컴퓨터 공학 기초 실험2 보고서

실험제목: Shifter & Counter

실험일자: 2019년 09월 24일 (화)

제출일자: 2019년 09월 30일 (월)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 화요일 0, 1, 2

학 번: 2015722031

성 명: 박 태 성

1. 제목 및 목적

A. 제목

Shifter & Counter

B. 목적

Register의 정보를 단방향, 양방향으로 이동하는 shifter와 클럭에 따라 순차적으로 상태 전이를 하는 counter를 구현해보면서 FSM 응용 능력을 증진시키고 sequential circuit과 flip-flop에 대해 학습 해 보는 것을 목적으로 둔다.

2. 원리(배경지식)

A. Moore FSM과 Mealy FSM

Moore FSM은 현재 상태를 반영하여 출력을 결정하는 순차 회로이다. 현재 상태만 고려하면 되기 때문에 비교적 이해가 쉽다. 그러나, state의 수가 비교적 많아 회로가 커지면 속도 저하 가능성이 크다.

Mealy FSM은 현재 상태와 input을 반영하여 출력을 결정하는 순차 회로다. 현재 상태와 input을 모두 고려해야해서 비교적 이해가 어렵다. 그러나, state의 수가 비교적 적어서 속도가 빠를 수 있다.

따라서, 경우에 따라 두 machine을 함께 사용하는 것이 좋겠다.

B. Ring Counter

Shift register의 마지막 flip-flop의 출력을 첫 번째 flip-flop의 input으로 연결한다. Clock pulse에 따라 bit shift가 된다. 순환을 반복하는 작업에 유용하다. 그러나, 수가 일정하게 증가하지 못한다. 신호등과 같이 반복 동작을 순차적으로 수행하는 곳에 사용하면 좋겠다.

3. 설계 세부사항

A. Shifter

i. Description

Signal	Description
reset_n	Active low에서 동작하는 reset signal로 Register의 값을 0으로 초기화
op	➢ NOP : No Operation (이전 값을 그대로 출력) ➢ LOAD : 입력된 데이터를 레지스터에 저장 ➢ LSL : 논리적으로 왼쪽 Shift를 수행 ➢ LSR : 논리적으로 오른쪽 Shift를 수행 ➢ ASR : 산술적으로 오른쪽 Shift를 수행

shamt	Shift Amount로 2bit

ii. I/O configuration

Module	Port	Name	Bandwidth	Description
shifter8	Input	clk	1-bit	Clock
		reset_n	1-bit	Active low에 동작
		op	3-bits	Operation
		shamt	2-bits	Shift Amount
		d_in	8-bits	저장하기 위한 입력 값
	Output	d_out	8-bits	Register의 값 출력

Module	Port	Name	Bandwidth	Description
LSL8 **LSR8** **ASR8**	Input	d_in	8-bits	저장하기 위한 입력 값
		shamt	2-bits	Shift Amount
	Output	d_out	8-bits	Register의 값 출력
mx4	Input	d0	1-bit	Multiplexer의 1번째 입력
		d1	1-bit	Multiplexer의 2번째 입력
		d2	1-bit	Multiplexer의 3번째 입력
		d3	1-bit	Multiplexer의 4번째 입력
		s	1-bit	Multiplexer의 선택 신호
	Output	y	1-bit	Multiplexer의 결과

iii. Truth Table

Input				Output							
op			shamt	O[7]	O[6]	O[5]	O[4]	O[3]	O[2]	O[1]	O[0]
0	0	1	0	1	1	1	0	1	1	1	0
0	1	0	1	1	1	0	1	1	1	0	0
0	1	1	1	0	1	1	0	1	1	1	0
0	1	0	2	1	0	1	1	1	0	0	0
1	0	0	2	1	1	1	0	1	1	1	0

B. Counter

i. Description

Signal	Description
reset_n	Active low에서 동작하는 reset signal로 Register의 값을 0으로 초기화
load	입력된 데이터를 Register에 저장
inc	Counter의 증가와 감소를 제어하는 신호

ii. I/O configuration

Module	Port	Name	Bandwidth	Description
cntr8	Input	clk	1-bit	Clock
		reset_n	1-bit	Active low에 동작
		load	1-bit	Register에 입력 값 저장
		inc	1-bit	Count 값을 증가/감소
		d_in	8-bits	저장하기 위한 입력 값
	Output	d_out	8-bits	Register의 값 출력
		o_state	3-bits	현재 상태 값을 출력

