디지털논리회로2

Assignment 2. Multiplier

학　　과　컴퓨터정보공학부
학　　번　2024402041
이　　름　심수정
제 출 일 자　2025.12.04 (목)

Assignment 2

1. Multiplier, Booth multiplication algorithm

A. Multiplier

Multiplier 는 곱셈기이다. 기본적인 곱셈 방식은 사람이 직접 손으로 계산하는 방식과 유사한 방식을 사용하지만 부호가 있는 정수를 처리할 때에는 추가로 부호 확장이나 다른 보정 로직이 필요하다는 단점이 있다. 이러한 단점을 보완하기 위해 booth multiplication algorithm 을 적용한 곱셈기를 설계한다.

B. Booth multiplication algorithm

이 알고리즘의 원리는 승수의 비트 패턴에 따라 덧셈과 뺄셈 연산을 수행하여 처리하는 것이다. 본 프로젝트에서는 Radix-2 알고리즘을 사용했다. 이 알고리즘은 현재비트와 이전비트의 두 개의 비트에 따라 4(같은 것을 묶으면 3 가지가 된다.)가지 규칙으로 부분 곱을 갱신한다. 먼저 00 이거나 11 일 때는 연산 없이 시프트만 수행한다. 01 일 때에는 피승수를 현재 부분 곱에 더하는 연산을 한다. 10 때에는 피승수를 현재 부분 곱에서 빼는 연산을 수행한다.

2. 설계 세부 사항

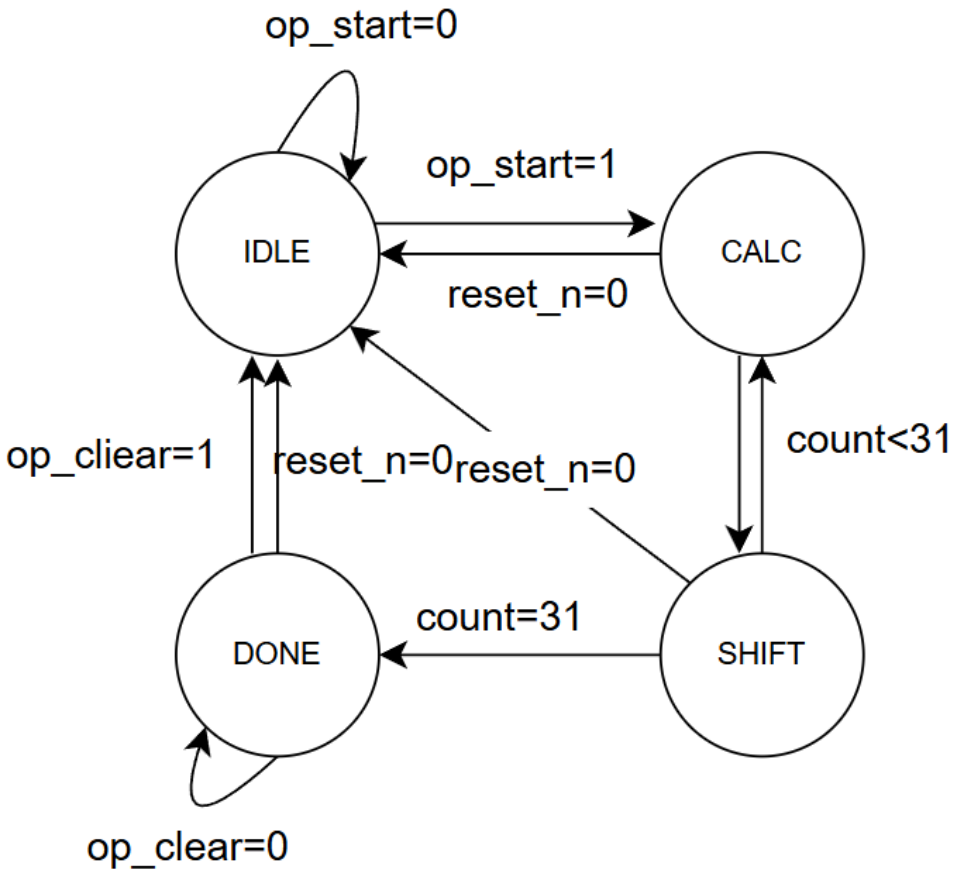
본 multiplier 는 Verilog 로 구현한 Radix-2 Booth 곱셈기로, 32 비트 승수와 피승수를 입력받아 내부 상태 머신을 통해 순차적으로 곱셈을 진행하도록 설계하였다. 연산 과정에서 p_reg 는 64 비트의 부분 곱 레지스터로 사용되며, 하위 32 비트에 승수를, 상위 32 비트에 부호 확장 및 Booth 연산 결과를 저장한다. 피승수는 m_reg 에 저장되며, Booth 알고리즘의 연산 규칙에 맞춰 상위 32 비트에 더해지거나 빼지도록 add_val 과 sub_val 을 활용하여 자리 정렬을 맞춰준다. Booth 알고리즘의 핵심인 두 비트 비교는 p_reg 의 최하위 비트와 이전 비트 역할을 하는 q_0 를 묶어 판단하며, 조합이 01 이면 p_reg 상위 비트에 m_reg 를 더하고, 10 이면 m_reg 의 2 의 보수를 더해 뺄셈을 수행하며, 00 과 11 일 때는 별도의 연산 없이 다음 단계로 넘어간다. 이후 SHIFT 단계에서는 p_reg 전체와 q_0 을 포함해 산술 우측 시프트를 수행하여 부호를 유지한 채 연산을 진행하며, 시프트할 때 p_reg[0]은 q_0 으로 이동해 다음 Booth 판정에 사용된다. 이러한 CALC 와 SHIFT 단계는 count 레지스터가 31 에 도달할 때까지 반복되며, 총 32 회의 Booth 사이클을 통해 결과를 만들어낸다. 전체 연산 흐름은 IDLE, CALC, SHIFT, DONE 의 네 가지 상태로 구성된 상태 머신으로 제어되는데, IDLE 에서는 입력을 초기화하며 op_start 신호가 들어오면 연산 준비를 마친 뒤 CALC 상태로 진입한다. CALC

