컴퓨터 공학 기초 실험 2

---

1. Half Adder

- Functional Description
  - ✓ Half adder 는 2 개의 1-bit 입력을 받아 sum 과 carry out 을 출력하는 가산기이다. A, B 두 개의 입력을 받아 sum 을 의미하는 S 와 carry out 을 의미하는 co 을 출력한다.
- Truth Table & Karnaugh Map
  - ✓ Truth Table

Input		**Output**	
a	b	co	Sum s
0	0		
0	1		
1	0		
1	1		

  - ✓ Karnaugh Map and Boolean Equation

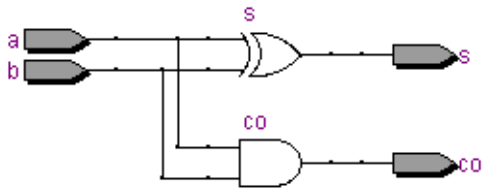
a \ **b**	**0**	**1**
0		
1		

Carry out co = _____

a \ **b**	**0**	**1**
0		
1		

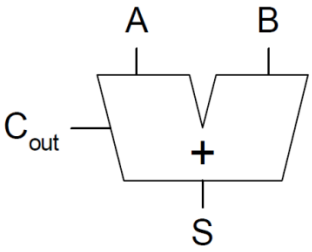
Sum s = _____

- Circuit and Graphical Symbol
  - ✓ Circuit



  - ✓ Symbol

    Symbol 은 일반적으로 아래의 두 개가 많이 사용된다.



2. Full Adder

- Functional Description
  - ✓ 2 개의 1-bit 입력과 1-bit carry in 을 입력으로 받아 sum 과 carry out 을 출력하는 가산기이다. a, b, ci 3 개의 입력을 받아 sum 을 의미하는 S 와 carry out 을 의미하는 co 을 출력한다.

- Truth Table & Karnaugh Map
  - ✓ Truth Table

Input			**Output**	
ci	a	b	co	s
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

- Karnaugh Map and Boolean Equation
  - ✓ About Sum - s

ci \ **ab**	**00**	**01**	**11**	**10**
0				
1				

Sum s = _____

(직접 boolean equation 을 작성하여 본다. – 레포트에 정리)

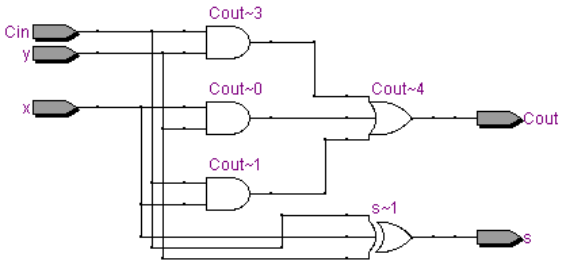
- About Carry Out - co

ci \ **ab**	**00**	**01**	**11**	**10**
0				
1				

Carry out co = _____

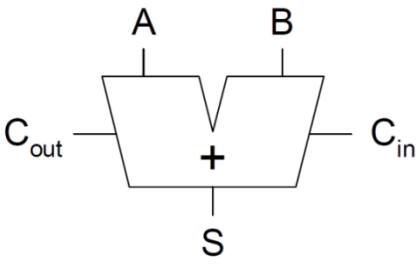
(직접 boolean equation 을 작성하여 본다. – 레포트에 정리)

- Circuit and Graphical Symbol
  - ✓ Circuit



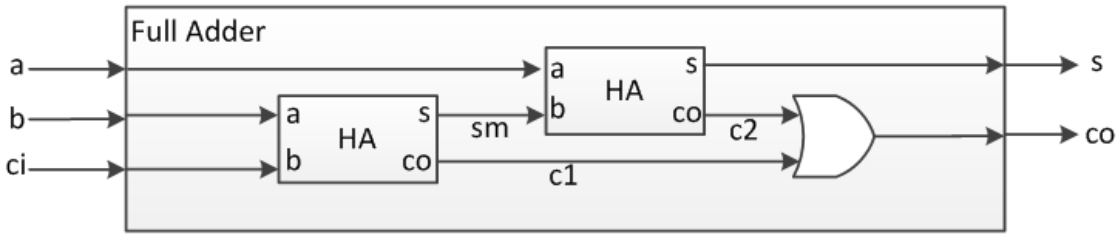
- Graphical Symbol

  Symbol 은 일반적으로 아래의 두 개가 많이 사용된다.



