, data B
		ci	8-bit, carry in
	Output	s	8-bit, sum
		co	1-bit, carry out

C. Register File

-I/O

Register File	Input	clk	1-bit, clock
		reset_n	1-bit, Active low
		we	1-bit, write enable
		wAddr	3-bit, write address
		rAddr	3-bit, read address
		wData	32-bit, write data
	Output	rData	32-bit, read data

-회로도

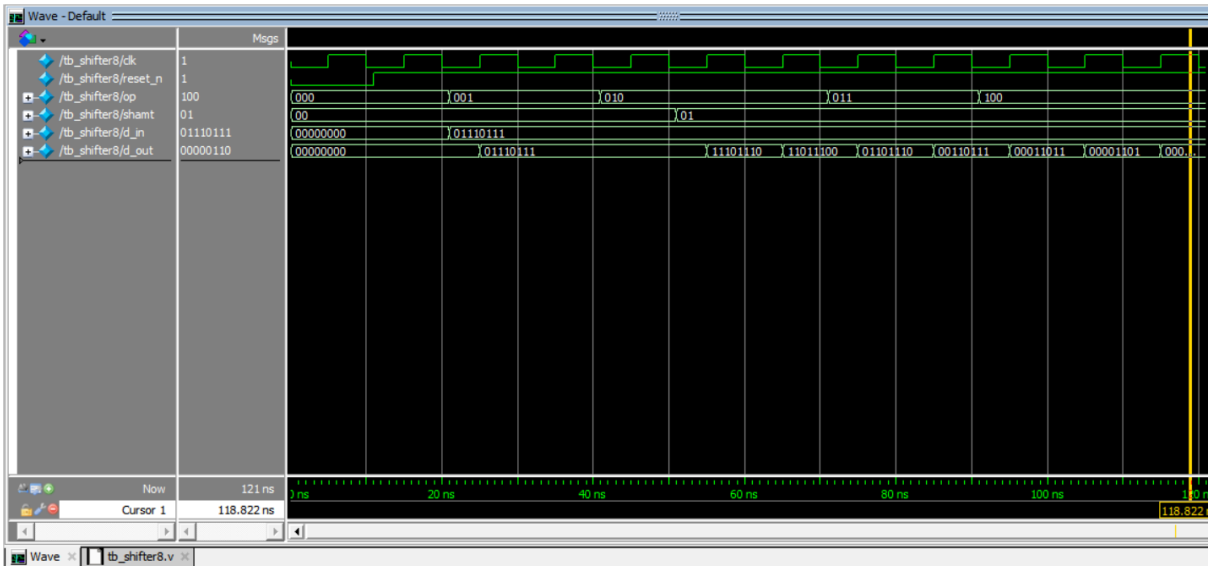


4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

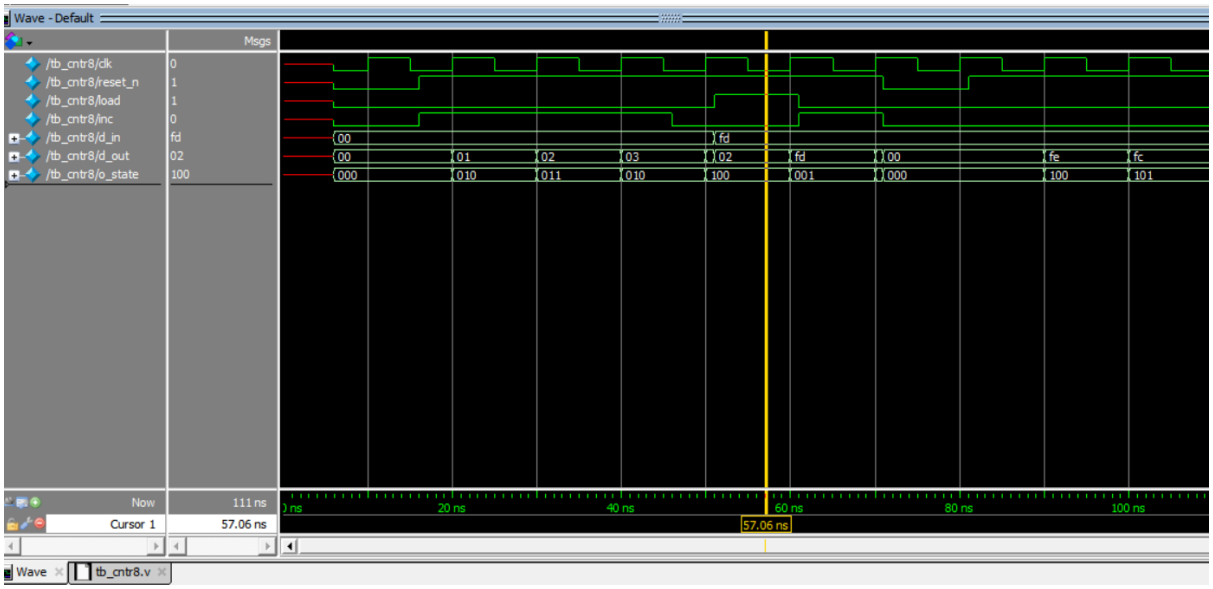
1) Shifter8

NOP, LOAD, LSL, LSR, ASR 명령어들을 각각 순서대로 테스트한 결과이다. 처음에 reset이 0으로 활성화되었다가 1로 비활성화된 뒤에 여전히 NOP(000)이면 결과 값도 8'b0이 된다. 그 다음 LOAD(001)을 통해 register에 8'b01110111을 저장하고 그 값을 출력한다. LSL(010)이 들어오고 shamt가 01로 1칸이 되면 왼쪽으로 한 칸 shift한 값이 된다. 빈 공간은 0으로 채워진다. 또한 다음 명령어가 들어오기 전까지는 clock이 상승하는 순간마다 업데이트 되어 계속 같은 명령을 이어서 수행하게 된다. LSR(011)이 들어오면 그 상태에서 오른쪽으로 한 칸 움직이고, ASR(100)이 되면 최상위 부호 비트를 유지한 채 오른쪽으로 한 칸 움직이게 된다. 이는 모두 명령이 먼저 들어오고 나서 그 다음 clock 상승 순간에 업데이트 되는 방식이다.



