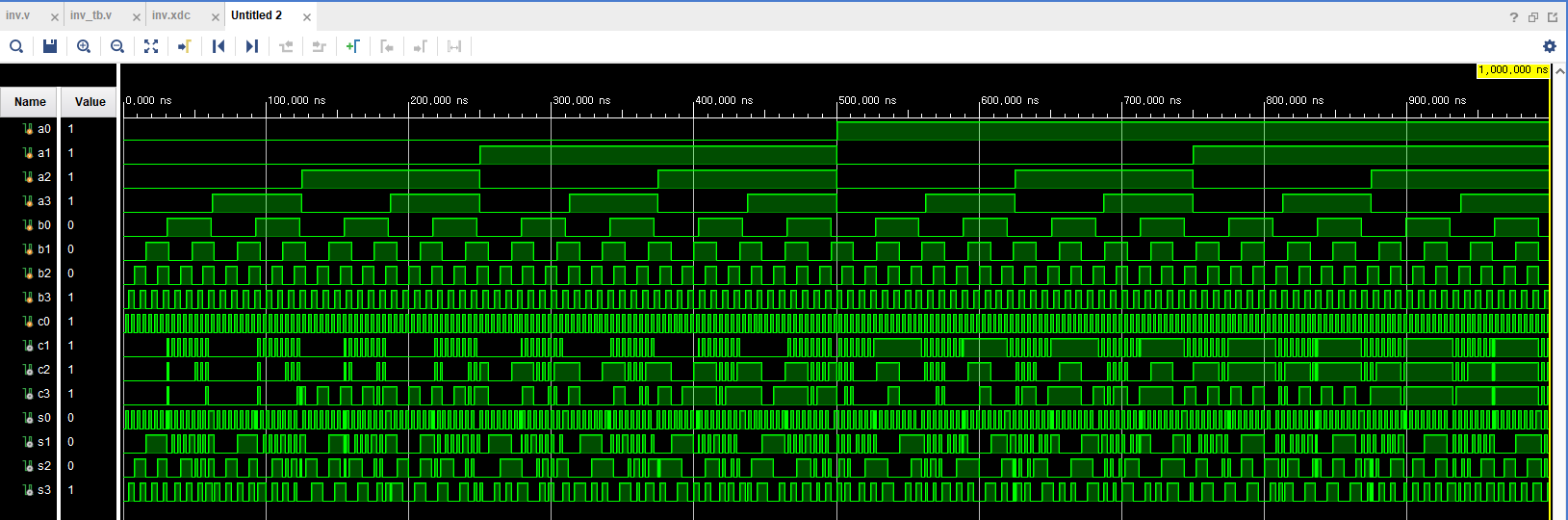
10주차 결과보고서

전공 : 컴퓨터공학과 학년 : 2학년 학번 : 20201597 이름 : 신동준

1. **4bit Binary Parallel Adder 의 결과 및 Simulation 과정에 대해서 설명하시오.**

(verilog source, 출력 예시, 과정 상세히 적을것)

|  |
| --- |
| `timescale 1ns / 1ps  module inv(  input a0,  input a1,  input a2,  input a3,  input b0,  input b1,  input b2,  input b3,  input c0,  output c1,  output c2,  output c3,  output c4,  output s0,  output s1,  output s2,  output s3  );  assign s0 = (a0 ^ b0) ^ c0;  assign c1 = (a0 & b0) | (c0 & (a0 ^ b0));    assign s1 = (a1 ^ b1) ^ c1;  assign c2 = (a1 & b1) | (c1 & (a1 ^ b1));    assign s2 = (a2 ^ b2) ^ c2;  assign c3 = (a2 & b2) | (c2 & (a2 ^ b2));    assign s3 = (a3 ^ b3) ^ c3;  assign c4 = (a3 & b3) | (c3 & (a3 ^ b3));  endmodule |



