10주차 결과보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1. 4bit Binary Parallel Adder 의 결과 및 Simulation 과정

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

* V파일 설명

4 bit Binary Parallel Adder는 4bit의 이진수를 서로 더하는 회로이다. 따라서, 각 하나의 bit를 더할 때 Carry 값과 Sum값이 출력된다.

이 회로의 입력 값은 A3A2A1A0와 B3B2B1B0의 각 자리와, 처음 Full adder를 하기 위한 Cin값이 필요하다. 출력 값은 A의 각 자리 수와 B의 각 자리 수를 합한 Sum값인 S0,S1,S2,S3 값이 있고, 각 자리 수에 맞게끔 Carry값도 출력된다.Ripple Carry Adder 회로에서

Ci = Ai-1&Bi-1 + Ci-1(Ai-1 & Bi-1)

Si = Ai ^ Bi ^ Ci이다.

* tb파일 설명

리스트 자료 구조를 사용하여서 기존의 tb파일들과 사뭇 다르다. initial begin 이후의 line들만 설명하자면, a와 b는 각각 4비트 이진수 0으로 초기화한다는 뜻이고, Ci는 논리 값 0으로 설정된다는 의미이다. 그 이후 always 구문은, a<= #10 a + 4’D0001의 의미는 10 time 마다 1씩 증가한다는 의미이다. b <= #20 a – 4’D0001은 20 타임 마다 ‘a’에서 1을 뺀 값으로 설정된다는 것으로 해석할 수 있다.

텍스트, 디스플레이, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

-Simulation 결과를 몇 개 분석해보자면,

a값이 1000, b값이 0101, cin은 1인 경우에는,

1110이 된다. cin은 1이기 때문에, a0와 b0를 더한 값은 sum은 1이고 Carry는 0이지만, cin까지 포함하여 sum값은 0, carry 값(C1)은 1이된다. a1과 b1을 더한 값은 0이지만, c1의 값 때문에 sum은 1이 되고, carry는 0이된다. a2와 b2의 sum은 1이고 carry는 0이다. a3와 b3의 sum은 1이고, carry – out은 0이다. 따라서, sum값은 1110이 되고, Cout은 0이 되는 것을 알 수 있다.

a 배열의 값이 0100, b 배열의 값이 0001의 값을 가질 때, Cin값을 0이라면,

s0은 1, c1은 0/s1은 0, c2는 0/s2는 1, c3은 0/s3는 0, cout은 0이 된다.

따라서, 결과 s 배열의 값은 0101이 된다.

a의 배열의 값이 1110, b 배열의 값은 1011로 생각해보자, cin은 1이다.