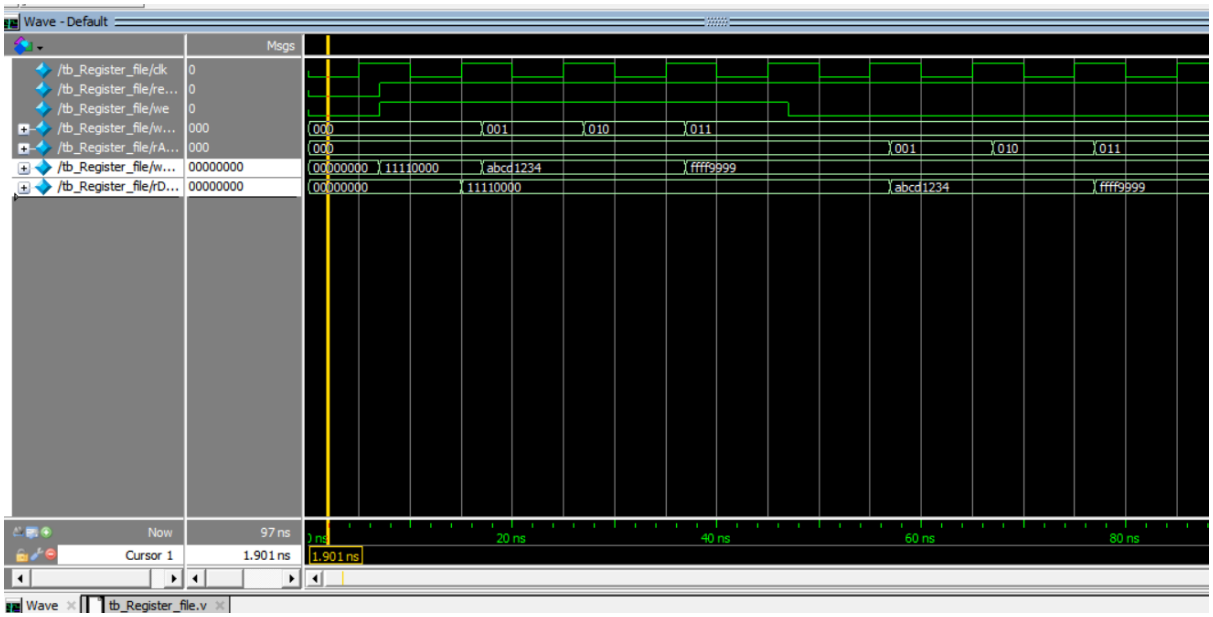
2) Counter8

처음에 reset이 0일 때는 다른 값들도 0으로 초기화 되었다가, 비활성화 되면서 증가 state부터 시작하게 된다. 첫번째로 증가하면 010, 두번째로 증가하면 011, 세번째로 증가하면 다시 010이 되며 이를 반복한다. 증가하는 동안 처음 data_in이 00이었으니 1씩 증가하여 03까지 간다. 또한 inc 신호가 켜진다. 그다음 신호가 꺼지면서 감소하게 되면 100 state로 가고 값도 02가 된다. 그리고 load 신호가 켜지면서 새로운 data_in으로 fd가 들어오게 되고, state도 load인 001이 된다. 이후 reset이 활성화되면 다시 0값들로 초기화 되고, 비활성화 되면 다시 값을 불러오며 감소시킨다.



3) Register File

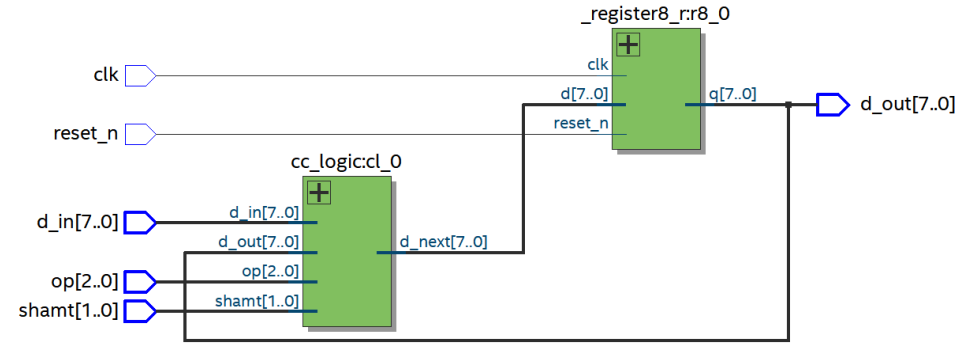
처음 reset이 0으로 활성화되었다가 1로 비활성화 되면서 we가 1로 쓰기 모드가 되면, wAddr 주소 000에 11110000을 저장한다. 이때 읽는 용인 rData도 reset일 때는 0값을 읽었다가 값을 저장하면 해당 값을 읽어온다. 그리고 001에는 abcd1234를 저장하고, 010에도 같은 값을 저장한다. 011에는 ffff9999를 저장한다. 이후 we를 0으로 비활성화 시켜서 쓰기 모드를 끝낸다. 그 다음 rAddr의 주소를 바꾸면서 해당 주소의 값을 차례대로 불러와서 확인한 모습이다.

