컴퓨터 공학 기초 실험2 보고서

실험제목: Multiplier

실험일자: 2019년 10월 22일 (화)

제출일자: 2019년 11월 04일 (월)

학 과: 컴퓨터정보공학과

담당교수: 이준환 교수님

실습분반: 화요일 0, 1, 2

학 번: 2015722031

성 명: 박 태 성

1. 제목 및 목적

A. 제목

Multiplier

B. 목적

Multiplier를 직접 구현해보면서 binary multiplication과 booth multiplication의 원리와 그 차이를 이해하여 본다. 그리고, radix-2와 radix4 구현 방식의 장점과 단점을 비교해보면서 하드웨어가 곱셈 연산을 수행하는 방법에 대해 학습하는 데 목적을 둔다.

2. 원리(배경지식)

A. Binary Multiplication

Binary multiplication은 실제 곱셈 연산과 가장 유사하다. Multiplicand와 multiplier를 1 bit씩 곱하면서 shift한다. Multiplier가 음수라면(양수 * 음수) multiplicand의 2의 보수를 한 번 더한다. Logic의 크기가 크고 수행 시간이 길다는 특징이 있다.

B. Booth Multiplication

Binary multiplication의 단점을 보완하였다. Partial products에서 무의미하게 덧셈 연산을 수행하는 product를 제외하고 연산을 수행한다. Radix-2, radix-4 외에 다양한 방식이 있다. Radix-2는 multiplier의 2개의 비트를 묶어 비트의 패턴에 따라 특정 operation을 수행한다. 결과는 binary multiplication과 같다. 수행 cycle 또한 같다. Radix-4는 3개의 비트를 묶는다. 수행 cycle이 radix-2의 1/2배이다.

C. Radix-2

Radix의 크기가 커질수록 연산 수행 시간은 짧아진다. 그러나, 고려해야하는 경우의 수가 증가하고 이에 따라서 logic의 크기도 커진다. Radix-2는 연속된 1이나 0이 있으면 덧셈 연산을 수행하지 않고 연속된 끝 부분만을 연산하여 곱셈 결과를 얻는다. 따라서, 연산 횟수가 줄어든다. 그러나, 0과 1이 반복되는 구간이 존재한다면, 연산 횟수가 증가한다. 경우에 따라서 성능 저하가 발생한다.

X_i	X_{i-1}	Operation	Y_i
0	0	Shift Only	0
0	1	Add and Shift	1
1	0	Subtract and Shift	-1
1	1	Shift Only	0

3. 설계 세부사항

A. Next state logic

Current state을 input으로 받아서 next state을 output으로 내보낸다.

B. Output logic

Current state을 input으로 받아 state에 따른 result와 연산 종료 signal를 output으로 내보낸다.

C. Active low D Flip-Flop

Reset이 0이면 0을 출력한다. 1이면 d를 q로 내보낸다.

D. Calculate logic

Radix-2에 기반하여 곱셈 연산을 수행한다. 결과를 output logic에 전달한다.

E. Multiplier

i. Encoding State

State	Encoding	
IDLE	0	0
EXEC	0	1
DONE	1	1

ii. I/O configuration

Module	Port	Name	Bandwidth	Description
Multiplier	Input	clk	1-bit	Clock
		reset_n	1-bit	Active low에 동작
		multiplier	64-bits	승수
		multiplicand	64-bits	피승수
		op_start	1-bit	Start operation signal
		op_clear	1-bit	Clear operation signal
	Output	op_done	1-bit	Done operation signal
		result	128-bits	Multiplier result

iii. State transition diagram

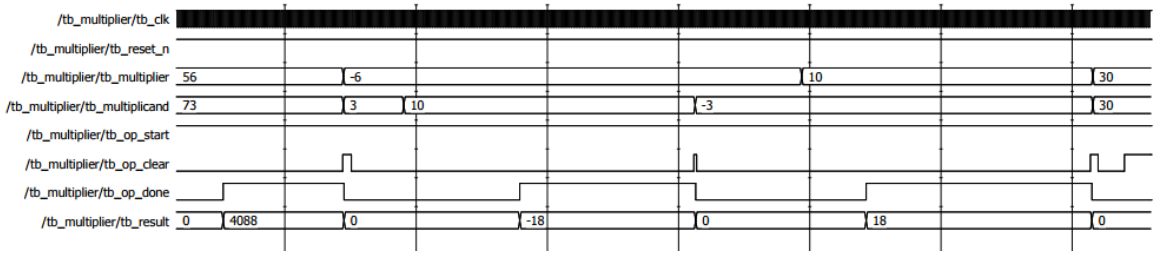
Current State	Reset_n	op_start	op_clear	op_done	count	Next state
X	0	X	X	0	0	IDLE
IDLE	1	1	X	0	0	EXEC
IDLE	1	0	X	0	0	IDLE
EXEC	1	X	0	1	64	DONE
EXEC	1	X	1	0	0	EXEC
DONE	1	X	X	1	!64	IDLE
DONE	1	X	X	1	64	DONE