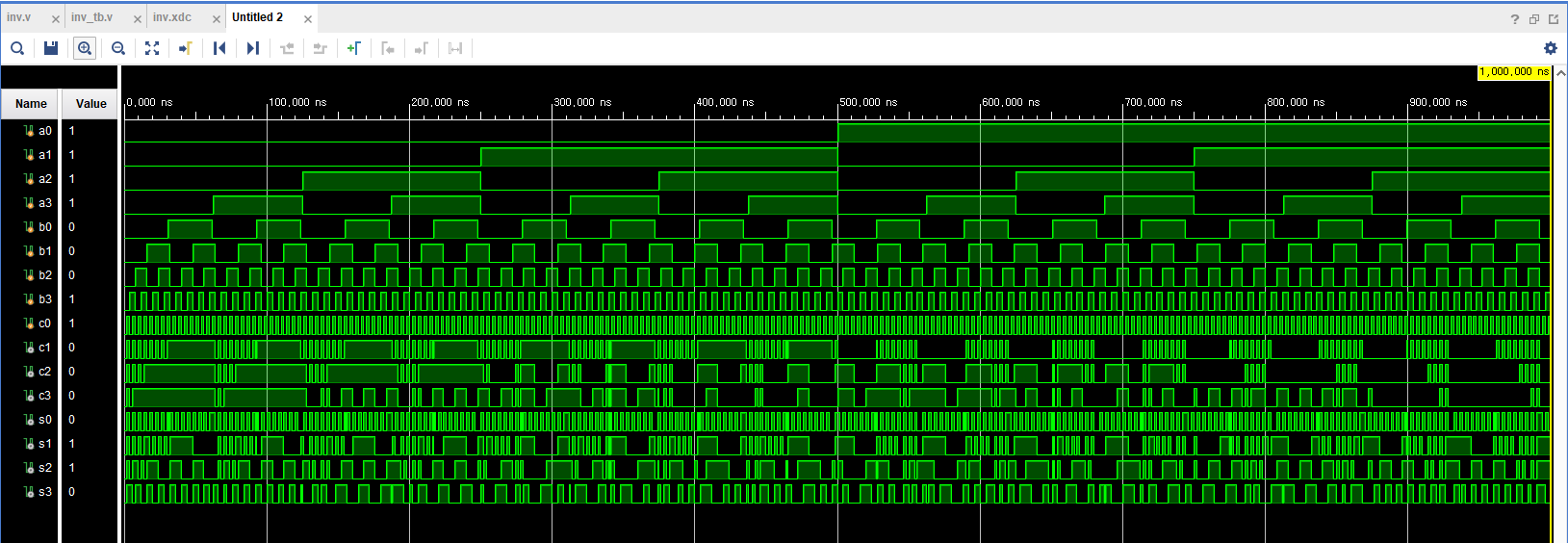
Full Adder를 sn = (an ^ bn) ^ cn; c(n+1) = (an & bn) | (cn & (an ^ bn)); 형태로 구성하고, 각각을 병렬로 연결하였다. 코드 상 n이 점화식 형태로 연결되며 full adder의 Cout이 다음 Full Adder의 Cin으로 들어오는 방식으로 4개가 연결된다.

최초 Full Adder의 Cin, a,b,c,d를 input으로 받고, 마지막 Full Adder의 Cout, a, b, c, d가 output이 되는 구조이다. Decimal로 15까지는 정상출력하고 이후는 over flow 되어 carry out 출력한다.

1. **4bit Binary Parallel Subtractor 의 결과 및 Simulation 과정에 대해서 설명하시오.**

(verilog source, 출력 예시, 과정 상세히 적을것)

|  |
| --- |
| `timescale 1ns / 1ps  module inv(  input a0,  input a1,  input a2,  input a3,  input b0,  input b1,  input b2,  input b3,  input c0,  output c1,  output c2,  output c3,  output c4,  output s0,  output s1,  output s2,  output s3  );  assign s0 = (a0 ^ b0) ^ c0;  assign c1 = (~a0 & b0) | (c0 & ~(a0 ^ b0));    assign s1 = (a1 ^ b1) ^ c1;  assign c2 = (~a1 & b1) | (c1 & ~(a1 ^ b1));    assign s2 = (a2 ^ b2) ^ c2;  assign c3 = (~a2 & b2) | (c2 & ~(a2 ^ b2));    assign s3 = (a3 ^ b3) ^ c3;  assign c4 = (~a3 & b3) | (c3 & ~(a3 ^ b3));  endmodule |



Full Subtractor를 sn = (an ^ bn) ^ cn; c(n+1) = (~an & bn) | (cn & ~(an ^ bn)); 형태로 구성하고, 각각을 병렬로 연결하였다. 코드 상 n이 점화식 형태로 연결되며 full Subtractor의 Cout이 다음 Full Subtractor의 Cin으로 들어오는 방식으로 4개가 연결된다.

