4-Bit Adder/Subtractor & Arithmetic Logic Unit

2017-13846 전기정보공학부 양준엽

1. 실험 목표

논리회로에서 구현되는 양수와 음수의 표현 방법과 처리 방식에 대해서 이해하고, 덧셈기/뺄셈기의 구성과 동작 원리를 이해하고 직접 구성해 본다. 또한, 덧셈과 뺄셈을 동시에 수행 가능한 단일 공유 Adder/Subtractor을 구성해보고, CPU의 핵심 모듈인 산술 논리 장치 구조를 이해한다.

1. 실험 이론

디지털 시스템에서는 한번에 처리하는 데이터의 단위가 4비트, 8비트, 16비트 32 비트 등으로 임의의 크기의 덧셈 뺄셈을 수행하는 경우, n-비트의 덧셈을 지원하는 Adder은 1 비트 Adder n개를 연결하여 간단하게 설계할 수 있다. 가장 낮은 비트 자리에서는 아래 단에서 올라오는 올림수가 없기 때문에 Half Adder 을 사용하고, 다음 비트 자리부터 모든 비트 위치에서는 Full Adder을 연결하여 구성한다. 뺄셈이 가능하도록 논리회로를 구성하기 위해서는 음수에 대한 표현을 고려하기 위해, 0~7의 경우 앞에 0을 붙여 양수로 인식하게 하고, -8~-1은 맨 앞에 1을 붙여 2의 보수로 음수를 표현하게끔 한다. 2의 보수 표현을 사용하면 같은 부호의 숫자를 더할 때 한계치보다 더 크거나 작아지는 오류가 발생할 수 있는데, 이를 오버플로우라고 한다. 이를 방지하게 위해서는 최상위 비트의 2개의 올림수를 조사하여 두 개의 값이 서로 같은 경우에는 정상으로, 값이 다른 경우에는 오버플로우로 인식 하게 한다. 따라서 오버플로우를 체크하기 위하여 최상위 2비트는 XOR에 연결한다.

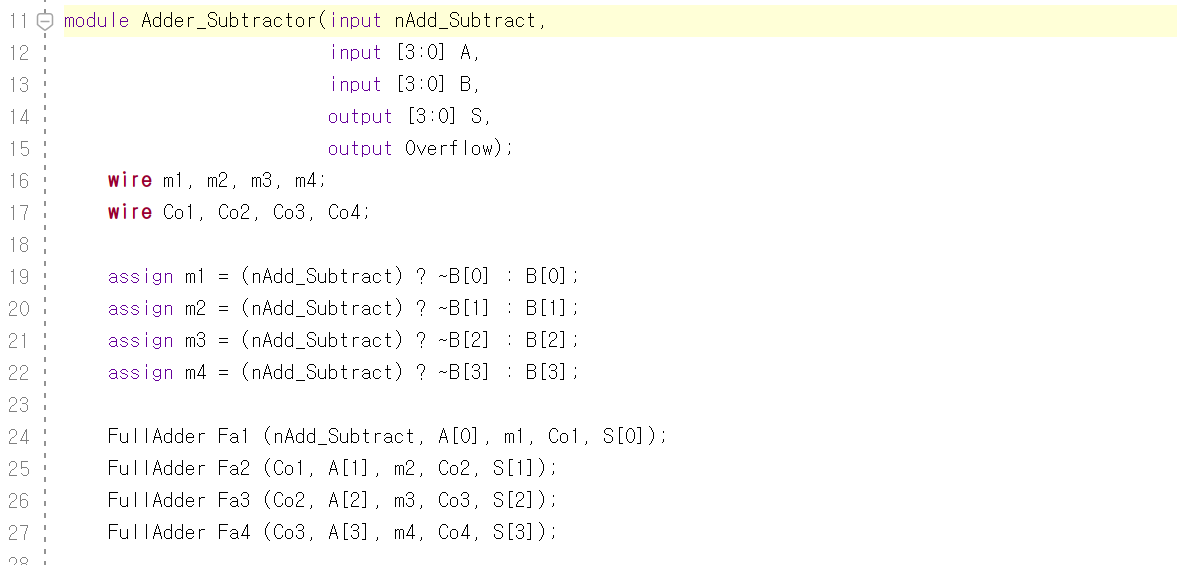
디지털 시스템에서 가장 핵심이 되는 모듈은 프로그램을 구성하는 명령어를 수행하는 CPU이다. 두개의 입력 A,B에 대하여 NOT 연산을 수행한다면 A,A’,B,B’ 에 해당하는 4가지 출력을 제공하며, AND, OR, NOT 을 지원하고 서로 조합한다면 16가지의 논리연산 결과 지원이 가능하다. 속도보다 최적화된 면적에 비중을 두어 설계한다면 사용빈도가 적거나 거의 없는 연산을 포함하지 않아도 된다.

