10주차 결과보고서

전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희

**1.**

**텍스트, 영수증, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 사각형, 텍스트, 컴퓨터이(가) 표시된 사진

자동 생성된 설명**

**도표, 텍스트이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 4-bit Binary Parallel Adder의 test bench, design source, simulation, schematic diagram이다. 4-bit Binary Parallel Adder을 구현하기 위해서 입력 값들을 input [3:0] A, B로 배열로 나타냈고 출력 값도 output [3:0] S와 output [3:0] C로 배열로 나타냈다. 덧셈 연산을 구현할 때는 S=C⊕A⊕B로, C는 C=AB+C(A⊕B)로 나타냈다. 이전 연산에서 나오는 carry out을 다음 연산의 input로 쓰이는 ripple carry adder로 각각의 A(A[0], A[1], A[2], A[3])와 B(B[0], B[1], B[2], B[3]), Cin의 연산 결과로 C(C[0], C[1], C[2], C[3])과 S(S[0], S[1], S[2], S[3])을 얻는데 최종 결과물은 S배열의 값과 최종 carry out인 C[3]이다. Simulation 결과를 살펴보면 다음과 같다. 예를 들어 1001(A[3]A[2]A[1]A[0]) + 0011(B[3]B[2]B[1]B[0]) + 1(Cin)로 입력 값이 들어 왔다면 1(A[0]) + 1(B[0]) + 1(Cin) = 1(S[0]) 이고 C[0] = 1이다. 0(A[1]) + 1(B[1]) + 1(C[0]) = 0(S[1]) 이고 C[1] = 1이고, 0(A[2]) + 0(B[2]) + 1(C[1]) = 1(S[2]) 이고 C[2] = 0이고, 1(A[3]) + 0(B[3]) + 0(C[2]) = 1(S[3]) 이고 C[3] = 0이다. 따라서 최종 carry out값은 0이 되고 S배열은 1101(S[3]S[2]S[1]S[0])의 값을 갖게 된다. 이를 FPGA의 동작 결과와 비교했을 때 FPGA가 제대로 동작함을 확인할 수 있다.

**2.**

**텍스트, 영수증, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명**

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

**도표, 텍스트, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 4-bit Binary Parallel Subtractor의 test bench, design source, simulation, schematic diagram이다. 4-bit Binary Parallel Subtractor을 구현하기 위해서 입력 값들을 input [3:0] A, B로 배열로 나타냈고 출력 값도 output [3:0] D와 output [3:0] b로 배열로 나타냈다. 뺄셈 연산을 구현할 때는 D=b⊕A⊕B로, b는 b=A’B+b(A⊕B)’로 나타냈다. 이전 연산에서 나오는 borrow out을 다음 연산의 input로 들어가고, 각각의 A(A[0], A[1], A[2], A[3])와 B(B[0], B[1], B[2], B[3]), bin의 연산 결과로 D(D[0], D[1], D[2], D[3])과 b(b[0], b[1], b[2], b[3])을 얻는데 최종 결과물은 D배열의 값과 최종 borrow out인 b[3]이다. Simulation 결과를 살펴보면 다음과 같다. 예를 들어 1101(A[3]A[2]A[1]A[0]) - 0011(B[3]B[2]B[1]B[0]) - 0(bin)로 입력 값이 들어 왔다면 1(A[0]) - 1(B[0]) - 0(bin) = 0(D[0]) 이고 b[0] = 0이다. 0(A[1]) - 1(B[1]) - 0(b[0]) = 1(D[1]) 이고 b[1] = 1이고, 1(A[2]) - 0(B[2]) - 1(b[1]) = 0(D[2]) 이고 b[2] = 0이고, 1(A[3]) - 0(B[3]) - 0(b[2]) = 1(D[3]) 이고 b[3] = 0이다. 따라서 최종 borrow out값은 0이 되고 D배열은 1010(D[3]D[2]D[1]D[0])의 값을 갖게 된다. 이를 FPGA의 동작 결과와 비교했을 때 FPGA가 제대로 동작함을 확인할 수 있다.