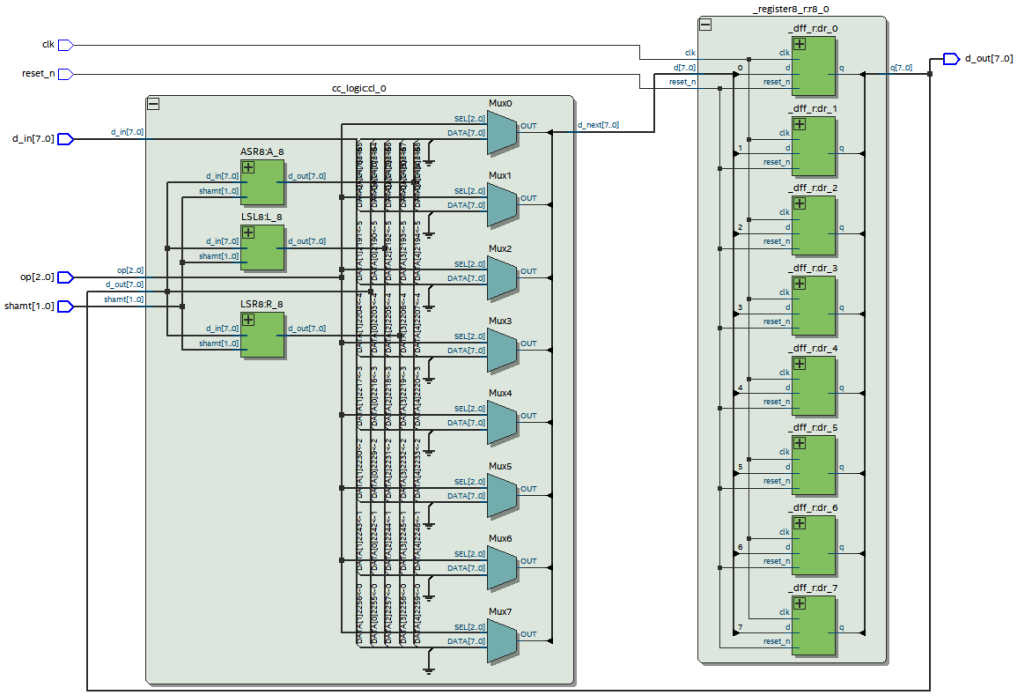


B. 합성(synthesis) 결과

1) Shifter8

Shifter8의 RTL viewer와 Flow summary이다. 회로의 모습과 크기를 확인할 수 있다.