상태에서는 Booth 연산 규칙에 따른 덧셈 또는 뺄셈을 수행하고, SHIFT 상태에서는 산술 우측 시프트와 카운터 증가를 수행한다. count 가 31 이 되면 DONE 상태로 이동하며, 이때 p_reg 의 전체 값을 result 로 출력하고 op_done 신호를 1 로 설정해 연산 완료를 표시한다. op_clear 신호가 들어올 경우 모든 레지스터와 상태는 즉시 초기화되어 다시 IDLE 상태로 돌아가도록 구성하였다.

3. State Encoding Table

State	Encoding	Description
IDLE	00	대기 상태. Op_start 의 신호를 기다리며 초기화를 수행한다.
CALC	01	연산 상태. 덧셈이나 뺄셈을 수행한다.
SHIFT	10	이동 상태. Shift registor 를 한 번 수행하고 카운터를 증가시킨다.
DONE	11	완료 상태. 결과값을 출력하고 op_done 신호를 1 로 유지함

4. State Diagram



모든 상태에서 reset_n 이 0 이 될 때는 IDLE 로 돌아간다.

5.I/O configuration

-Input Pin

Name	Bits	Description
clk	1	클럭
reset_n	1	Active low 리셋
multiplier[31:0]	32	승수
multiplicand[31:0]	32	피승수
op_start	1	연산 시작
op_clear	1	연산 초기화

