디지털 논리2 및

컴퓨터 공학 기초 실험2   
보고서

과제제목: Memory & Bus

실험일자: 2017년 10월 27일 (금)

제출일자: 2017년 11월 02일 (목)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 금요일 5, 6, 7

학 번: 2012722028

성 명: 장 한 별

1. 제목 및 목적
   1. 제목

Memory & Bus

* 1. 목적

Address에 기반하여 데이터를 저장하고 불러오는 기능을 이해하고 이를 이용하여 간단한 hardware인 ram을 설계한다. 또한 여러 component간에 data를 전송할 수 있도록 연결해주는 component인 bus의 개념에 대해서 이해하고, 이를 Verilog를 통해서 설계하고, 구현해 보도록 한다.

1. 원리(배경지식)
2. Memory

Memory는 Data를 저장하고 전송 하기 위해서 data를 불러 올 수 있는 기억 장치이다. Memory에는 많은 Memory가 존재하지만, Address를 기준으로 사용 구간을 할당 해 놓았다. 이를 Memory Map 이라고 한다. 이에 대해서 address decoder에 의해서 상위 bit를 통해서 구간을 나눌 수 있다. 예를 들어서, 8bit의 memory가 존재한다고 가정해보자. 하나의 구간을 0000\_0000 에서 0001\_1111로 나누고, 다른 하나의 구간을 0001\_0000 에서 0000\_ffff로 또다른 하나의 구간을 0100\_0000에서 0101\_ffff로 나누어 놓았다고 가정해보자. 8bit모두를 보고 판별하기에는 매우 효율적이지 않다. 이때, 상위 2번째비트와 3번째비트를 제외하고는 비트 값이 변하지 않는다 따라서, 특정 bit만으로 해당 Address구간을 나누어 쉽게 확인할 수 있도록 하는 과정을 Address decoding이라고 한다. 이러한 구조를 가진 Memory는 다른 Component에서 data를 저장하거나 불러올 때, 입력 받은 Address에 따라서 사용된다.

* 1. ROM

ROM 은 Read Only Memory 의 약자로 읽기 기능만 가능하고 내용을 수정하지 못한다. 그리고 한 번 ROM에 입력된 데이터는 절대 지워지지 않는 특성이 있는데 컴퓨터가 전원이 꺼져도 해당 데이터는 지워지지 않아 컴퓨터의 운영체제 등에 사용된다.

* 1. SRAM

SRAM 은 RAM 중 Staitc RAM 을 나타낸다. SRAM은 flipflop에 bit 데이터를 저장하여 보존하는 특성이 있다.

SRAM으로 데이터를 쓸 때 Word선 전위를 high, bit선의 전위를 부여해서 flipflop 상태를 결정후 word 선 전위를 low 시켜 1상태를 만든다

* 1. DRAM

DRAM 은 RAM 중 Dynamic RAM 을 나타낸다. DRAM은 capacitor 와 transistor 에 bit 데이터를 저장하고 보존하는 특성이 있다. 이때 capacitor 의 전하가 짧은 주기를 가지고 있기 때문에 지속적인 충전이 되지않으면 전하를 잃어 반복적인 충전이 불가피하다.

데이터를 쓸 때, Word선 전위를 high, Bit선 전위를 high, Word선 전위를 low 시켜 1 상태를 만든다.

1. Bus

시스템상 Bus의 뜻은 본래 bus의 뜻과 유사하게 여러 가지 component간의 data를 전달시켜주는 component이다. 즉, 하나의 component로부터 data를 받아서 다른 component로 전달해주는 component이다. Bus는 새로운 component들을 추가하기가 쉽고, 가격이 저렴하다는 특징을 가지고 있다. Bus는 특징 중에 한 순간에 하나의 데이터만 전송이 가능하다는 단점이 있고 이때 다른 데이터들은 기다려야 한다. 이때 전송 요청을 하는 쪽을 Bus master라고하고, 전송 요청을 받는 쪽을 Bus slave라고한다. 여기서 component간의 데이터를 전송해주는 과정 중에 여러 개 의 전송 동작이 요구되면 오류가 발생된다.

이를 해결하기 위하여, 특정 조건에 대해 특정 data를 transfer하겠다 하는 circuit이 바로 arbiter이다. Arbiter는 특정 grant값일 때 특정한 request을 받아 어떠한 grant로 넘어가도록 하는 역할을 한다.

Read operation 수행 시 select 신호를 register를 통해서 거치게 되는데 만약 select 신호가 register를 거치지 않을 경우 발생하는 문제는 clock 과 동기화(synchronous) 되지 않아 원하는 신호를 mux 가 처리하지 못해 원하는 결과 값이 나오지 않게 된다. 이러한 문제를 해결하기 위해 read operation 수행 시 select 신호를 register 를 통해 거치게 해서 synchronous 하게 만든다.