Module	Port	Name	Bandwidth	Description
cla8	Input	a	8-bits	CLA의 입력 A
		b	8-bits	CLA의 입력 B
		ci	8-bits	Carry in
	Output	s	8-bits	CLA의 출력
		co	1-bit	Carry out

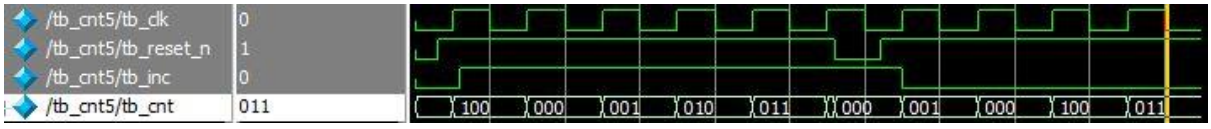
iii. Truth Table

Input			**Output**							
load	inc	reset	O[7]	O[6]	O[5]	O[4]	O[3]	O[2]	O[1]	O[0]
X	X	0	0	0	0	0	0	0	0	0
1	X	1	0	0	0	0	1	1	0	0
0	1	1	0	0	0	0	1	1	0	1
0	1	1	0	0	0	0	1	1	1	0
0	1	1	0	0	0	0	1	1	1	1
0	0	1	0	0	0	0	1	1	1	0
0	0	1	0	0	0	0	1	1	0	1
0	0	1	0	0	0	0	1	1	0	0

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

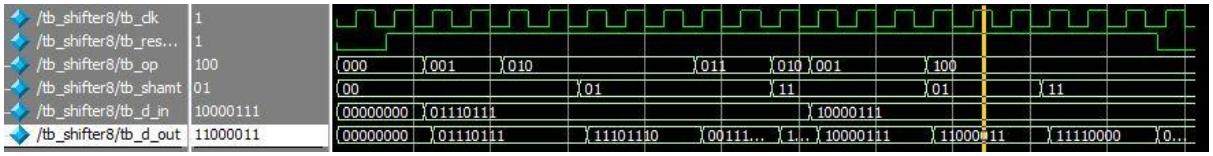
i. Cnt5



Clk이 rising edge일 때, inc가 0이면 이전 state이 출력되고 inc가 1이면 다음 state이

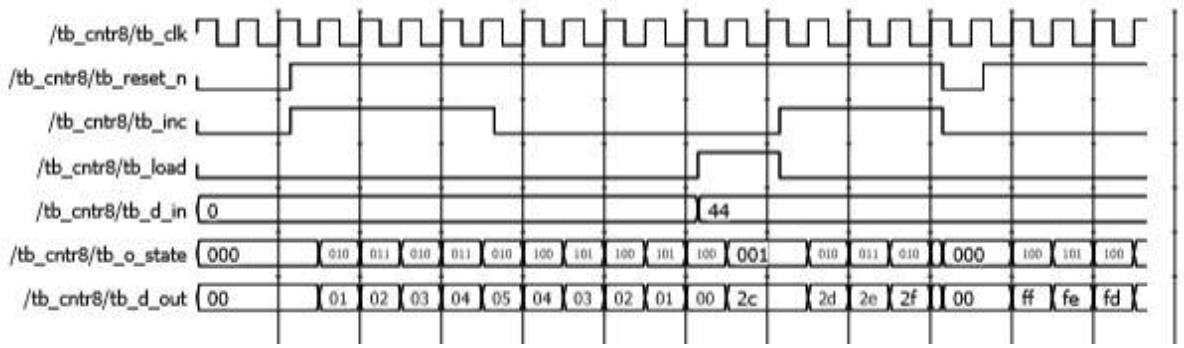
출력되는 것을 확인하였다. 처음에 state 0로 testbench를 작성했는데 tb_inc가 0이었기 때문에 이전 state인 100이 출력된다. 여기서 다음 state, 이전 state으로 간다는 것은 수가 1씩 증가하거나 감소하는 것을 각각 의미한다. Active low이기 때문에 reset_n에 0을 주면 state 0으로 상태가 바뀐다.

ii. Shifter8



tb_op가 000일 때는 no operation(NOP)이므로 shift가 발생하지 않는다. 현재 register의 값을 출력한다. Tb_op가 001이면 입력된 데이터를 출력한다. Clock이 rising edge일 때, 입력된 데이터 tb_d_in이 출력된다. Tb_op가 010이면 tb_shamt에 따라서 logical shift(left)가 발생한다. Tb_op가 011이면 tb_shamt에 따라 logical shift(right)가 된다. Tb_op가 100이면 tb_shamt에 따라 arithmetic shift(right)이 된다. 이때는 MSB로 빈자리가 채워진다. Active low이기 때문에 reset_n에 0을 주면 0으로 상태가 바뀐다.