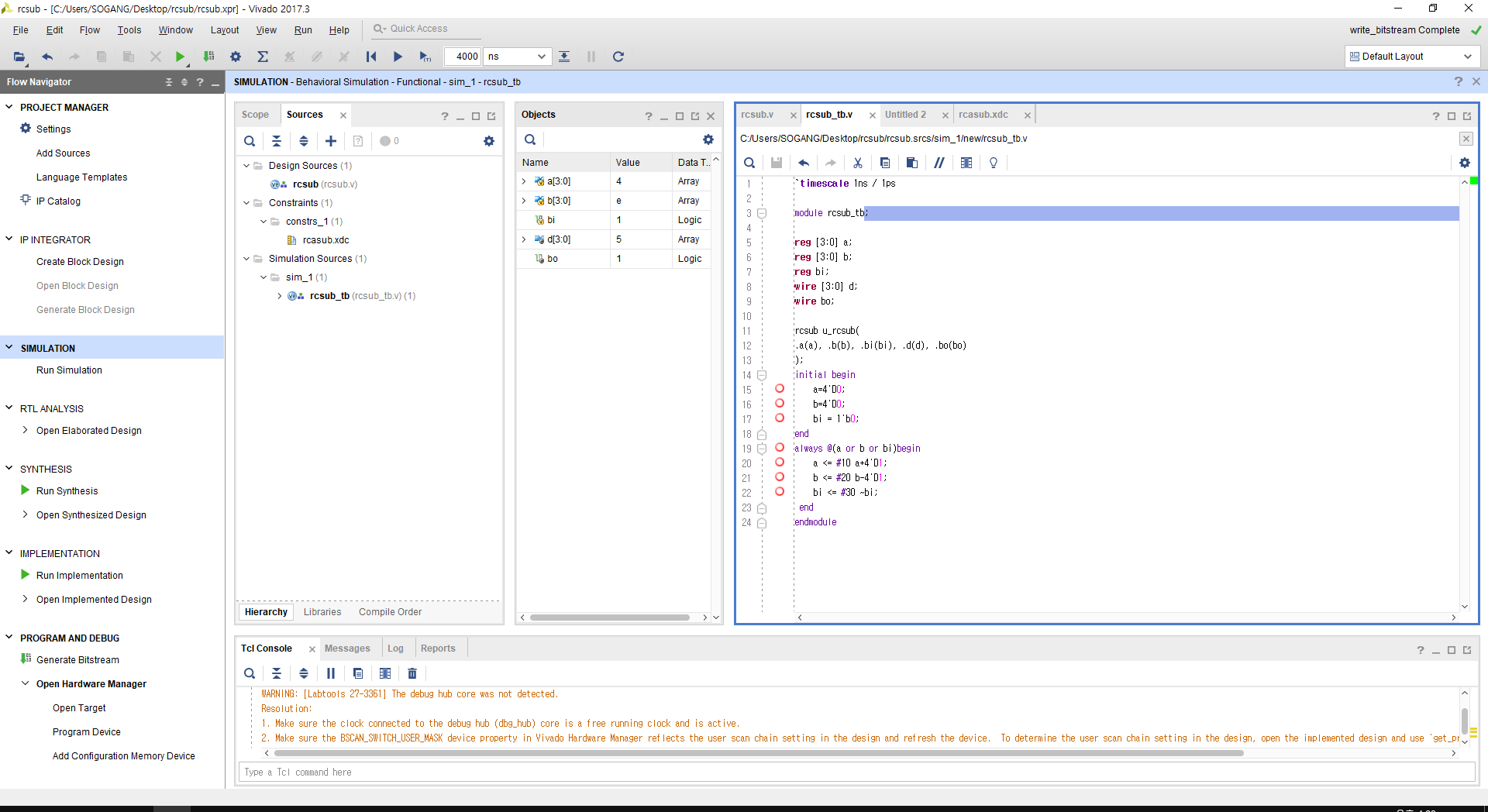
s0은 0, c1은 1/s1은 1, c2는 1/s2는 0, c3은 1/s3는 1, cout은 1이 된다.

s의 배열의 값은 1010이다.

이 두 binary값을 더했을 때를 생각해봤을 때, 1110 + 1011 = 11010이다. Cout값이 1이기 때문에 11010이 맞다.