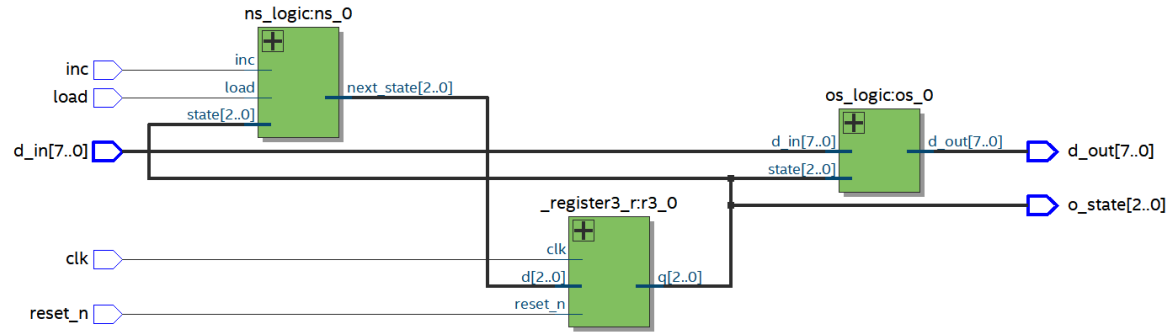
Flow Summary

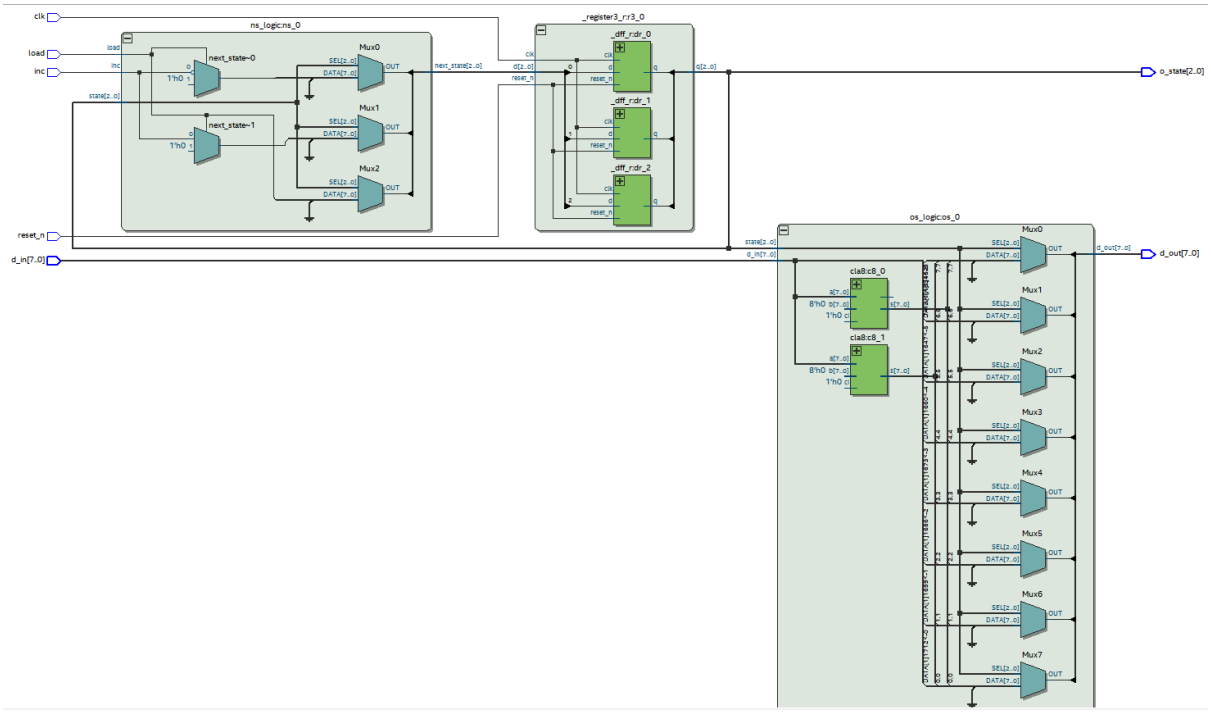
<<Filter>>

Flow Status	Successful - Mon Oct 16 13:43:01 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)	24 / 41,910 (< 1 %)
Total registers	8
Total pins	23 / 499 (5 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

2) Counter8

Counter8의 RTL viewer와 Flow summary이다. 회로의 모습과 크기를 확인할 수 있다.

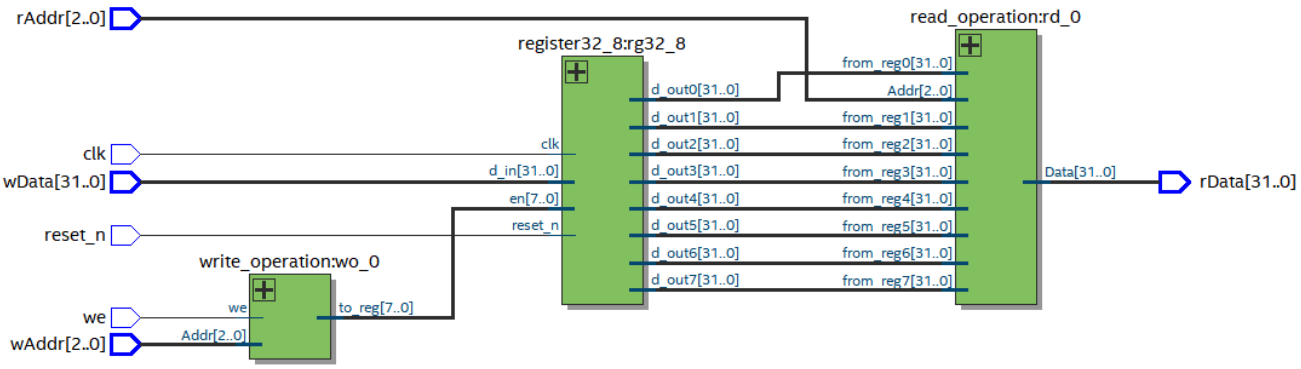




Flow Summary	
<<Filter>>	
Flow Status	Successful - Mon Oct 16 20:40:16 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cntr8
Top-level Entity Name	cntr8
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	25 / 41,910 (< 1 %)
Total registers	3
Total pins	23 / 499 (5 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

3) Register File

Register File의 RTL viewer와 Flow summary이다. 회로의 모습과 크기를 확인할 수 있다.



Flow Summary

<<Filter>>

Flow Status	Successful - Wed Oct 18 12:08:47 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Register_file
Top-level Entity Name	Register_file
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	123 / 41,910 (< 1 %)
Total registers	256
Total pins	73 / 499 (15 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

5. 고찰 및 결론

A. 고찰

Shifter를 구현하고 실행해봤을 때 값이 계속 이어서 shift되지 않고 연산 종류 별로 따로 따로 계산되는 상황이 발생하였다. 다시 회로도를 확인해보니, next state값이 다시 input으로 들어가야 되는데 그 부분이 빠진 것 같아 top module에서 instance 할 때 제대로 값을 써주었다. 그 결과 정상적으로 값이 나올 수 있었다.

