Term Project

이름 홍세정
학과 컴퓨터정보공학부
학번 2017202087

I. 서론

문제:

- 3개의 가산기를 SystemC를 이용하여 상위수준 모델링을한다.
- 3개의 가산기는 RCA(ripple carry adder), CLA(carry look ahead adder), CSA(carry select adder)이다.
- 각 가산기의 구조를 이해하고 가산기마다 clock cycle delay를 모델링하여 실제 모듈의 동작을 모방한다.
- 이번 프로젝트에서 회로의 딜레이는 FA(full adder)가 연산하는 시간을 1cycle로 가정한다.
- 또한 SystemC 시뮬레이션의 cycle period는 1ns로 정한다.

하드웨어와 소프트웨어를 통합하여 설계하기 위해 SystemC가 필요하다. 하드웨어나 소프트웨어의 개발이 수월하도록 지원한다. SystemC code로 변환하여 전체 시스템의 동작에 대한 검증을 수행할 수 있다. 또한 하드웨어 부분의 동작을 확인할 수 있다. SoC(System on chip) 설계를 할 수 있다.

II. 본론

Term Project는 SystemC를 이용하여 가산기를 구현하였다. 3개의 가산기 RCA(ripple carry adder), CLA(carry look ahead adder), CSA(carry select adder)에 delay 비교를 하고 분석한다.

프로젝트에서 사용된 파일은 main.cpp, SYSTEM.h, Adder.h, TB.h이 사용되었다.

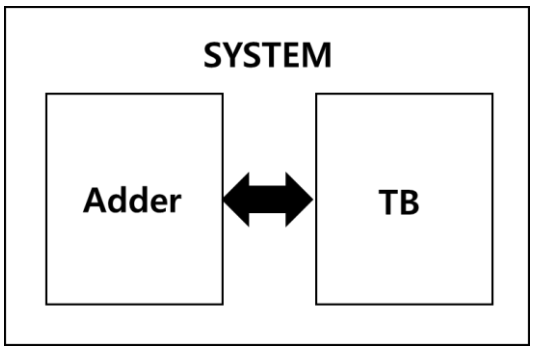


Figure 1. Environment

Figure 1과 같이 나타낼 수 있다.

main에서 SYSTEM을 실행하고 SYSTEM에서 Adder와 TB를 연결하는 역할을 한다.

- Main.cpp

  SYSTEM을 할당하고 수행한다.

  Testbench를 시작한다.

- SYSTEM.h

  Adder와 TB를 연결하는 signal들은 input name에 _sig를 붙여주는 형태로 만들어주었다.

- Adder.h

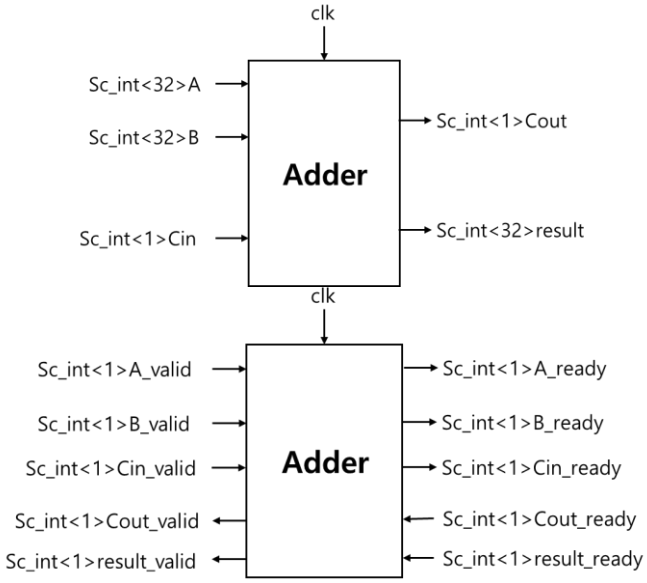
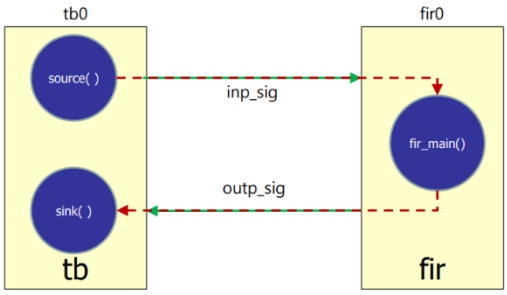