- IDLE: op_start가 1이면 EXEC state로 바뀐다. 그 외에는 상태를 유지한다.

- EXEC: count가 64가 되었다면 DONE state로 바뀐다. 연산 도중 multiplier나 multiplicand가 바뀌어도 수행 중이던 연산의 결과가 출력된다. 연산 도중 op_clear가 high가 된다면, 연산을 중지하고 출력과 내부의 register들의 값을 모두 low로 초기화한다. 연산이 완료되면 op_done을 결과로 출력한다.
- DONE: state에 따라서 op_done signal과 0 또는 곱셈의 결과로 출력한다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과



Multiplier와 multiplicand의 부호에 따른 연산 결과는 다음과 같다.

i. 양수 * 양수

56 * 73 = 4088

ii. 음수 * 양수

-6 * 3 = -18

iii. 음수 * 음수

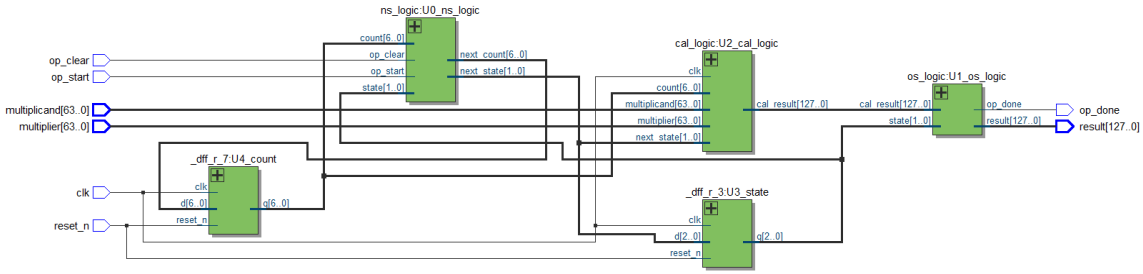
-6 * -3 = 18

연산 도중 op_clear가 high가 된다면, 연산을 중지하고 출력과 내부의 register들의 값을 모두 low로 초기화한다. 연산이 완료되면 op_done을 결과로 출력한다. 연산 도중 multiplier나 multiplicand가 바뀌어도 수행 중이던 연산의 결과가 출력된다. 64 clock 후에 op_done이 1, 그리고 연산 결과가 출력된다.