1. 설계 세부사항
2. Memory

|  |  |  |  |
| --- | --- | --- | --- |
| <I/O description> | | | |
| Port | Name | Bandwidth | Description |
| Input | clk | 1 bit | Clock |
| cen | 1 bit | chip enable |
| wen | 1 bit | Write enable |
| addr | 5 bit | address |
| din | 32 bit | data in |
| Output | dout | 32 bit | data out |
| Reg | dout | 32 bit | data out |
| mem | 32 \* 32bit | memory space |
| 위 표는 memory의 I/O Description이다. input으로는 clk, cen, wen, addr, din이 들어가고 output으로는 dout이 나오게 된다. 또한 사용되는 reg는 dout과 mem이다. 이때 mem 은 memory 공간으로써 32bit 내부에 가 32개의 data를 사용하도록 설계하였다. | | | |

|  |
| --- |
| <Block diagram> |
|  |
| 위 그림은 memory의 block diagram이다. input으로써 cen, wen, addr, din, clk이 들어가고 이때, addr 은 5비트이고, din은 32bit이다. 또한 output으로는 32bit의 dout이 나오게 된다. Cen의 뜻은 chip enable 이다. 해당 ram이 동작을 할 것인지 안할 것인지 판별하는 역할이다. Wen은 write enable 이다. 해당 signal이 1로 들어오면 address가 가리키는 memory에 입력 값 din을 write하는 구조로 설계하였다. 이때 cen, wen모두 1이면 addr이 가리키는 memory에 입력 값 din을 write하고 0을 출력하도록 설계하였고, cen이 1이고 wen이 0이면 din을 신경쓰지 않고, addr이 가리키는 memory 값을 dout에 출력하고, cen이 0 이면 wen은 신경쓰지 않고, dout은 그냥 0을 출력하는 식으로 설계하였다. |

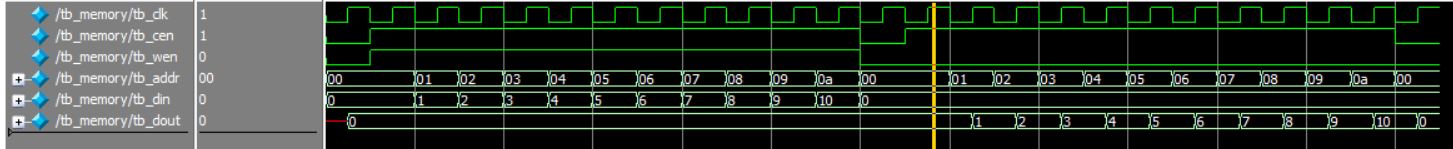
1. Bus

|  |  |  |  |
| --- | --- | --- | --- |
| <I/O Description> | | | |
| Port | Name | Bandwidth | Description |
| Input | clk | 1 bit | clock pulse |
| reset\_n | 1 bit | reset signal |
| M0\_req | 1 bit | Master0 request |
| M1\_req | 1 bit | Master1 request |
| M0\_wr | 1 bit | Master0 write, read enable |
| M1\_wr | 1 bit | Master1 write, read enable |
| M0\_address | 8 bit | Master0 address |
| M1\_address | 8 bit | Master1 address |
| M0\_dout | 32 bit | Master0 data out |
| M1\_dout | 32 bit | Master1 data out |
| S0\_dout | 32 bit | Slave0 data out |
| S1\_dout | 32 bit | Slave1 data out |
| Output | M0\_grant | 1 bit | Master0 grant |
| M1\_grant | 1 bit | Master1 grant |
| M\_din | 32 bit | Master din |
| S0\_sel | 1 bit | Slave0 select signal |
| S1\_sel | 1 bit | Slave1 select signal |
| S\_wr | 1 bit | Slave write or read enable |
| S\_address | 8 bit | Slave address |
| S\_din | 32 bit | Slave data in |
| wire | to\_mx | 2 bit | From flip flop to mux |
| 위 표는 bus의 I/O Description이다. Input과 output값을 해당 값들로 설정 하였고, 이때 사용된 wire는 flip flop에서 3 input 32bit mux의 sel 신호로 들어가기 위하여 설정해 주었다. | | | |