**3.**

**텍스트, 영수증, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

**도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 BCD Adder의 test bench, design source, simulation, schematic diagram이다. 이를 구현할 때는 앞서 구현한 4-bit binary parallel adder을 이용하여 구현해주었고, 연산 결과가 9보다 클 때에는 0110을 더하여 결과 값을 얻도록 해주었다. 연산 결과가 9보다 클 때는 Cout이 1이 된다. 입력 값으로는 input [3:0] A, B로 배열과 Cin로 나타냈고 출력 값도 output [7:0] S와 output [6:0] C로 배열과 K, Cout로 나타냈다. Cout을 나타내기 위해서 S[3]S[2]+S[3]S[1]을 포함시켜야 하는데 이 두 값이 다 0의 값을 가져도 overflow가 1일 때 Cout 값도 1이 되어야 하므로 최종 Cout 값은 K+ S[3]S[2]+S[3]S[1]로 나타냈다. A(A[3], A[2], A[1], A[0])과 B(B[3], B[2], B[1], B[0])의 덧셈 결과는 S(S[3], S[2], S[1], S[0])이고, 이에 대한 각각의 carry out은 C(C[2], C[1], C[0])과 이 연산에서의 최종 carry out 값인 K이다. 연산 값이 9가 넘을 경우 0110을 더해야 하므로 2개의 4-bit binary parallel adder가 필요한데, 이를 계산하기 위해서 출력 값으론 S(S[7], S[6], S[5], S[4])와 이에 대한 각각의 carry out 값인 C(C[6], C[5], C[4], C[3]) 배열을 사용하였고, 입력 값으론 S(S[3], S[2], S[1], S[0])과 Cout, 0을 사용하였다. 만약 연산 결과 값이 9를 넘지 않아서 Cout값이 0이라면 최종 입력 값으론 0000과 S[3]S[2]S[1]S[0]이 들어가서 결과 값은 S[3]S[2]S[1]S[0]와 동일하게 나올 것이고 결과 값이 9보다 크다면 Cout 값이 1이 되면 최종 입력 값으론 0110과 S[3]S[2]S[1]S[0]이 들어가서 두 입력 값에 대한 덧셈 연산 결과 값이 나올 것이다. Simulation 결과를 살펴보면 다음과 같다. 예를 들어 1001(A[3]A[2]A[1]A[0])(=9) + 0011(B[3]B[2]B[1]B[0])(=3)로 입력 값이 들어 왔다면 먼저 1(A[0]) + 1(B[0]) + 0(Cin) = 0(S[0]) 이고 C[0] = 1이다. 0(A[1]) + 1(B[1]) + 1(C[0]) = 0(S[1]) 이고 C[1] = 1이고, 0(A[2]) + 0(B[2]) + 1(C[1]) = 1(S[2]) 이고 C[2] = 0이고, 1(A[3]) + 0(B[3]) + 0(C[2]) = 1(S[3]) 이고 K = 0이다. S배열은 1100(S[3]S[2]S[1]S[0])의 값을 갖게 된다. 이때 S[3]와 S[2]가 둘 다 1의 값을 가지므로 최종 Cout 값은 1이 되고 이는 두 번째 4-bit binary parallel adder의 입력 값으로 들어가서 0110 + 1100(S[3]S[2]S[1]S[0])을 수행하게 된다. 수행 과정은 전과 똑같고, 결과 값은 0010(S[7]S[6]S[5]S[4])가 된다. 이 때 Cout 값인 1은 십의 자릿수를 의미하므로 최종 결과 값인 1 0010은 9+3 = 12를 나타낸다. 이를 FPGA의 동작 결과와 비교했을 때 FPGA가 제대로 동작함을 확인할 수 있다.

**4.**

이번 주차 실험에서는 4-bit binary parallel adder, 4-bit binary parallel subtractor, BCD adder을 Verilog로 구현해보고 simulation 결과와 FPGA의 동작을 확인해보았다. Verilog로 구현할 때 입력 값과 출력 값에서 배열을 사용하여 보다 편리하게 코드를 작성할 수 있었고, BCD adder을 구현할 때 2개의 4-bit binary parallel adder을 사용하여 덧셈 연산 값이 9를 넘지 않을 때는 Cout이 0고, 출력 값으로는 첫번째 4-bit binary parallel adder의 결과 값과 똑같이 0000에서 1001사이의 값으로 나오지만 덧셈 연산 값이 9를 넘어갈 때는 Cout이 1이 되고, 출력 값으로 0000에서 1001사이의 값으로 나오는 것을 볼 수 있었다.

**5.**

집적 회로를 일컫는 IC(Integration circuit)은 반도체 물질들로 구성된 칩에서 전자회로의 집합이다. 여러 전자기기들의 핵심 부품으로서 칩을 구성하는 트랜지스터가 몇 개인지에 따라 MSI, SSI, VLSI 그리고 LSI로 나뉜다. 먼저 MSI는 중간 규모 집적회로로 Medium-Scale Integration이다. 수백개의 트랜지스터로 구성되어 있으며 병렬 가산회로 등에 사용된다. SSI는 소규모 집적 회로로 Small-Scale Integration이고 수십개의 트랜지스터로 구성되어 있다. VLSI는 초고밀도 집적 회로로 Very-Large-Scale-Integration인데, 수십만개에서 수십억개의 트랜지스터로 구성되어 있다. 마지막으로 LSI 대규모 집적 회로로 Large-Scale Integration이고 수만개의 레지스터로 구성되어 있다.