B. 합성(synthesis) 결과

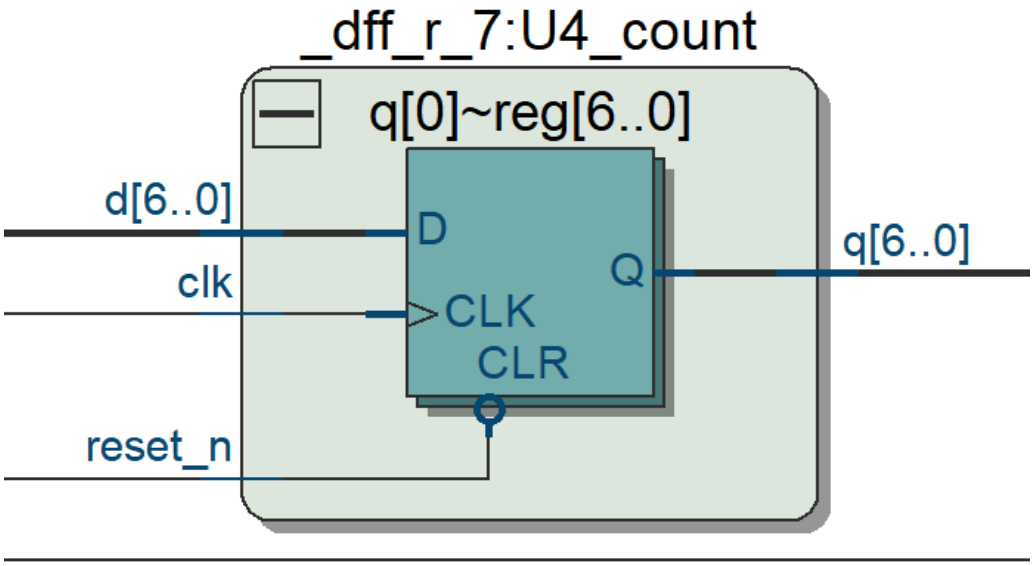
i. RTL Viewr

1. Top Module

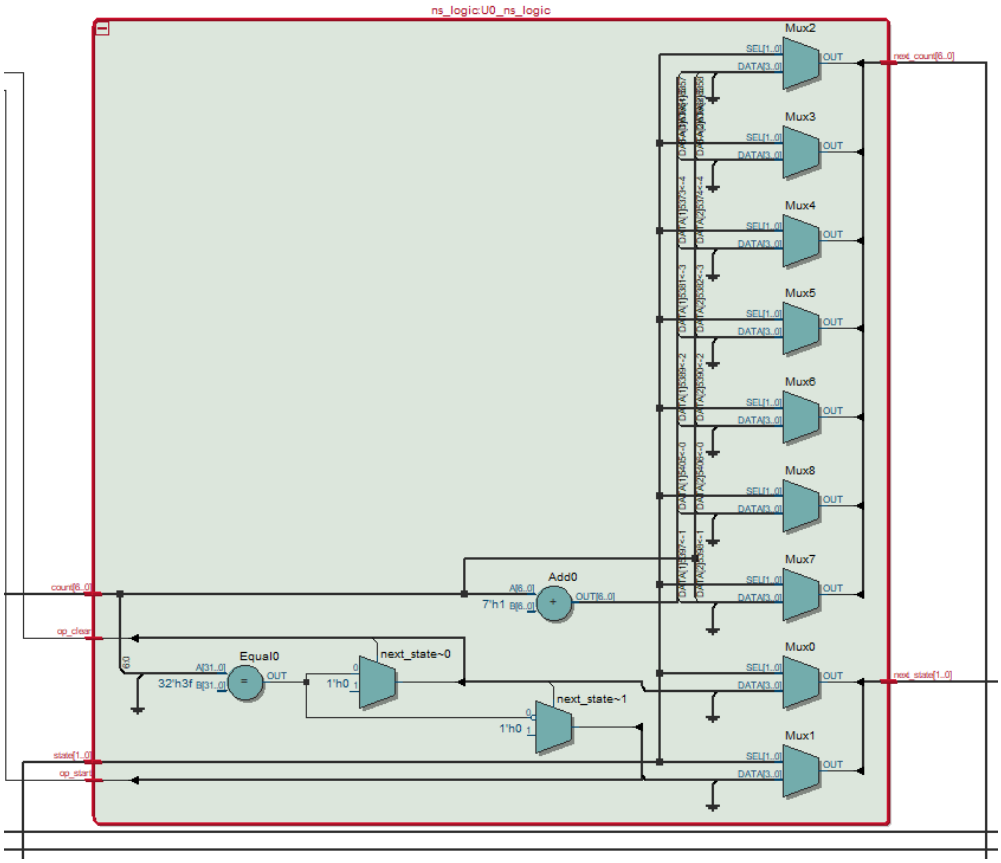


Top module은 ns_logic, cal_logic, os_logic, 그리고 2개의 active low d F/F으로 구성된다.