iii. Cntr8

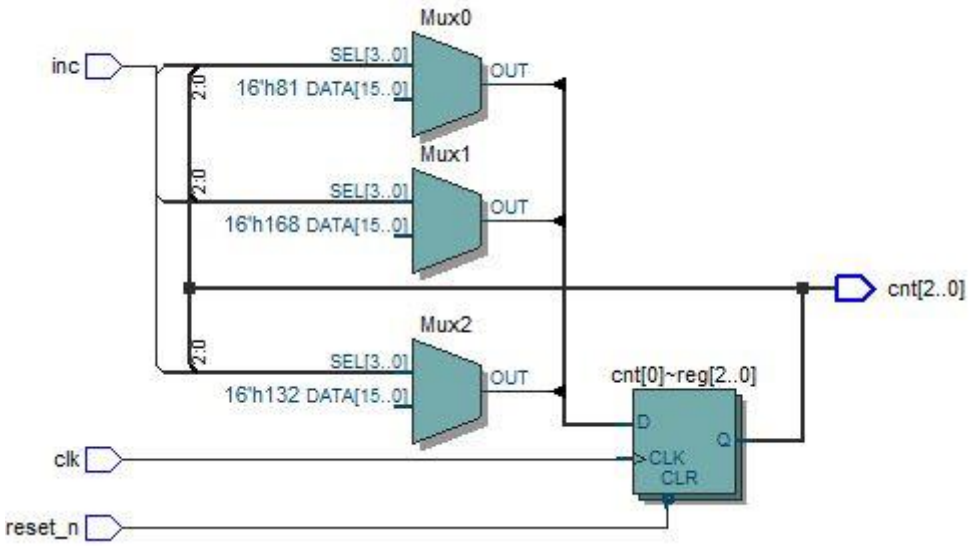


tb_o_state이 010 또는 011이면 tb_d_in을 increament한다. Increment는 CLA가 담당한다. Tb_o_state이 101 또는 100이면 tb_d_in을 decreament한다. Decreament는 CLA가 담당한다. Tb_o_state이 001 이면 tb_d_in 값을 load한다. Load가 1이면 입력으로 들어와 있는 data 를 count 값에 할당한다. 이후에 state에 따라 값이 증가하고 감소하는 것을 확인했다. Active low이기 때문에 reset_n에 0을 주면 0으로 바뀐다.

B. 합성(synthesis) 결과

i. Cnt5

1. RTL Viewer

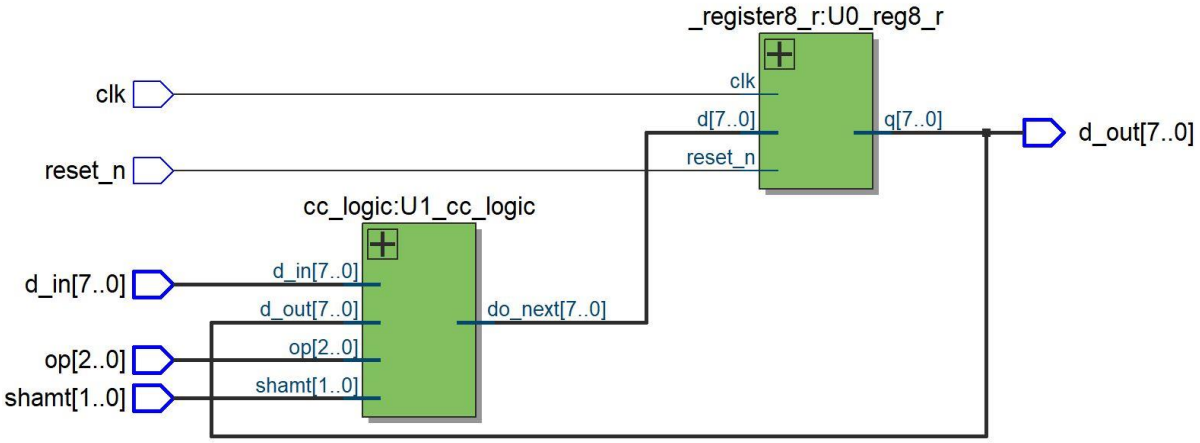


2. Flow Summary

Flow Summary	
Flow Status	Successful - Mon Sep 23 23:57:17 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	cnt5
Top-level Entity Name	cnt5
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	3
Total pins	6
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