|  |  |  |
| --- | --- | --- |
| <Module Description> | | |
| Classification | Name | Description |
| Module | bus | Bus |
| instance | U0\_bus\_arbit | Arbitrator |
| U1\_mux2 | 2 input 1bit mux |
| U2\_mux2\_8bit | 2 input 8bit mux |
| U3\_mux2\_32bit | 2 input 32bit mux |
| U4\_bus\_addr | Address decoder |
| U5\_dff\_r | 2 input resettable d flip flop |
| U6\_mux3\_32bit | 3 input 32bit mux |
| 위 표는 최상위 모델 bus의 Module description이다. 해당하는 모듈 7개를 instance하여 설계하였다. arbitator에서 master의 request를 받아 해당하는 grant를 결정하게 되고, 이 grant의 신호로 mux의 select signal로 들어가게 된다. 그리고, 선택된 master의 address의 주소 값을 찾아서 address decoder에서 s0\_sel을 선택할지 s1\_sel을 선택할지 결정하게 된다. 이후 slave에서 dout을 내보내게 되고, sel에 따라서 din을 dout으로 출력하게 된다. | | |

|  |
| --- |
| <arbitrator> |
|  |
| 위 그림은 Arbitrator 의 state diagram이다. 해당 state일때의 조건과, 다음 state로 넘어갈 때의 조건의 그림과 같이 설계하였음을 확인할 수 있다. |

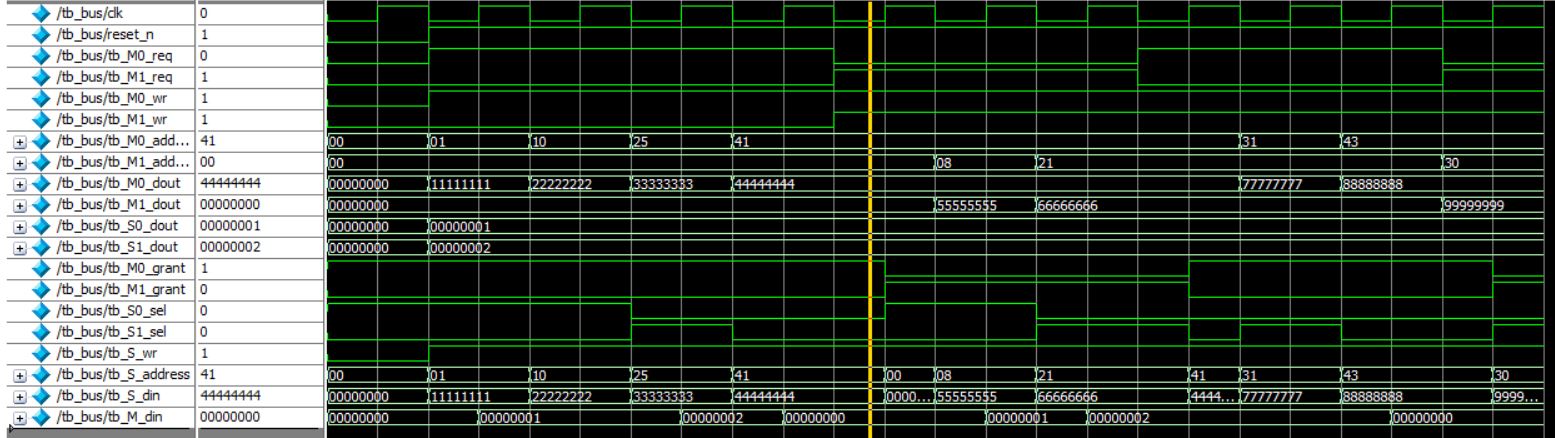
1. 설계 검증 및 실험 결과
   1. 시뮬레이션 결과
      1. memory



< memory 의 wave form>

위 그림은 memory의 waveform화면이다. 처음 cen과 wen을 다 0으로 주어서 dout이 0이 출력되고 있고, 그 다음 cen과 wen을 모두 1로 넣어주어서 해당하는 address에 din값을 넣어주고 있고 dout에는 넣어주는 din값이 그대로 출력되고 있음을 확인할 수 있다. 또한 중간쯤 넘어가서 0부터 10까지 메모리에 넣어준 뒤 wen과 cen값을 모두 0으로 바꾸어 초기화 한 뒤에 cen만 1로 넣어서 해당하는 address에 저장되어있는 값들을 출력하고있다. Address에 1부터 10까지의 메모리주소를 순차적으로 호출하고있고, 이 주소 값을 받고 clk이 rising edge일 때 해당하는 메모리주소에 저장되어있는 값이 출력됨을 확인할 수 있다.

