SE273 Term Project Report

13조

201911189 한현영, 201911127 이승우

I. 도입

A. 목적

본 프로젝트의 궁극적인 목적은 다양한 ALU를 구현하는데 있다. Adder Subtractor, Multiplier를 그 목적에 맞게 Combinational, Sequential logic / Hierarchical design / Synchronous design과 같이 다양한 방식으로 설계한다. 이후 test bench를 작성하여 고안한 logic이 설계 목적에 맞는지, input 대비 output이 정상적으로 도출되는지에 대해 검증하는 과정을 거친다.

B. 목표 및 기준 설정

|  |  |
| --- | --- |
| 목표 | 기준 |
| Adder  정상 작동 | Testbench 내 임의의 두 8bits operand 간의 가산이 정상적으로 작동하는지 검증한다. (e.g. 00000110 + 00001101 = 00010011) |
| Subtractor  정상 작동 | Testbench 내 임의의 두 8bits operand 간의 감산이 정상적으로 작동하는지 검증한다. (e.g. 00000110 - 11110011 = 00010011) |
| Multiplier  정상 작동 | Testbench 내 임의의 두 8bits operand 간의 감산이 정상적으로 작동하는지 검증한다. (e.g. 00001111 \* 00001010 = 10010110) |
| 전체적인 state 정상 작동 | Testbench 내 임의의 두 8bits operand 가산, 감산, 곱셈이 정상적으로 작동하는지 검증한다. 더불어 여러 signal(SYS\_CLK, SYS\_RESET\_B, OP\_CODE 등)에 logic이 정상적으로 대응하는지 관찰한다. 또한 DATA\_A, DATA\_B에 대한 DATA\_C 가 정상적으로 출력되는지 관찰한다. |

C. 팀원간 역할

|  |  |
| --- | --- |
| 이름(학번) | 역할 |
| 한현영(201911189) | SM diagram design / Adder(Subtractor), Top calculator, Testbench implementation / code version management(Git) |
| 이승우(201911127) | 채우시오 |

II. 합성 및 분석

A. State Diagram

B. 시뮬레이션을 통한 모듈 검증

III. 결과 및 논의

A. 결과 도출

아래 결과처럼, 설계한 Adder, Subtractor, Multiplier 모두 input에 따라 정상적으로 작동함을 알 수 있다. 연산 결과 뿐 아니라, 외부적인 signal의 I/O 처리도 정상적으로 함을 볼 수 있다.

--밑에 사진 첨부--

B. 토의

i. 제시된 주요 설계요소 및 제한요소를 만족하는가

제시된 Reference에서 언급된 Block Diagram, Signal Description 모두 포함하여 설계하였고, 주어진 OP\_CODE(001, 010, 100) 외 다른 값이 들어올 경우 예외까지 처리하였다.

1. VHDL의 세 가지 표현 방식인 Structural Description, Behavioral Description, Data Flow Description의 방법을 모두 사용하여 설계하였는가

2. Shift and add algorithm에 의해 SM chart를 작성한 후 multiplier를 설계하였는가

3. 설계한 Adder, Subtractor, Multiplier가 부호를 고려하는가

부호를 고려한다. Testbench를 통해서도 확인 가능하다.

--밑에 사진 첨부--

ii. Top-down 방식의 설계가 적용되었는가

구현에 앞서 각 state를 잇는 조건과 해당 state에서 어떠한 action을 취해야 하는지에 대한 State Machine을 구상하였다. 이후 ALU에 진입하였을 때 어떠한 algorithm을 사용하여 연산을 할지 궁리하였다.

iii. Hierarchical Design 방식의 설계가 적용되었는가

가령 ALU의 경우 세 operation 모두 ADDER를 기반으로 작동한다. 기본적으로 ADDER를 구현한 뒤, 덧셈과 곱셈은 Sign <= ‘0’인 ADDER를, 뺄셈은 Sign <= ‘1’인 ADDER에 기인하여 동작한다.

iv. Synchronous Design 방식의 설계가 적용되었는가

v. Testbench의 작성 요령 및 임의의 입력에 대해서 안정적으로 동작하는가

vi. 기타 논의사항

Multiplier 구현 시 매 state마다 shift and add를 해주는 방식으로 설계하였다. 이 때문에 코드가 길어졌는데, 이를 간결하게 표현할 수 있는 방안에 대한 논의가 필요하다. 가령 각 adder. shifter를 modularize하여 표현할 수 없는지 생각해보아야 한다.