-output pin

Name	Bits	Description
op_done	1	연산 완료

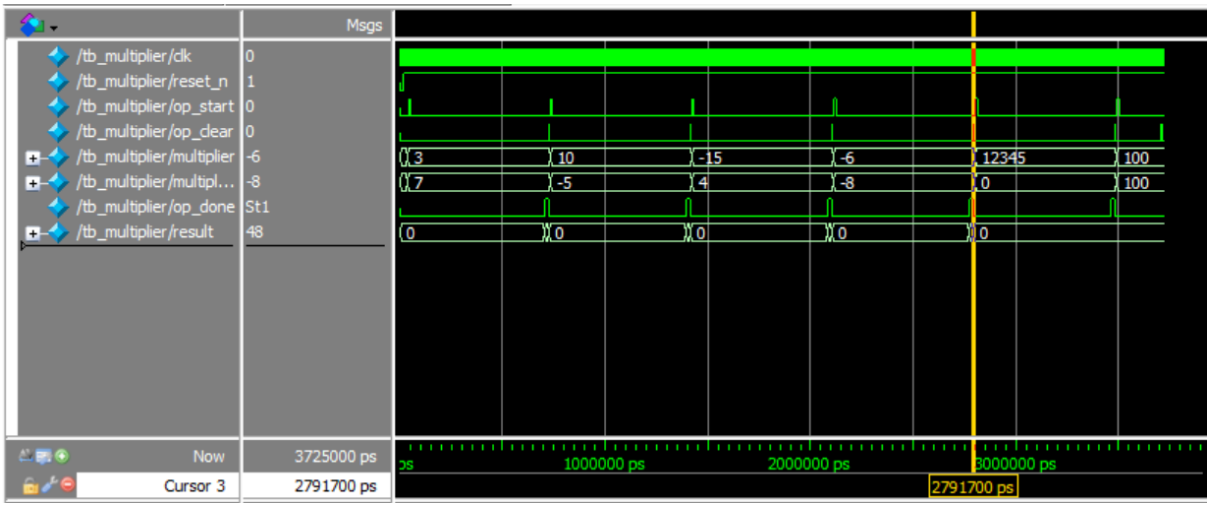
result[63:0]	64	결과

6. Testbench 시뮬레이션 결과

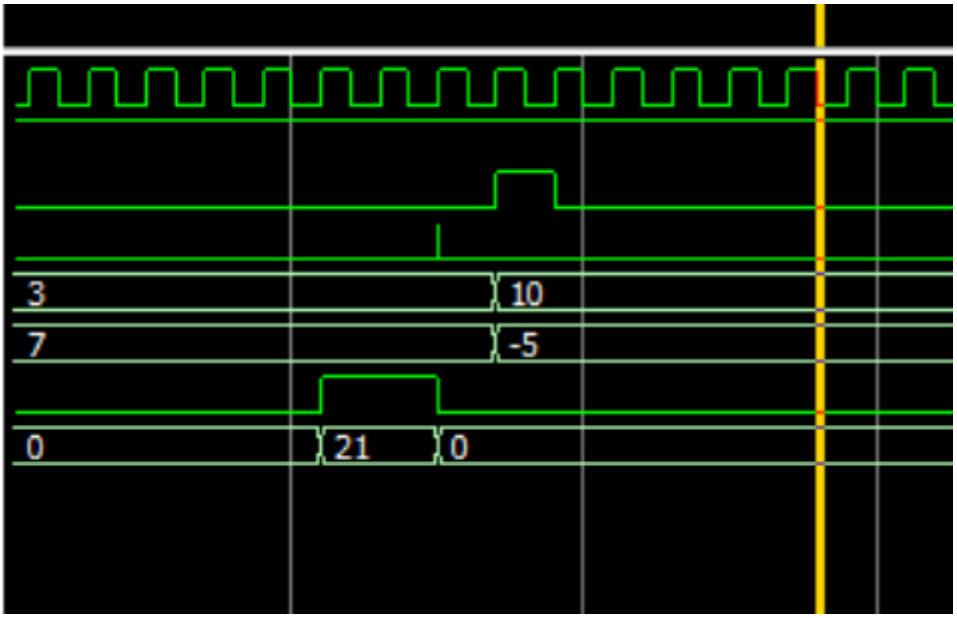
-Transcript 에서 확인할 수 있는 결과

```
[PASS]          3 *          7 =         21 (Expected:         21)
[PASS]         10 *         -5 =        -50 (Expected:        -50)
[PASS]        -15 *          4 =        -60 (Expected:        -60)
[PASS]         -6 *         -8 =         48 (Expected:         48)
[PASS]      12345 *          0 =          0 (Expected:          0)
----------------------------------------
```

