Ripple carry adder와 Carry select adder

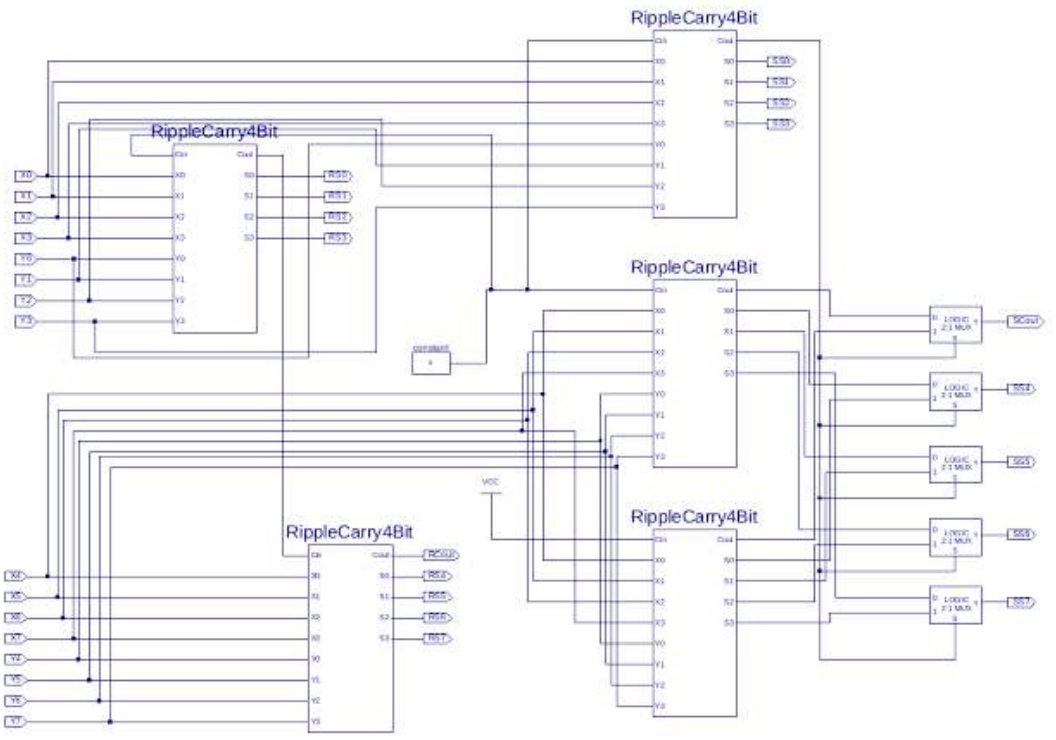
2017-10-11
2016-17101 김종범
2016-17274 이도윤

실험4. 8-Bit Carry Select Adder 구현

8-Bit ripple carry adder와 8-Bit carry select adder 구현 및 동작을 비교해보았다.

8-Bit ripple carry adder는 4-Bit ripple carry adder 두 개를 이용하여 만들 수 있다. X[7:0]와 Y[7:0]을 더한다고 할 때, X[3:0]과 Y[3:0]을 한 4-Bit ripple carry adder에 연결해주고, 첫 번째의 4-Bit ripple carry adder의 Cout과 X[7:4]와 Y[7:4]을 다른 4-Bit ripple carry adder에 연결하면 된다. 자세한 구현은 아래 회로도의 왼쪽 부분을 참고하면 된다.

8-Bit carry select adder는 3개의 4-Bit ripple carry adder을 사용하여 만들 수 있다. 첫 번째 4-Bit ripple carry adder는 8-Bit ripple carry adder를 만들 때 첫 번째 4-Bit ripple carry adder과 같은 방법으로 만들면 된다. 두 번째, 세 번째는 각각 첫 번째 4-Bit ripple carry의 Cout 값이 0일 때의 상황, 1일 때의 상황을 각각 연결해서 만들어주면 된다. (즉, 예상한 Cout과 X[7:4], Y[7:4]를 연결하여 만들어진다.) 그 후, 실제 Cout 결과에 따라서 2:1 Mux를 이용하여 결과를 반환하면 된다. 자세한 구현은 아래 회로도의 오른쪽 부분을 참고하면 된다.



추가적으로 8-Bit ripple carry adder와 8-Bit carry select adder에 같은 입력을 동시에 인가하였을 때 발생하는 시뮬레이션 결과를 아래와 같이 첨부한다. 왼쪽 부분에서 Xn이 X의 n번째 bit이고 Yn이 Y의 n번째 bit를 의미한다. 또한 RSn은 Ripple Carry adder의 n번째 bit의 결과값을 의미하고 SSn은 Ripple Carry adder의 n번째 bit의 결과값을 의미한다.

8-Bit ripple carry adder와 8-Bit carry select adder를 비교하였을 때, 대부분의 입력값에 대해서 carry select adder가 delay가 적음을 확인할 수 있었다.

8-Bit ripple carry adder에서 가장 시간 지연이 길 것으로 예상되는 case들은 “00000001 + 11111111”, “11010001 + 00101111”등과 같이, X[0]+Y[0] = 2이고, X[i]+Y[i] = 1 (for i in [1, 7])인 경우라고 생각하였다. 왜냐하면, 최종 Cout값이 계산될 때 최대로 지나는 회로의 개수가 $3+2\times7=17$이고 (각각, 첫 FullAdder : 3, 다른 7개의 FullAdder : 2), 위와 같은 Case가 회로를 10개를 지나야 Cout의 값이 확정되는 경우이기 때문이다. 해당 Case에 대한 두 예시의 시뮬레이션 결과를 아래와 같이 첨부한다. 실제로 RSn의 값을 볼 때 n이 커질수록 Delay가 생기는 Propagation Delay를 관찰할 수 있었다.