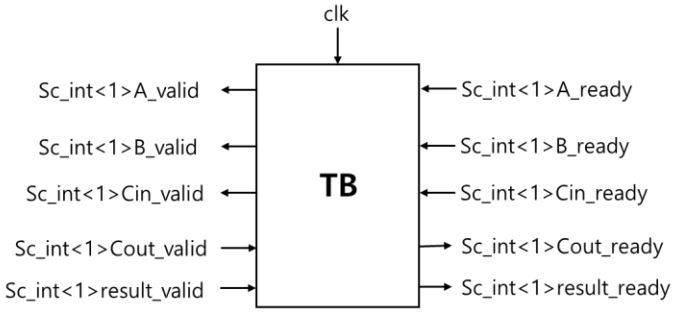
Figure 2. Adder input, output

Figure 2은 Adder의 input과 output을 나타낸 것이다.

Testbench에서 입력을 받아와 adder의 연산을 수행한다.



- TB.h



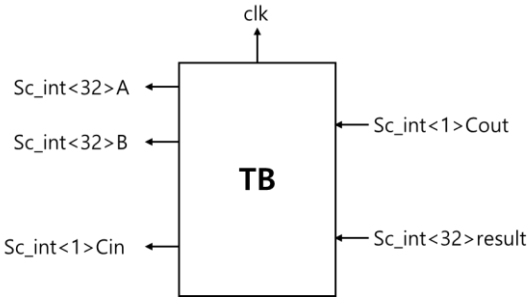


Figure 3. TB input, output

Figure 3은 Adder의 input과 output을 나타낸 것이다.

Adder와 input과 output이 반대로 들어간다. 즉, Adder의 input이 TB의 output, Adder의 output이 TB의 intput이 된다. Clock cycle은 Adder와 TB 모두 input으로 들어간다.

Symbols	Discription
Tg	Gate delay
$Txor = 2Tg$	XOR delay
$TFA = 2TXOR$	Full adder delay
$TMUX = 2Tg$	MUX delay
$Pclock = TFA$	Clock period

Table 1. Time symbols

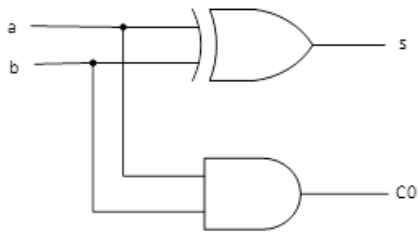
Delay는 다음과 같이 나타낼 수 있다.

3개의 가산기의 구조를 설명한다.

가산기는 4bit를 예로 들어 설명한다.

RCA(Ripple Carry Adder)

두 수를 더하는 가산기 중 가장 기본에 되는 가산기는 ripple carry adder이다.



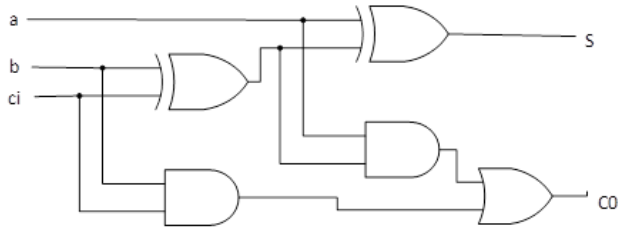


Figure 4. half adder, full adder

Figure 4은 RCA에서 사용되는 half adder와 full adder의 회로이다. Full adder는 half adder 두 개가 합쳐진 형태이다.