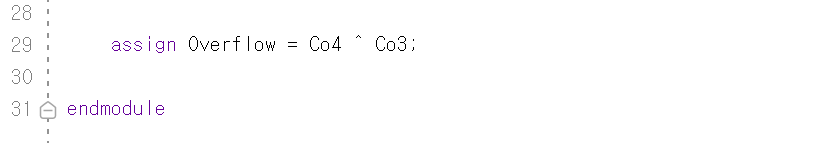
니블단위의 두 입력 A와 B 를 더하거나 빼는 연산과 입력된 수의 보수를 구하는 연산을 기본으로 한다. 이 연산만으로 반복수행을 통해 원하는 연산을 모두 수용할 수 있다.

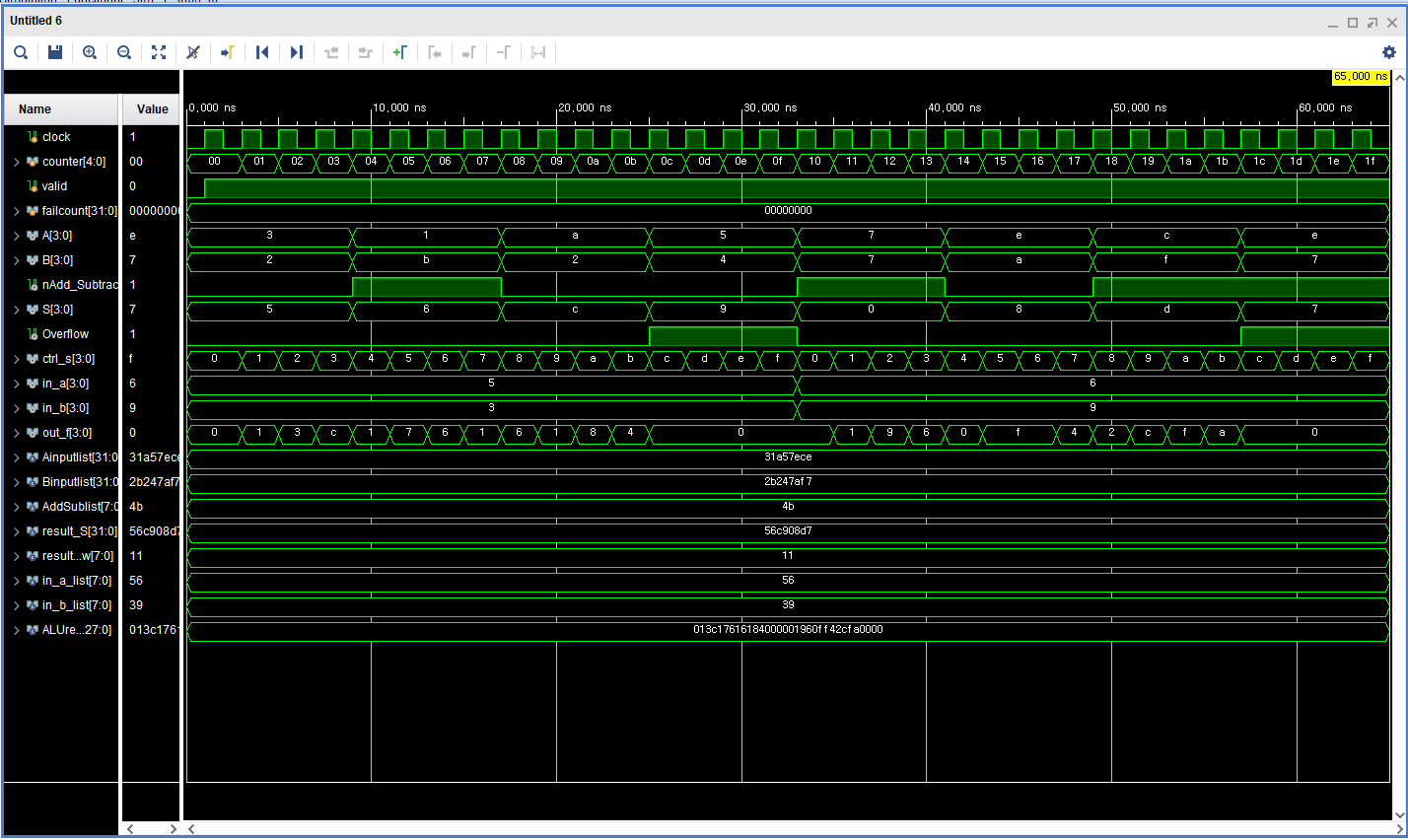
1. 사용 장비

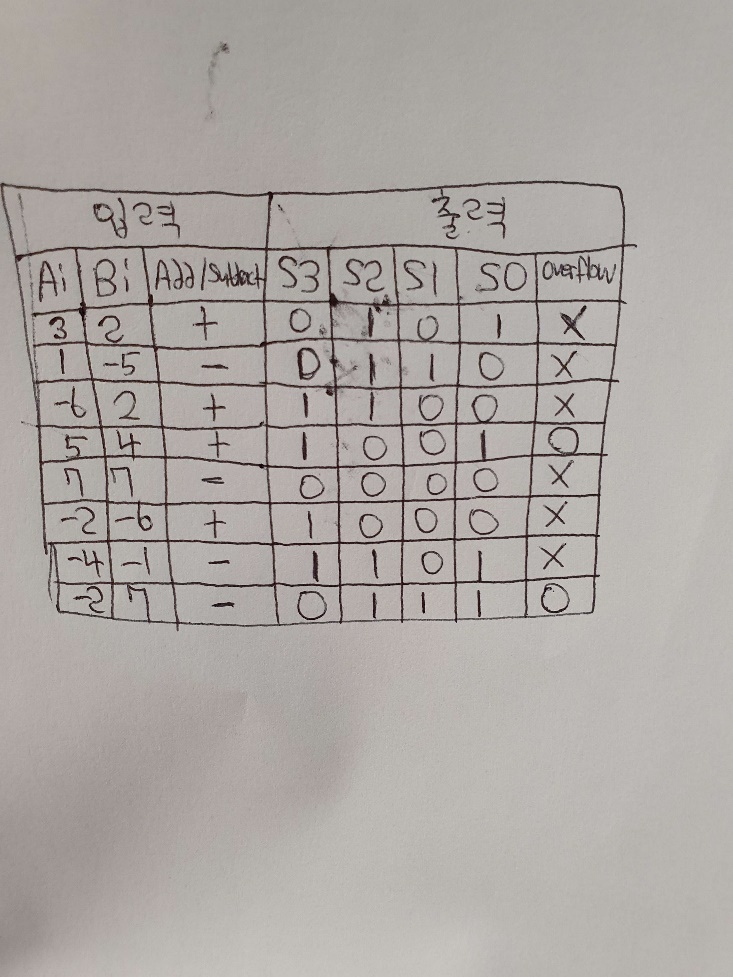
PC, Xilinx Vivado 소프트웨어

1. 토론 및 고찰사항
2. (1)