2. 4bit Binary Parallel Subtractor의 결과 및 Simulation 과정

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

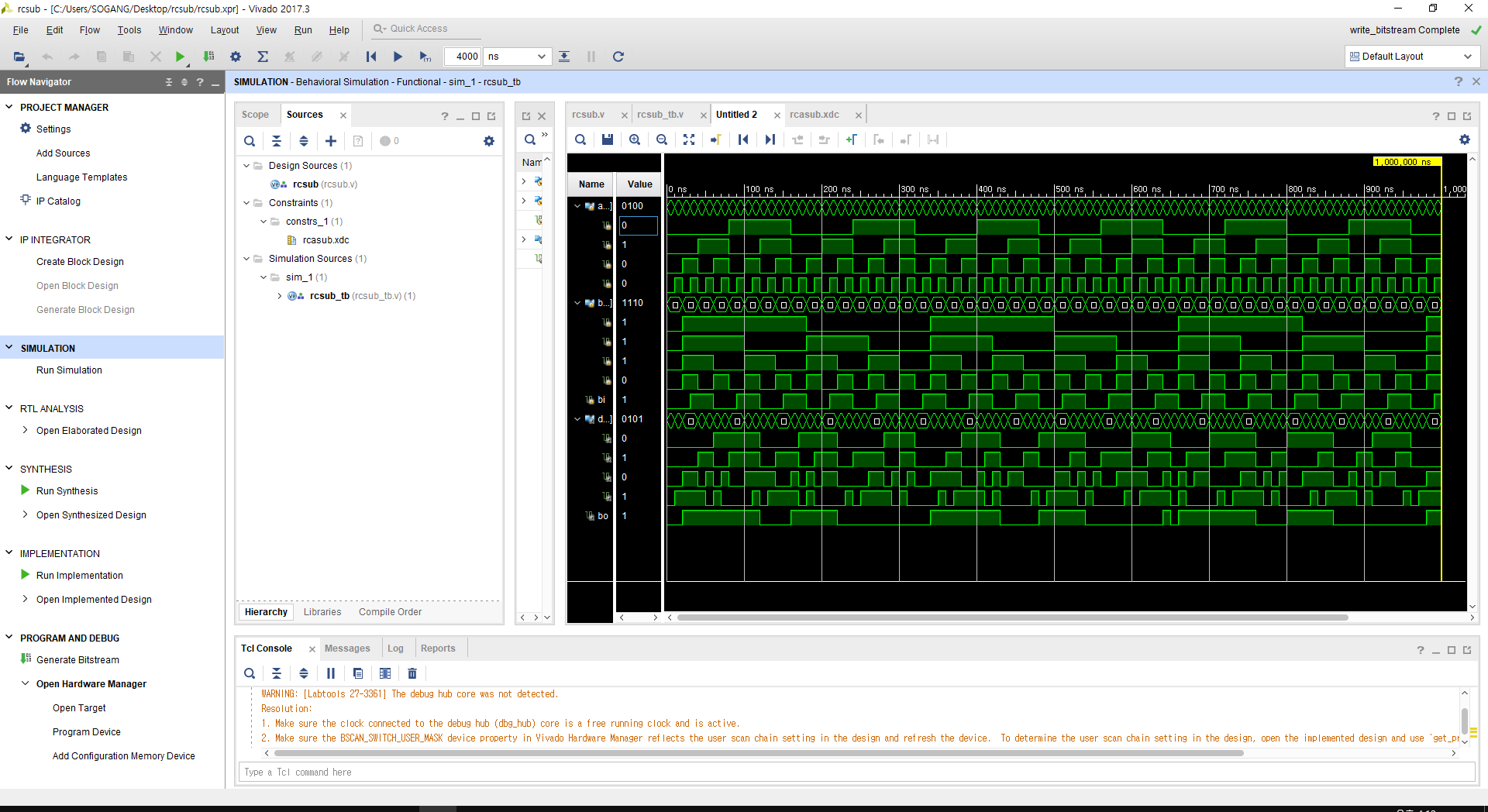
* v파일 설명

4 bit Binary Parallel Subtractor의 경우 Adder는 Full-Adder를 4개 사용했지만, Full Subtractor 4개를 사용한다. 이는 각 4bit의 이진수에 자리 빌림 수(Borrow)와 차(Difference)를 출력하고, 이 논리식은

Di = Ai ^ Bi ^ bi

bi+1 = (Ai ^ Bi)’∙bi + (Ai’•Bi)이다.

tb파일의 경우 위 4 bit Binary Parallel Adder의 경우와 동일하기 때문에 설명을 생략하도록 하겠다.



-simulation example 설명

a 배열의 값이 1000, b 배열의 값이 0100, bin값이 0이라고 했을 때이라고 했을 때, bout의 값은 0이고 d 배열의 값은 0100이라고 할 수 있다.

a의 배열들의 값이 0100과, b 배열의 값이 1110인 경우, bin의 값은 1인 경우를 살펴보자면, d배열의 결과는 0101이다. 0100 – 1110 = 0110이지만, bin값까지 고려한다면, bout값은 1, d 배열의 값은 0101이 된다.

a의 배열이 0100이고 b의 배열이 1110, bin 값은 0인 경우에는 d 배열의 값은 0110이다. 그 이유는 두 이진수를 빼면 음수가 되는데, borrow out값을 고려하면, 0110이 되기 때문이다.