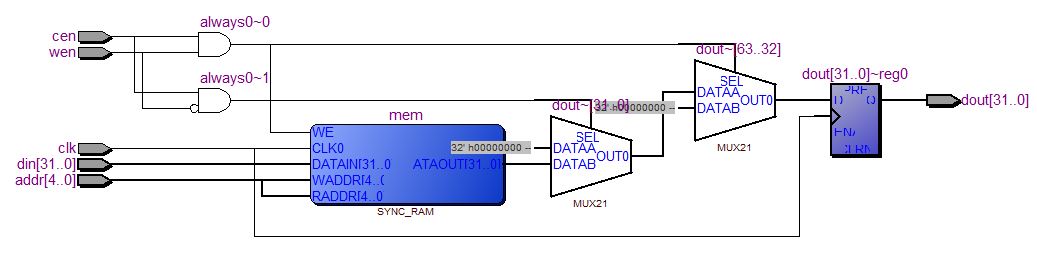
* + 1. bus



< bus 의 wave form >

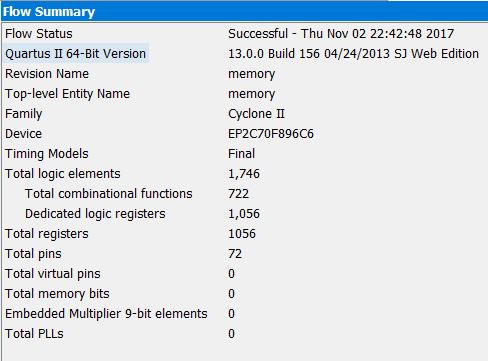
위 그림은 bus의 testbench의 Wave form화면이다. 처음 모든 값을 0으로 초기화해준뒤 reset신호를 0으로 주며 M0 grant에서 시작하였고 이후로 1을주었다. S0\_dout과 S1\_dout은 1,2로 설정하였다. Address에서 0부터 20까지는 slave0로 설정하였기 때문에 16진수로 표현했을 때 처음 01,10을 넣어주었을 때는 s0\_sel이 1로 set되어있고, 20부터40까지는 slave1로 설정하였기 때문에 25를 넣는 순간 s1\_sel이 1로 set되고 s0\_sel은0으로 clear되는 것을 확인할 수 있다. 또한 40부터는 예외 값으로 어떤 slave도 받지 않는 것으로 설정하였기 때문에 s0\_sel, s1\_sel모두 0으로 clear되는 것을 확인할 수 있고 이때의 M\_din값도 0으로 바뀌는 것을 확인할 수 있다. 이후 M1\_req를 1로 주어 M1 grant의 지시를 따르게 되고, 해당 address값에 55555555와 66666666을 순차적으로 넣어주었고 address값이 해당 범위 내에 있는 address를 주었기 때문에 output값이 출력이 잘 되고있는 것을 확인할 수 있다.

* 1. 합성(synthesis) 결과
     1. memory



< memory 의 RTL Viewer >

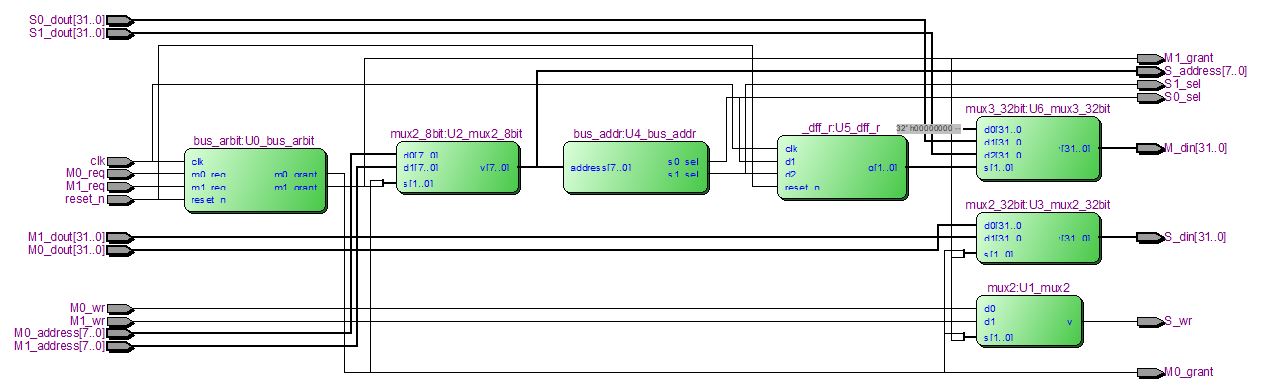
위 그림은 Memory의 RTL Viewer이다. 해당하는 input값들을 넣어주었을 때 해당하는 output값이 나오도록 설계하였고 위에 design했던 table대로 구현이 되어있음을 확인할 수 있다.