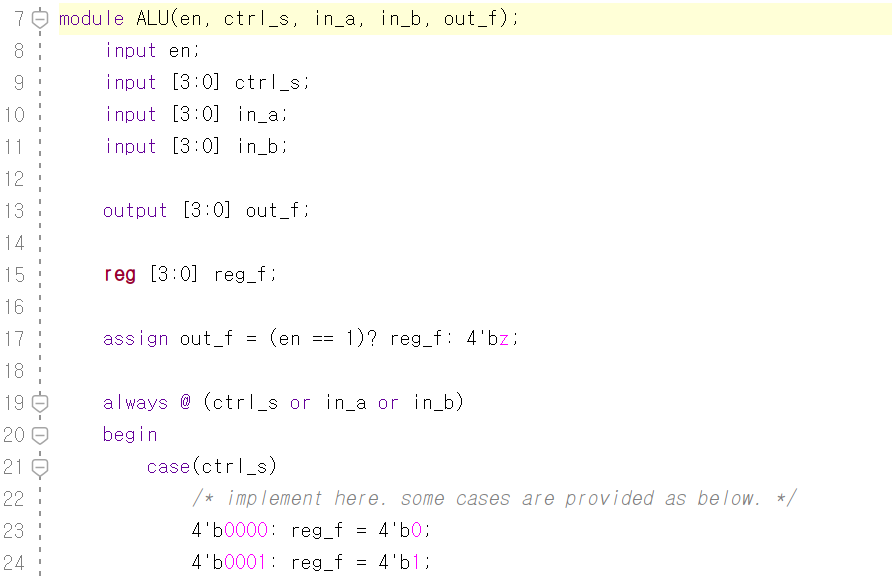


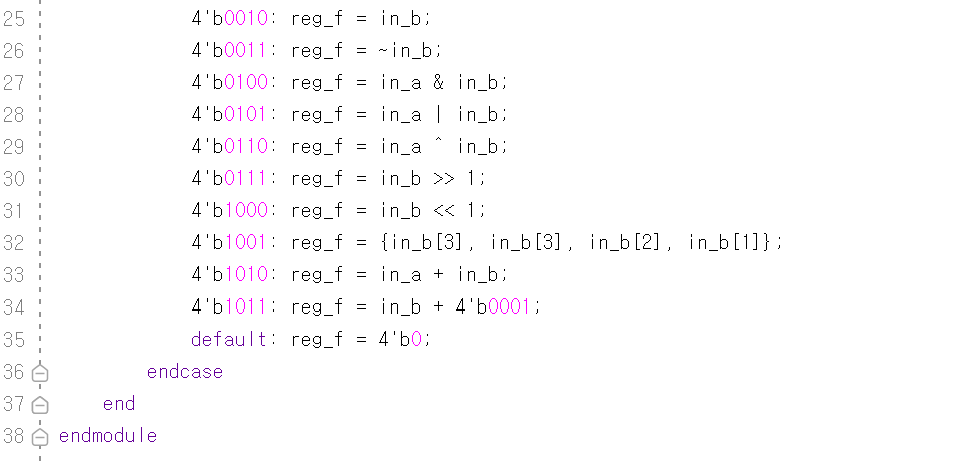




5+4, -2 -7을 할 때 결과 9,-9가 나와야 하는데 양수는 최댓값이 7, 음수는 최솟값이 -8이라 오버플로우가 발생한다. 오버플로우도 잘 작동했고, 나머지 덧셈, 뺄셈 다 제대로 작동했다.

(2)





위의 시뮬레이션 결과를 보면 failcount가 전부 0으로 verlilog가 잘 짜여있음을 알 수 있다. 4’b1001 경우 위의 2개와 달리 >>> 기호가 작동하지 않아 직접 Arithmetic Shift Right을 수행하는 코딩을 해 작성하였다.

(3) 예를 들어 15x 13를 한다고 했을 때 (1111) x (1101) 일 때, 한 곳(A)에는 초기값 0000, P에는 1111을, Q에는 1101을 저장한다. P레지의 맨 오른쪽 1과 Q레지의 1101을 곱하고 그 값을 초기값 A레지에 더한다. 그다음, AP 레지를 Logical Right Shift 한다. 마찬가지로, P레지의 그다음 오른쪽1, 그다음, 그다음에 대해 똑 같은 작업을 진행하면 정수형 산술 곱셈이 가능하다.

1. 참조

Etl.snu.ac.kr/Lab06\_4-bit\_Adder\_Subtractor.pdf