ii. Shifter8

1. RTL Viewer

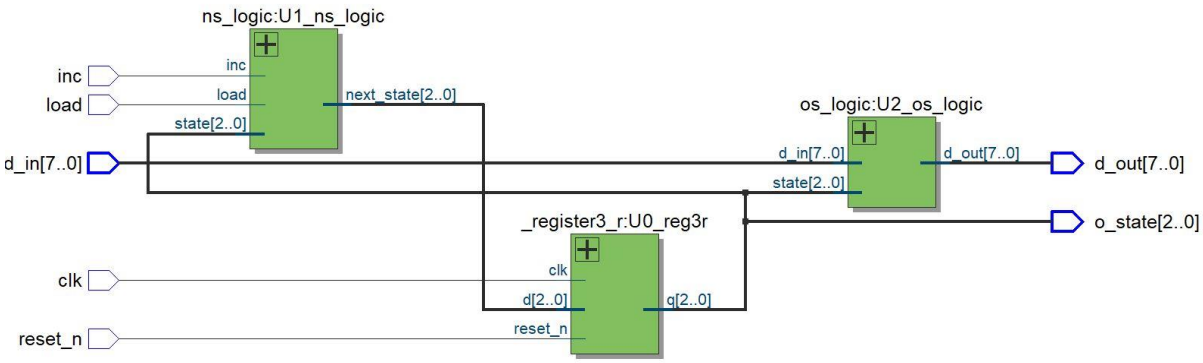


2. Flow Summary

Flow Summary	
Flow Status	Successful - Tue Sep 24 01:33:18 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 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 until Partition Merge
Total registers	N/A until Partition Merge
Total pins	N/A until Partition Merge
Total virtual pins	N/A until Partition Merge
Total block memory bits	N/A until Partition Merge
Total PLLs	N/A until Partition Merge
Total DLLs	N/A until Partition Merge

iii. Cntr8

1. RTL Viewer



2. Flow Summary

Flow Summary	
Flow Status	Successful - Tue Sep 24 02:24:18 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 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
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

5. 고찰 및 결론

A. 고찰

예상했던 waveform이 나오지 않아 model-sim에서 waveform에 하위 모듈을 추가하여 오류 발생 지점을 추적했다. Dot operator를 사용하지 않고 기억에 의존하여 port들을 연결하다가 실수를 했다. 에러를 찾는데 waveform은 유용한 tool이다. 편의성 문제로 Dot operator를 회사에서는 많이 쓰지 않는다고 들었다. 그러나, 이번 실습을 통해서 나는 dot operator를 적절히 사용하면 오류 발생률을 줄일 수 있을 것이라 생각한다.

B. 결론

i. Loadable counter & Ring counter

Loadable counter는 사전에 초기값 설정을 할 수 있다. Loading이 사전에 초기값을 설정하는 의미이다. 경우에 따라 속도를 빠르게 해 줄 수 있다. 그러나, 초기값을 저장하는 레지스터를 추가 마련해야해서 크기가 증가한다. 설정한 값부터 수를 세기 시작해야하는 시계 같은 곳에 사용하면 좋겠다.

Ring Counter는 shift register의 마지막 flip-flop의 출력을 첫 번째 flip-flop의 input으로 연결한다. Clock pulse에 따라 bit shift가 된다. 순환을 반복하는 작업에 유용하다. 그러나, 수가 일정하게 증가하지 못한다. 신호등과 같이 반복 동작을 순차적으로 수행하는 곳에 사용하면 좋겠다.

ii. Barrel shifter

Barrel shifter는 sequential circuit이 아니라 combinational circuit으로 구성한다. 한 번의 연산으로 전체 data를 shift한다. 따라서, clock의 rising edge마다 shift하는 일반적인 shifter보다 속도가 빠르다. Barrel shifter는 Multiplexer(mux)의 수가 사용할 수 있는 bit 수를 결정한다. Mux의 수는 $n * \log_2 N$으로 구한다. 예로, 32bits barrel shifter를 구현하려면 mux는 160개가 필요하다. Barrel shifter는 mux의 bandwidth에 따라 shift 할 수 있는 범위가 결정된다. Nbit shift하고 싶다면 최소 nbit mux가 있어야 한다. 예로 16bit shift하고 싶다면 최소 16bit mux가 필요하다.

6. 참고문헌

Barrel shifter/ https://www.techopedia.com/definition/17863/barrel-shifter

Ring Counter/http://www.vlsiencyclopedia.com/2011/10/ring-counter.html

Moore FSM과 Mealy FSM/

https://www.tutorialspoint.com/automata_theory/moore_and_mealy_machines.htm