-전체 waveform

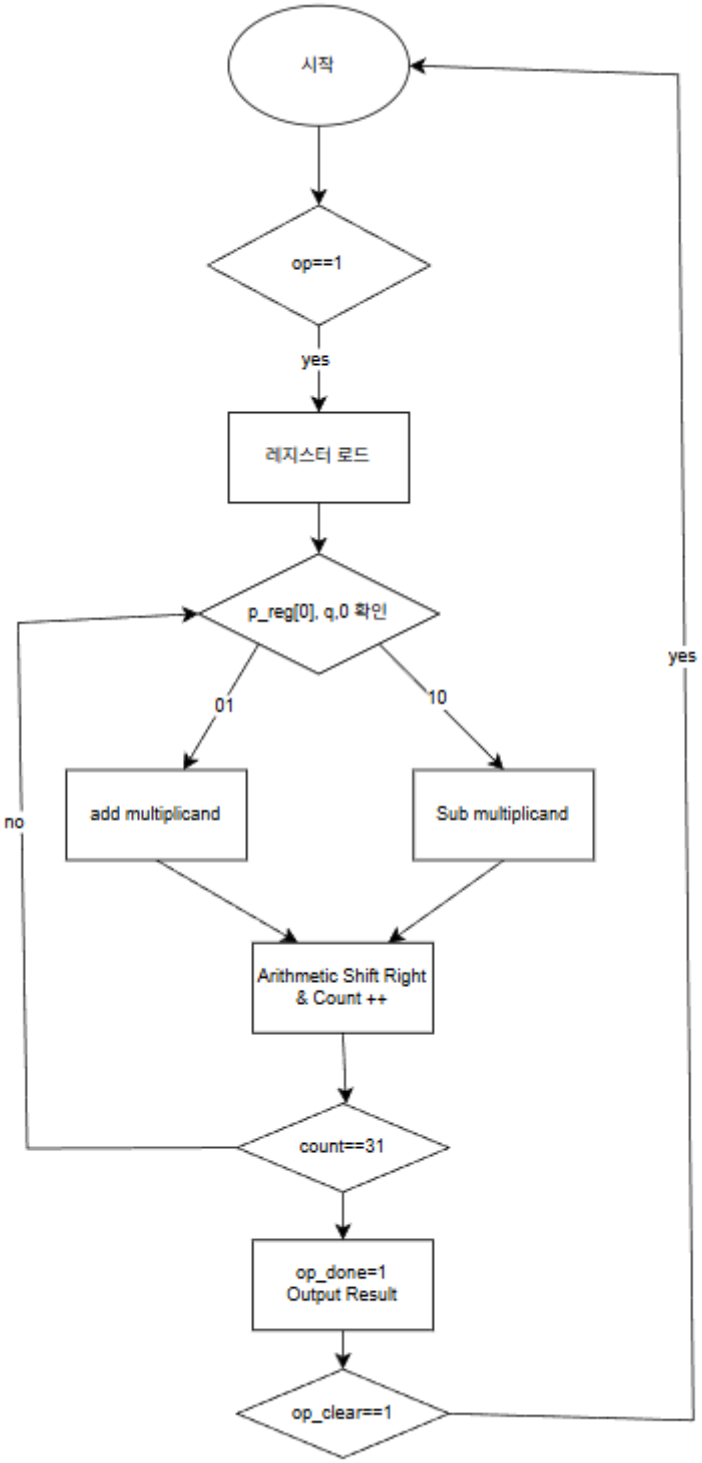


-확대 waveform



clear_done (밑에서 두번째 줄) 이 1 일 때 result (가장 밑 줄)가 출력됨을 확인할 수 있다.

7. Flow Summary



Multiplier 모듈은 Radix-2 Booth Algorithm 을 사용해 32-bit 곱셈을 수행하며, 전체 동작은 IDLE, CALC, SHIFT, DONE 의 4 개의 상태를 순환하면서 제어된다.

A. IDLE 상태

IDLE 는 초기화 상태나 대기 상태로 reset_n 의 신호가 Low 이거나 프로그램 초기 상태일 때 들어간다. Op_start 신호가 1 이 될때까지 이 상태에 머문다.

op_start 신호가 1 이 되면 피승수를 내부 레지스터에 저장하고, 승수를 곱셈 결과 레지스터의 하위 32 비트에 로드한다. q_0, count 를 0 으로 초기화 하고 CALC 상태로 넘어간다.

B. CALC 상태

CALC 상태는 연산 상태로 Booth Algorithm 을 적용한다. 현재 승수의 최하위 비트 p_reg[0]와 이전 비트 q_0 를 검사해 그 두 값에 따라 다른 연산을 수행한다. 두 비트가 01 일 때에는 피승수를 p_reg 의 상위 32 비트에 더하는 연산을 하고, 두 비트가 10 일 때에는 피승수를 p_reg 의 상위 32 비트에서 뺀다. 00 이거나 11 일 때에는 연산을 수행하지 않는다. 연산을 하고나면 바로 SHIFT 상태로 넘어간다.