Decimal하게 정리를 한다면, 결과에 borrow out 값(-16)을 더 해줬기 때문에 올바르다.

3. BCD Adder의 결과 및 Simulation 과정

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

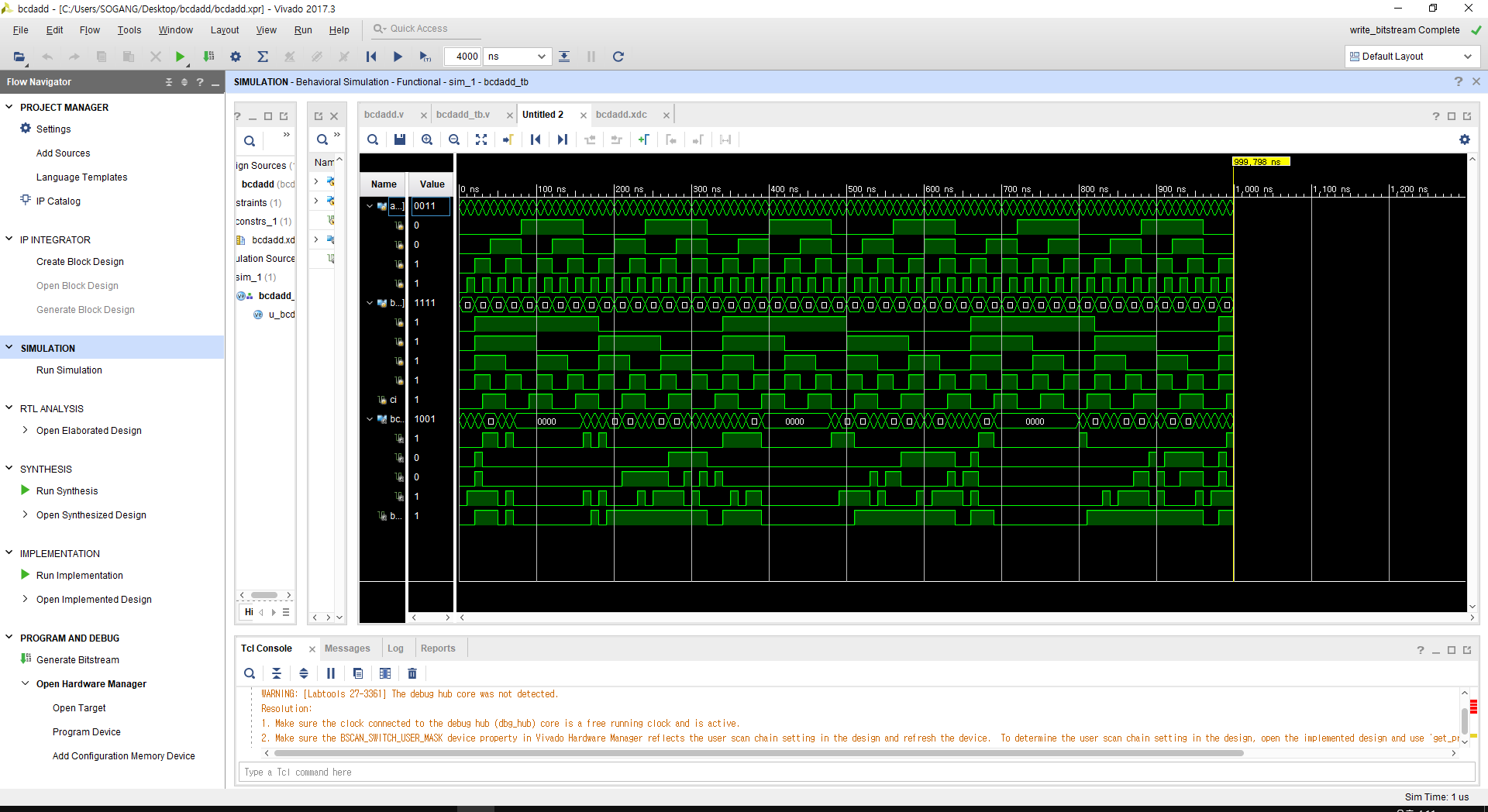
자동 생성된 설명

- V파일 설명

십진수의 덧셈을 하기 위한 연산기이고, BCD는 한 자리가 4bit로 이루어집니다. 10보다 큰 경우에는 윗자리에 올림 수(Carry)가 발생하기 때문에 추가적인 연산이 필요하다.