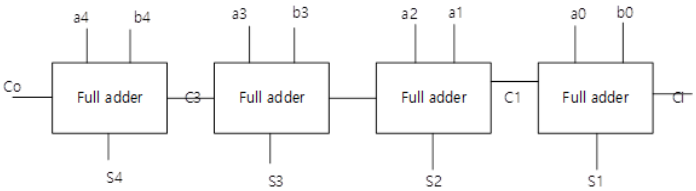


Figure 5. Rca-4bit adder

Figure 5은 RCA 4bit adder이다.

n개의 가산기, 감산기를 병렬로 연결하면 N비트의 2진 연산이 가능하다. 복수개의 비트들로 구성된 2진수 2개를 더해 결과를 출력하는 조합회로로 전가산기들을 차례로 연결하여 아랫단의 자리올림 출력이 윗 단의 자리올림 입력으로 들어가도록 회로를 구성한다. n개의 전가산기를 연결하면 n비트로 구성된 2개의 2진수를 더할 수 있다. 단점으로는 윗 단이 입력으로 받아 계산을 할 수 있으므로 전체 계산시간이 많이 걸린다는 단점이 있다. n-bits RCA는 full adder를 n개 연결시켜 만드는 가산기인데 여기서 n이 커질수록 full adder 연결사이에서 delay가 발생하게 되어 연산속도가 더 느려 진다.

Fulladder의 delay는 $TFA = 2TXOR$이다. RCA 32bit는 Full adder가 총 32개가 연결되어 있는 회로이므로 $32TFA = 64TXOR = 128Tg$를 가진다. Cout은 128Tg를 가지고 result는 4bit $31 \times 2 \times Tg + TXOR + TXOR = 62Tg + 2Tg + 2Tg = 66Tg =$ 13.75cycle이다. 따라서 14cycle에 걸리게 된다.

CLA(Carry Look ahead Adder)

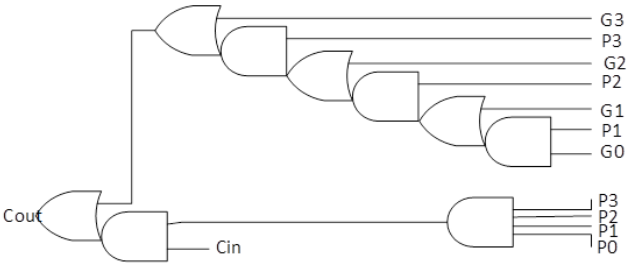


Figure 6. carry look ahead adder

올림 수를 미리 계산하는 가산기이다. CLA(carry look ahead adder)는 carry를 미리 보는 가산기인 ripple carry adder를 어떻게 하면 더 효율적으로 바꿀 수 있는가에 대한 고민에서 탄생한 회로이다.

32bit CLA를 만들 때 32비트를 통째로 만들지 않고 작은 비트를 여러 개 이어붙여 만들게 된다. CLA는 오버헤드가 크기 때문에 오버헤드를 줄이기 위해 4bit CLA를 8개를 이어 붙여 만든다.

$T4bit_CLA = Tpg + Tpgblock + TCin + TFA = 1Tg + 6Tg + 0 + 4Tg = 11Tg = 2.75\ clock\ cycle$로 4-bit CLA의 cycle을 계산할 수 있다.

32bit는 계산하였을 때 $Tpg + Tpgblock + TCin + TFA$ = 1Tg + 64Tg + 4Tg = 69Tg = 14.3cycle이다. 따라서 총 15cycle이 걸리게 된다.

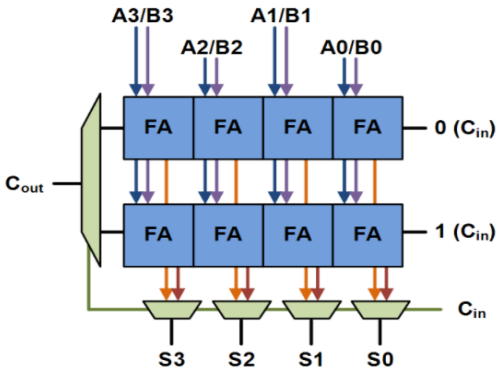
Bit가 늘어날수록 계산이 많아지기 때문에 cycle이 RCA보다 줄어들지 않는다.