< memory 의 Flow Summary >

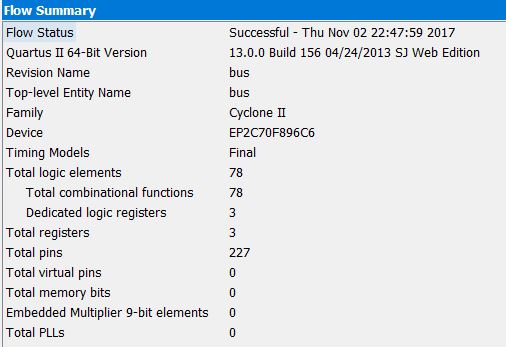
위 그림은 Memory의 Flow Summary이다. Total logic elements가 1746개 사용되었음을 확인할 수 있고, total register는 1056개 사용되었고, total pins는 72개가 사용되었음을 확인할 수 있다.

* + 1. bus



< bus 의 RTL Viewer >

위 그림은 bus의 RTL Viewer이다. 해당하는 input값들을 넣어주었을 때 해당하는 output값들이 나오도록 설계하였고 위에 design했던 table대로 구현이 되어있음을 확인할 수 있다.



< bus 의 Flow Summary >

위 그림은 bus의 Flow Summary이다. Total logic elements가 78개 사용되었음을 확인할 수 있고, total register는 3개 사용되었고, total pins는 227개가 사용되었음을 확인할 수 있다.

1. 고찰 및 결론
   1. 고찰

이번 실습시간에는 memory와 bus를 구현하였다. Memory는 간단히 해당 조건에 따라서 입력 값을 해당 메모리 주소 값에 저장하고 특정 신호에 따라서 무엇을 출력하는 간단한 문제였다. 따라서 memory를구현하는데에는 큰 어려움이 없었지만 문제는 bus였다. 처음 bus의 개념을 수업시간에 들었을 때는 크게 어려움을 느끼지 않았다. 그냥 데이터를 전송해주는 어떠한 component구나 라고 간단하게 생각했지만, 구현을 직접 하는 데에는 여러 가지 조건도 많이 필요했고 decoder, arbitrator등 여러 가지 모듈도 필요하였다.

처음 arbitrator를 구현 하는 데에는 강의 자료를 보고 마냥 구현을 하였는데 이해를 하지않고 구현을 하다 보니 오류 가나도 어디서 오류가 났는지 찾을 수가 없었다. 따라서 인강을 다시 듣고, 개념을 하나하나 이해를 하고 다시 구현을 하니 오류 부분을 정확히 찾을 수 있었다. 또한 까다로웠던 부분은 mux부분이었다. 처음 3input mux를 제외하고는 모두 3항연산자를 사용하여 구현을 하였는데, select신호가 모두 2비트로 받아야 된다는 점을 뒤늦게 깨닫고, case문으로 다시 재 구현을 하였고 모두 2비트신호로 주었다. 생각해보니 0이면 G0을 받고 1이면 G1을 받는 신호를 받는 경우라면, 굳이 2비트로 신호를 주지 않고 1비트로도 구현이 가능할 것으로 생각이 든다. 다음 번 모듈을 구현을 할 때는 이런 점도 고려해서 다른 방식으로 짜보는 것도 좋을 것 같다.

* 1. 결론

이번 실습을 통해 bus와 memory의 개념을 이해할 수 있었다. Memory에서 특정 영역에 address를 지정해야 하고, 이때 이 address는 decoder를 통해 decode를하여 구간을 나눠 놓아야 한다. 그렇게 하지 않으면 주소에 접근한다거나, 데이터를 수신 즉, 데이터를 transfer할 때 오류가 발생할 가능성이 크다. 또한 memory는 data를 저장하고, 복사 할 수 있는 기억장치, 즉 register의 역할을 한다고 보면 된다. 따라서 원하는 address를 통해 특정 register에 저장되어있는 값을 불러오거나 저장을 할 수 있는 역할을 한다는 뜻이다. 이때, data를 전송해주는 역할을 하는 것이 bus이다. Bus는 한 순간에 하나의 데이터만 전송이 가능하다는 단점이 있고 이때 다른 데이터들은 기다려야 한다. 이때 전송 요청을 하는 쪽을 master라고하고, 전송 요청을 받는 쪽을 slave라고한다. bus는 한 개의 data만 transfer 하므로 여러 개의 slave나 master가 있어도, master가 지정한 특정 address에만 접근하게 된다.

1. 참고문헌

공진흥 / 컴퓨터공학기초실험2 / 새빛관303호(광운대학교) / 2017년.

이준환 / 디지털논리회로2 / 참빛관B101호(광운대학교) / 2017년.