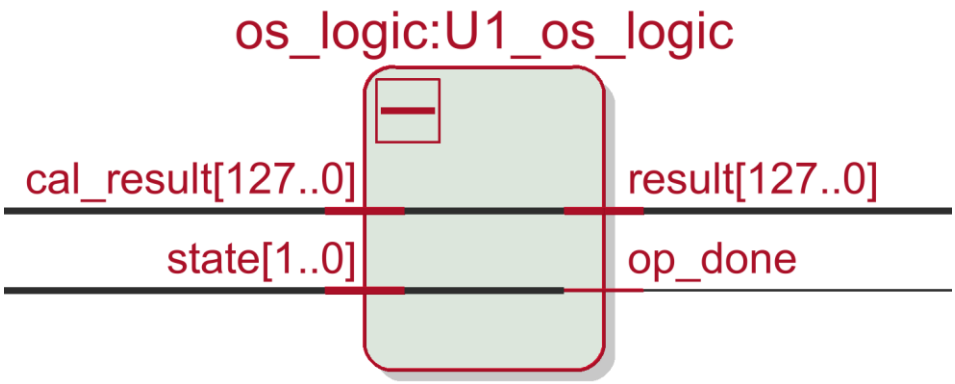
2. D F/F



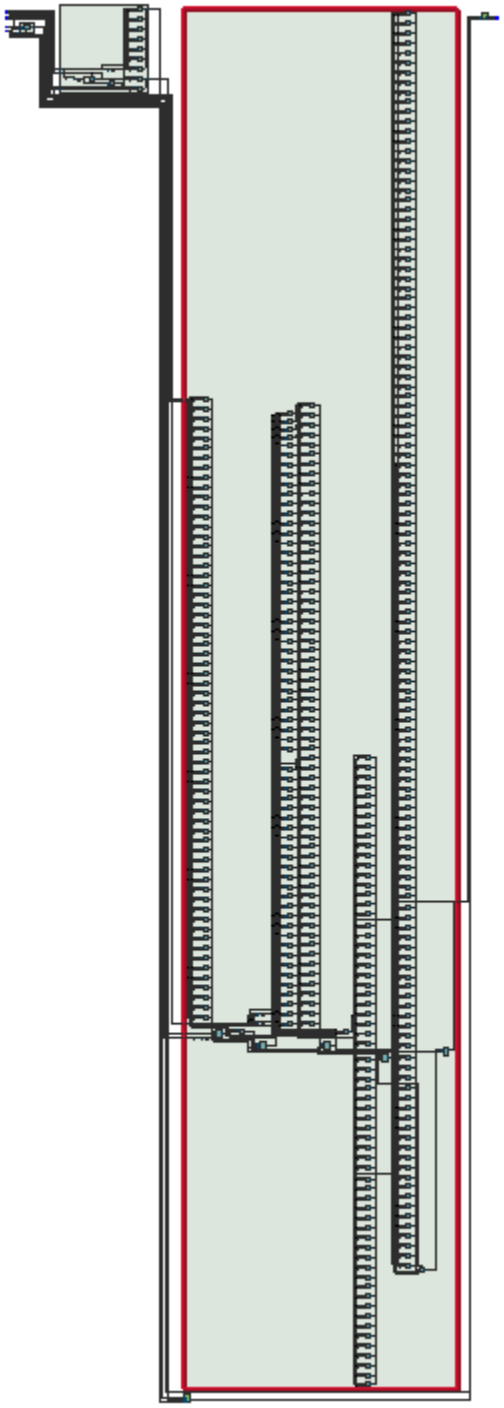
3. Next state logic



4. Output logic



5. Calculate logic



ii. Flow Summary

Flow Summary	
Flow Status	Successful - Sat Nov 02 17:48:06 2019
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Lite Edition
Revision Name	multiplier
Top-level Entity Name	multiplier
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	124 / 41,910 (< 1 %)
Total registers	330
Total pins	261 / 499 (52 %)
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 %)

Total pin은 261이고, logic utilization은 124이다.

5. 고찰 및 결론

A. 고찰

본 과제는 radix-2를 기반으로 수행하였음을 밝힌다. Radix-2는 연속된 1이나 0이 있으면 덧셈 연산을 수행하지 않고 연속된 끝 부분만을 연산하여 곱셈 결과를 얻는다. 따라서, 연산 횟수가 줄어든다. 그러나, 01이 반복되는 구간이 존재한다면, 연산 횟수가 증가한다. 경우에 따라서 성능 저하가 발생한다.

B. 결론

Radix-4가 radix-2에 비해서 수행 시간이 빠르다. 따라서 연산 효율을 고려한 설계를 한다면 radix-4를 구현하는 것이 좋다. 그러나, 고려할 경우의 수가 많아 크기가 커지는 단점이 있다. Radix-2가 수행 시간은 느리지만 고려할 경우의 수가 적어 설계는 비교적 간단하다.

6. 참고문헌

이준환/ 디지털 논리회로2/ 광운대학교/ 컴퓨터공학기초실험2-Multiplier/ 2019-2학기