32-bits CLA와 32-bits RCA의 크기 속도 비교

CLA는 RCA가 계산이 완료될 때까지의 시간이 많이 걸리는 단점을 보완하기 위해 모든 올림수가 동시에 구해져 계산시간을 단축시키는 가산기이다. n-bits RCA는 full adder를 n개 연결시켜 만드는 가산기인데 여기서 n이 커질수록 full adder 연결사이에서 delay가 발생하게 되어 연산속도가 더 느려 진다. CLA는 Carry만 계산해주는 carry look-ahead block이 존재하면서 carry를 앞서 보면서 계산할 수 있어 속도가 빨라지게 된다.

CSA(Carry Select Adder)

계산 시간을 단축하기 위해 생각해낸 덧셈기다.



Carry 입력 값은 '0'이나 '1' 중 하나이므로 0'과 '1'을 각각 carry 입력으로 가정한 별도의 덧셈을 미리 수행하고 실제 carry 입력에 따라 가정에 맞는 덧셈 결과 선택한다.

장점: 계산 속도를 빠르게 할 수 있다.

단점: 상위비트로 갈수로 경우의 수가 2를 곱한 것만큼 증가한다.

III. 실험결과

```
A: 0000_0000_0000_0000_0000_0000_0000_0000
B: 1111_1111_1111_1111_1111_1111_1111_1111
Cin: 1
Cout: 0
File_Result: 0000_0000_0000_0000_0000_0000_0000_0000
File_Cout: 1
A: 0000_0000_0000_0000_0000_0000_0000_0000
B: 1111_1111_1111_1111_1111_1111_1111_1111
Cin: 0
Cout: 0
File_Result: 1111_1111_1111_1111_1111_1111_1111_1111
File_Cout: 0
A: 0101_0101_0101_0101_0101_0101_0101_0101
B: 1010_1010_1010_1010_1010_1010_1010_1010
Cin: 0
Cout: 0
File_Result: 1111_1111_1111_1111_1111_1111_1111_1111
File_Cout: 0
A: 0000_1111_0000_1111_0000_1111_0000_1111
B: 0000_1111_0000_1111_0000_1111_0000_1111
Cin: 0
Cout: 0
File_Result: 0001_1110_0001_1110_0001_1110_0001_1110
File_Cout: 0
A: 0000_1111_0000_1111_0000_1111_0000_1111
B: 0000_1111_0000_1111_0000_1111_0000_1111
Cin: 1
Cout: 0
File_Result: 0001_1110_0001_1110_0001_1110_0001_1111
File_Cout: 0
```

A와 B, Cin을 읽어온 후 cout과 result 값을 출력하는 것을 확인할 수 있다.

또한 File_Result, File Cout 값과 Result, Cout값과 값이 맞는지 확인해 보았다.

IV. 결론

3개의 가산기를 SystemC를 이용하여 상위수준 모델링을 할 수 있었다.

이번 프로젝트에서 회로의 딜레이를 이해하고 각 32bit 가산기의 딜레이를 구할 수 있었다.

참고문헌

- 2020년 2학기 컴퓨터정보공학부 하드웨어 소프트웨어통합설계 강의 자료
- 2018년 2학기 컴퓨터정보공학부 디지털논리회로2 강의 자료
- 2진법 계산기

  https://ko.calcuworld.com/%EC%88%98%ED%95%99/2%EC%A7%84%EB%B2%95-%EA%B3%84%EC%82%B0%EA%B8%B0/
- https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M05_arith1.pdf
- CSA그림

  https://en.wikipedia.org/wiki/File:Carry-select-adder-detailed-block.png