- Structural Description

  Full adder 는 아래의 그림과 같이 half adder 와 or gate 를 사용하여 표현 가능하다.

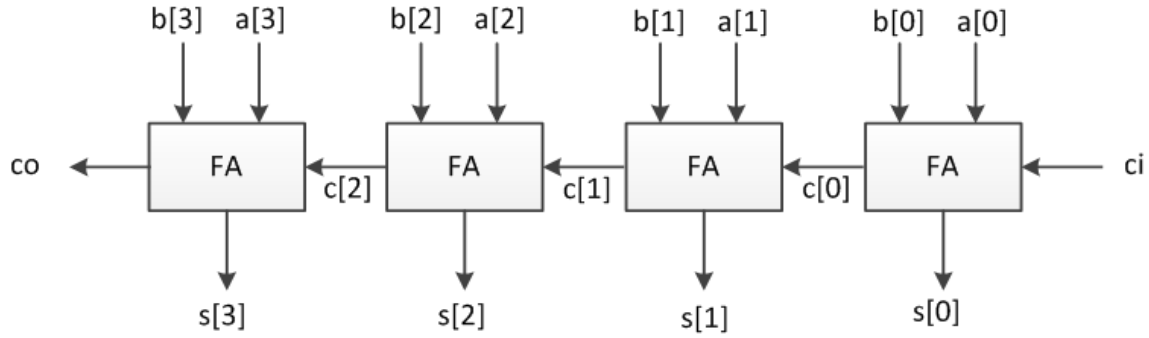


<u>(실습 및 보고서에는 full adder 에 대하여 half adder 를 이용하여 구현하는 것과 바로 gate 를 사용하여 구현하는 것 어느 것을 사용하여도 무관)</u>

3. Ripple Carry Adder

- Functional description
  - ✓ N-bit ripple carry adder 는 n-bit 를 가지는 두 개의 수를 더하기 위한 가장 간단한 형태의 가산기이다. Ripple carry adder 는 더하기 하려는 수의 bit 개수만큼 full adder 를 연결하여 구현한다.
  - ✓ N-bit ripple carry adder 의 worst-case delay 는 각각 1-bit full adder 의 carry delay 를 모두 더한 시간이 지난 후에 결과 값이 나오기 때문에 연산 속도가 느리다.

- Structural description



4. Verilog 구현

- 프로젝트 생성 및 검증에 대한 자세한 내용은 Lab1 을 참고
- Design specification
  - ✓ Module hierarchy description
    - 기술된 top module 과 sub module 의 이름은 표와 **반드시 동일**해야 한다.

구분	이름	설명
Top module	rca4	4 bits ripple carry adder
Sub module	fa	Full adder
Sub module	ha	Half adder

- File configuration
  - ✓ rca4.v – 4 bits ripple carry adder 구현
  - ✓ fa.v – full adder 를 half adder instance 혹은 gate instance 를 이용해 구현
  - ✓ ha.v – half adder 를 gate instance 를 이용해 구현
  - ✓ gates.v
    - 기본 gate 들을 gates.v 에 저장하여 이후의 구현에서는 해당 gate 들을 사용하여 구현 (exclusive or(XOR)의 경우, inverter, and 와 or gate 를 instance 하여 구조적으로 구현)

4.1. Half adder

- Module specification
  - ✓ Module name: ha
  - ✓ File name: ha.v
  - ✓ I/O configuration
    - I/O 는 표와 **반드시 동일**해야 한다.
    - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
    - 다음 표에 나온 wire/reg 는 예시이다.

구분	이름	비트 수	설명
Input	a	1 bit	Input data a
	b	1 bit	Input data b
Output	co	1 bit	Carry out
	s	1 bit	Sum

4.2. Full adder

- Module specification
  - ✓ Module name: fa
  - ✓ File name: fa.v
  - ✓ I/O configuration
    - I/O 는 표와 **반드시 동일**해야 한다.
    - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
    - 다음 표에 나온 wire/reg 는 예시이다.

구분	이름	비트 수	설명
Input	a	1 bit	Input data a
	b	1 bit	Input data b
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	1 bit	Sum
Wire	c1	1 bit	Half adder carry
	c2	1 bit	Half adder carry
	sm	1 bit	Half adder sum

4.3. RCA

- Module specification
  - ✓ Module name: rca4
  - ✓ File name: rca4.v
  - ✓ I/O configuration
    - I/O 는 표와 **반드시 동일**해야 한다.
    - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
    - 다음 표에 나온 wire/reg 는 예시이다.