위 Verilog coding에서 c값과 s값은 Full adder를 사용한 RCA회로와 동일합니다. 하지만, 이를 십진수의 덧셈을 하기 위해서 바꿔본다면, 10 이상의 수에 6을 더한 후 나타내야 한다.

* simulation example 설명



a의 값이 0011인 경우, b 값이 1111이고, cin이 1인 경우에는

Decimal하게 add를 한다면 a = 3이고 b= 15이고 cin = 1이므로, 결과는 19이다.

하지만 10 이상의 수의 경우에는 다른 자리로 구현했기 때문에, 1 1001이다. 맨 앞의 자리는 10을 의미하고 나머지 4자리의 숫자는 9를 의미한다.

또한 a 값이 0010, b의 값이 0011이고, cin은 0인 경우에는

Decimal하게 add를 한다면 a= 2이고, b =3이고, cin은 0이기 때문에 결과는 5이어야 한다. 이는 10을 넘어가지 않기 때문에 0 0101이다.

4. 결과 검토 및 논의 사항.

Ripple Carry Adder인 4 bit의 binary Parallel Adder와 Subtractor를 구현하며, 다시 한번 Full -Adder, Full – Subtractor의 개념을 복습할 수 있었고, 이전까지 Verilog을 코딩하면서 배열을 통한 input과 output은 해본 적이 없었지만, 이번 기회에 배열을 통해 입 출력을 하면서 array 문법을 조금 살펴볼 수 있었다. 또한, 이러한 배열에 testbanch를 어떻게 주고, 어떻게 초기화할지를 살펴볼 수 있었다. 또한, 회로 동작 중 나오는 값인, cout을 제외한 carry 값 같은 경우에는 FPGA 회로에 나타낼 필요성이 없어 wire 변수와 xdc 파일을 적절히 활용해서 시뮬레이션 및 회로에 결과를 나타내지 않게 할 수 있음을 알게 되었다.

5. 추가 이론 조사 및 작성.

CLA(Carry Lookahead adder)는 carry값을 한번에 연산하여, 낮을 자리 수부터 순차적으로 연산하는 Ripple Carry Adder인, 실습 과정과 다르게 빠른 연산을 구현할 수 있다.

Ci+1 = Ai ∙ Bi + (Ai ⊕ Bi ) ∙ Ci = Gi + Pi ∙ Ci

Ci+2 = Gi+1+Pi+1Ci+1 = Gi+1+Pi+1Gi+Pi+1PiCi

Ci+3 = Gi+2+Pi+2Ci+2 = Gi+2+Pi+2Gi+1+Pi+2Pi+1Gi+Pi+2Pi+1PiCi

Ci+4=Gi+3+Pi+3Ci+3= Gi+3+Pi+3Gi+2+ Pi+3Pi+2Gi+1+Pi+3Pi+2Pi+1Gi + Pi+3Pi+2Pi+1PiCi

즉, Carry 값들을 미리 P값과 G값을 알고 있으면 무리 없이 빠르게 연산을 진행할 수 있다. 위에서 P값과 G값은 각각 Ai ^ Bi와 AiBi이다. 이러한 CLA의 CLA generator를 하나 더 연결한 2 – level CLA도 존재한다. 이 2 level은 CLA에서는 4-bit로 그룹을 묶어서 각 그룹의 최종 Carry만을 연산하여 다음 레벨로 전달한다. 다음 2-level에서는 전달받은 Carry를 통해 나머지 Carry들을 연산한다. 이 4그룹이 빠르게, 동시에 연산이 진행되고 있기 때문에 더더욱 빠른 연산을 구현할 수 있다.