Counter을 구현하는 과정에서, 증가하는 과정은 성공하였으나 감소하는 과정에서 값이 제대로 나오지 않았다. Cla로 변수들을 보내고 계산하는 과정에 문제가 있는 것 같아서 다시 문법을 살펴보고 다른 값들로 변경을 해봤지만 1씩 감소되지 않고 4씩 감소되거나 아예 다르게 나오는 등 계속 문제가 발생하였다. Inc와 dec의 wire에 대한 이해가 아직 부족한 것 같아 더 공부를 해봐야 될 것 같다.

B. 결론

Shifter와 Counter를 설계하면서 Mealy machine에 대한 이해를 더 잘 할 수 있었다. 출력과 입력을 둘 다 신경 써야 하니 Moore FSM보다는 더 복잡했지만 State수를 줄이거나 좀 더 자세히 설계하는 데엔 효과적인 것 같다는 생각이 들었다. 또한 Register File을 설계하면서, 이러한 Register File을 통해 다른 FSM들을 설계하는 데에 이용할 수 있을 것 같다. Register에 각각 값들을 저장할 수 있으니, 다른 프로그램에서 여러 값들을 저장할 때 사용할 수 있기 때문이다.

-loadable counter와 ring counter의 장단점 및 응용분야

Loadable counter는 사용자가 설정한 값들을 원하는 때에 load할 수 있으니 counter를 이용하는 다른 여러 프로그램을 설계할 때 더 편할 것이다. 초기 값도 load하여 시작 값을 설정하고, 종료 값도 설정할 수 있어 범위도 설정하는 데에 도움이 될 것이다. 하지만 그만큼 설계하는 게 복잡하여 회로의 크기도 커지며, 비용도 증가한다는 단점이 있다.

이러한 loadable counter는 여러 디지털 회로에서 타이밍과 순서 제어에 사용될 수 있으며, 데이터의 도착 및 전송 타이밍을 조절하는 통신 시스템에 응용될 수 있을 것이다.

Ring counter는 다른 카운터에 비해 간단한 구조를 가지기 때문에 회로의 크기도 작고 비용도 절감할 수 있다는 장점이 있다. 또한 ring 형태의 구조이기 때문에 일정한 주기를 가지고 반복할 수 있는 순환 기능을 활용할 때 유용할 것이다. 하지만 특정 구조 때문에 한정된 응용분야에서만 쓰일 수 있다는 단점이 있다. 특정 순서로만 설정이 가능하기 때문에, 사용자가 임의로 순서를 정하는 데엔 어려움이 있을 수 있다. 이러한 Ring counter는 간단하고 순환적인 특성을 필요로 하는 분야에서 활용될 것이며, 주로 주기적인 제어가 필요할 때, 디지털 시계처럼 동기 신호를 생성하여 반복해야 될 때, 다중 주파수 선택

과 같은 분야에서 활용될 수 있을 것이다.

-barrel shifter

Barrel shifter는 입력 데이터의 bit수에 따라 여러 bit를 좌우로 이동시키는 데에 사용되는 논리회로이다. 한번의 연산만으로 데이터의 여러 bit를 이동할 수 있으며, 이러한 이동은 한번에 병렬 처리하여 이동되기 때문에 빠른 데이터 이동을 할 수 있다. 주로 ALU의 연산과정에서 사용하게 된다. Barrel shifter에서 n-bit register를 n-bit 만큼 shift시킬 때, 각 bit위치를 제어할 수 있는 멀티플렉서가 필요하다. 따라서 전체 멀티플렉서의 수는 n이 될 것이다. 또한 Bandwidth인 대역폭은 한 bit를 이동시키는 데에 필요한 데이터를 고려하게 되므로, n-bit만큼 이동하기 위해서는 마찬가지로 n의 대역폭이 필요할 것이다.

6. 참고문헌

David Money Harris and Sarah L. Harris / Digital Design and Computer Architecture / Elsevier / 2007