구분	이름	비트 수	설명
Input	a	4 bits	Input data a
	b	4 bits	Input data b
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	4 bits	Sum
Wire	c	3 bits	Carry wire

4.4. Gates

- 자세한 내용은 컴퓨터공학기초실험 2 강의자료 참고

- Inverter

```
module _inv(a,y);
input a;
output y;
assign y=~a;
endmodule
```

- 2-to-1 nand gate

```
module _nand2(a,b,y);
input a,b;
output y;
assign y=~(a&b);
endmodule
```

- 2-to-1 and gate

```
module _and2(a,b,y);
input a,b;
output y;
assign y=a&b;
endmodule
```

- 2-to-1 or gate

```
module _or2(a,b,y);
input a,b;
output y;
assign y=a|b;
endmodule
```

- 2-to-1 xor gate

```
module _xor2(a,b,y);
input a, b;
output y;
```

Using inverter, and, and or gate

```
endmodule
```

- **gates.v로 저장한다.**

5. Report

- 레포트는 공지사항에 올린 보고서 양식에 맞추어 작성하고, 다음의 사항에 대하여서도 **추가적**으로 작성한다.
  - ✓ 2 의 보수 원리에 대하여 설명한다.
  - ✓ 검증에서 Half adder, full adder 와 4-bits RCA 는 RTL Viewer 와 Flow Summary 를 확인하며 레포트에 삽입한다. 4-bits RCA 의 경우, input 의 수가 9 개이기 때문에 exhaustive verification 이 아닌 directed verification 을 실시한다.
    - Directed verification 을 위하여 작성한 testbench scenario0 를 선택한 이유에 대해서 작성한다.
  - ✓ 검증에서 4-bits RCA 에서 radix 를 decimal 로 할 때와 unsigned 로 할 때의 결과를 비교하고, 해당 결과가 올바른 지 검증하여라.
  - ✓ 결론에 4-bit RCA 를 이용하여 32-bit RCA 를 설계하는 방법에 대하여 논의한다.
- **Source code 제출 시 RCA 프로젝트에 gates.v, ha.v, fa.v, rca4.v, tb_ha.v, tb_fa.v, tb_rca4.v 를 함께 프로젝트 파일에 추가하여 한 폴더에 제출한다.** (즉, ha 와 fa 에 대한 프로젝트는 제출하지 않는다.)

- 채점기준

세부사항		점수	최상	상	중	하	최하
소스코드	Source code 가 잘 작성되었는가? (Structural design 으로 작성되었는가?)	10	10	8	5	3	0
	주석을 적절히 달았는가? **(반드시 영어로 주석 작성)**	20	20	15	10	5	0
설계검증 (보고서)	보고서를 성실히 작성하였는가? (보고서 형식에 맞추어 작성)	30	30	20	10	5	0
	합성결과를 설명하였는가?	10	10	8	5	3	0
	검증을 제대로 수행하였는가? (모든 입력 조합, waveform 설명)	30	30	20	10	5	0
총점		100					

6. Submission

- 제출기한
  - 자세한 제출기한은 KLAS 와 일정을 참고
- 과제 업로드
  - ✓ Source code 와 report 를 같이 ZIP 파일로 압축하여 KLAS(종합정보서비스) 과제 제출에 해당 과제 upload
  - ✓ 업로드 파일명은 (요일#)_(학번)_Assignment_#.zip
    - 요일번호
      - 실습 미수강은 0
      - 월요일 0, 1, 2 교시 1
      - 화요일 0, 1, 2 교시 2
      - 수요일 0, 1, 2 교시 3
    - Ex) 월요일 반 수강, 2019110609, Assignment 1 제출 시 1_2019110609_Assignment_01.zip 으로 제출
  - ✓ Report 명은 (학번)_(요일#)_Assignment_#.pdf
    - 요일 번호는 위의 업로드 파일명과 동일하게 진행
  - ✓ Ex) 수요일 반 수강, 2019110609, Assignment 1 제출 시 3_2019110609_Assignment_01.pdf 으로 제출
  - ✓ Report 는 PDF 로 변환해 제출 (미수행시 감점)
- Source code 압축 시 db, incremental_db, simulation 폴더는 삭제 (미수행시 감점)
- Source code 압축 시 ~.bak 파일 삭제 (미수행시 감점)