최초 Full Subtractor의 Cin, a,b,c,d를 input으로 받고, 마지막 Full Subtractor의 Cout, a, b, c, d가 output이 되는 구조이다. Decimal로 0~15까지는 정상출력하고 이후는 over flow 또는 under flow되어 carry out 출력한다.

1. BCD Adder 의 결과 및 Simulation 과정에 대해서 설명하시오.

(verilog source, 출력 예시, 과정 상세히 적을것)

|  |
| --- |
| `timescale 1ns / 1ps  module inv(  input a0,  input a1,  input a2,  input a3,  input b0,  input b1,  input b2,  input b3,  input c0,    output c1,  output c2,  output c3,  output c4,  wire s0,  wire s1,  wire s2,  wire s3,  output C1,  output C2,  output C3,  output S0,  output S1,  output S2,  output S3,  output k0  );  assign s0 = (a0 ^ b0) ^ c0;  assign c1 = (a0 & b0) | (c0 & (a0 ^ b0));    assign s1 = (a1 ^ b1) ^ c1;  assign c2 = (a1 & b1) | (c1 & (a1 ^ b1));    assign s2 = (a2 ^ b2) ^ c2;  assign c3 = (a2 & b2) | (c2 & (a2 ^ b2));    assign s3 = (a3 ^ b3) ^ c3;  assign c4 = (a3 & b3) | (c3 & (a3 ^ b3));    assign k0 = c4 | (s3 & s2) | (s3 & s1);    assign S0 = (0 ^ s0) ^ c0;  assign C1 = (0 & s0) | (c0 & (0 ^ s0));    assign S1 = (k0 ^ s1) ^ C1;  assign C2 = (k0 & s1) | (C1 & (k0 ^ s1));    assign S2 = (k0 ^ s2) ^ C2;  assign C3 = (k0 & s2) | (C2 & (k0 ^ s2));    assign S3 = (0 ^ s3) ^ C3;    endmodule |

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

앞선 4bit-Binary Adder를 이용해서 4bit씩 2개의 input을 받아서 하나로 합친다. 이때 10이상인지의 여부를 판별해주기 위해 10 이상인지 여부를 k0 = c4 | (s3 & s2) | (s3 & s1);로 판단 해준다. 만약 10 이상이라면 두번째 4bit-Binary Adder에서 이전의 계산 결과에 6을 더해주고 그렇지 않다면 0을 더하는 연산을 해준 결과값을 출력한다. 이때 4bit는 일의 자리를 표현, K는 십의 자리를 표현한다.

1. **결과 검토 및 논의 사항.**

4bit adder와 4bit subtractor 둘 다 4bit씩 2개의 수를 입력 받고, 합, 차 연산을 한다.

그리고 over-flow, under-flow에서 동일하게 Carry Out을 출력한다. 이는 Full Adder, Subtractor가 병렬 연결되면, 4bit 이상 nbit까지 확장 가능하다.

또한, BCD Adder의 경우 십의자리와 일의자리로 나눠서 고려하게 되는데 이때 각각 10을 나타내는 방식을 생각해보면 10000과 01010으로 차이가 생긴다. 이 차이를 계산해보면 10 이후로는 항상 6씩 차이가 나게 되는데, 6을 더해줌으로써 변환하게 된다. BCD형태의 10 이상 판정은 k-map을 그려보면 3개의 and와 or로 간단하게 표현할 수 있다.

1. **추가 이론 조사 및 작성.**

만약 2의 보수형태로 표현된, 즉 음수가 포함된 덧셈 뺄셈에서는 어떻게 해야하는지 알아보았다. 만약 2의 보수형태로 표현된 수가 input 되었다면 음수 판정이 필요한데, 만약 음수라면 모든 수를 XOR로 뒤집고, 1을 더해주는 과정이 필요하다. 양수라면 0을 입력, 음수라면 1을 입력하는 과정을 추가하여 4bit-adder에서 full adder에 넣기전 XOR을 통과 시키는 과정이 필요하게 되는 것이다.