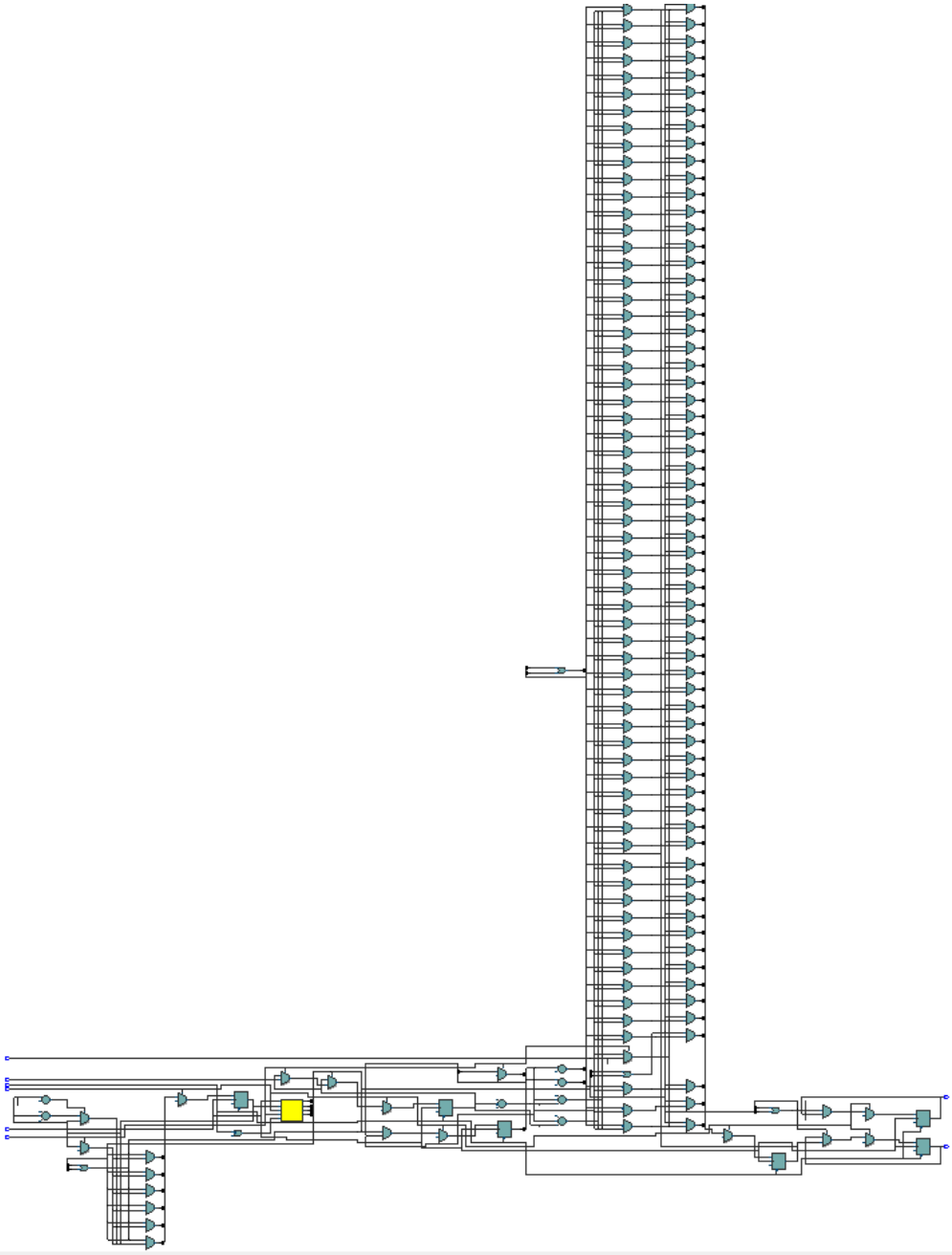
C. SHIFT 상태

SHIFT 상태에서는 오른쪽 쉬프트를 진행하고, 카운터를 증가시킨다. 쉬프트는 최상위 비트는 유지한 채 p_reg 와 q_0 를 포함한 전체 레지스터를 오른쪽으로 1 비트 이동시키는 것을 말한다. Count 가 31 보다 작을 때에는 다시 CALC 상태로 돌아가서 다음 비트 연산을 수행한다. Count 가 31 이 되었을 때에는 DONE 상태로 넘어간다.

D. DONE 상태

DONE 상태에서는 최종 연산결과를 result 로 내보내고, op_done 을 1 로 바꾸어 연산이 완료 되었음을 표시한다. Op_clear 신호가 들어오기 전까지 이 상태를 유지한다. Op_clear 신호가 들어오면 모든 레지스터를 초기화하고 다시 IDLE 상태로 돌아간다.

8. RTL Viewer



RTL Viewer 에서는 Booth 연산을 위한 덧셈기, 쉬프터, FSM 제어 블록이 자동 생성된 구조를 확인할 수 있다.