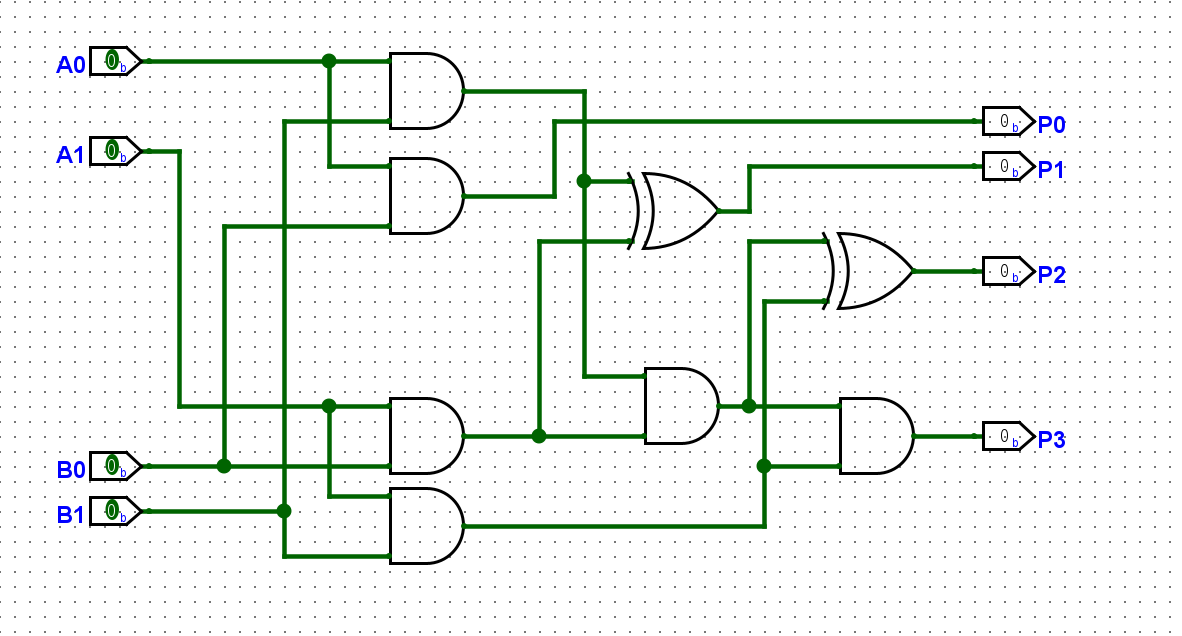
**(설계 과정이 기술하여야 합니다. 과정이 없으면 상당한 감점이 있습니다.)**

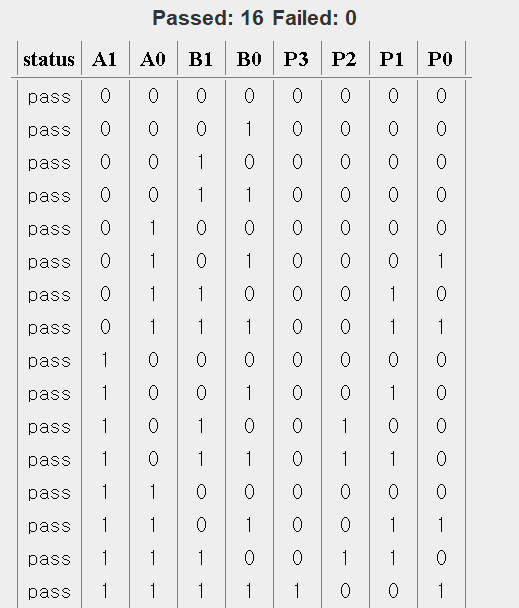
1. Non-negative binary multiplier를 설계하고 이를 logisim-evolution을 이용하여 검증하고자 한다,
2. **Two-bit multiplier**를 설계하여라. 즉  **을 수행하는 multiplier를 설계**하여라. 설계 과정을 설명하고, 이를 logisim-evolution을 이용하여 구현하고, logisim-evolution의 test vector를 이용하여 정상적으로 동작함을 보여야 한다. (15점)

**[5월 중간고사 설명]**

**A1A2를 잘못 표현하신 것 같아서 A1A0으로 고쳐서 작성을 시작했습니다. 각각 A1과 B1이 MSB이며 A0와 B0가 LSB입니다. 출력값도 마찬가지고 P3가 MSB P0가 LSB로 표현한 것입니다.**

A1A0와 B1B0를 각각 곱하면 처음에 4개의 곱이 각각 이루어지므로 4개의 AND 게이트가 필요하다. 그리고 두 번의 HALF ADDER가 필요해서 XOR 게이트와 AND 게이트를 이용해 회로를 구성했고 Test vector 까지 성공적으로 돌아갔다. Xor -> sum AND-> carry를 의미함





1. (a)에서 구현한 **two bit multiplier를 최대한 이용하여** 4 비트와 3 비트를 곱하는 **multiplier**를 설계하여라. 즉  **을 수행하는 multiplier를 설계**하여라. 설계 과정을 설명하고, 이를 logisim-evolution을 이용하여 구현하고, logisim-evolution의 test vector를 이용하여 정상적으로 동작함을 보여야 한다. (test vector는 주어진 test vector 파일을 이용하여 테스트한다. 파일 내에 주어진 변수 이름은 자신의 설계 맞추어 변경하여 사용하여도 됨.) (설계 시에 (a)에서 구현한 **two bit multiplier를 최대한 이용하지 않으면 0점 처리됨.)** (25점)

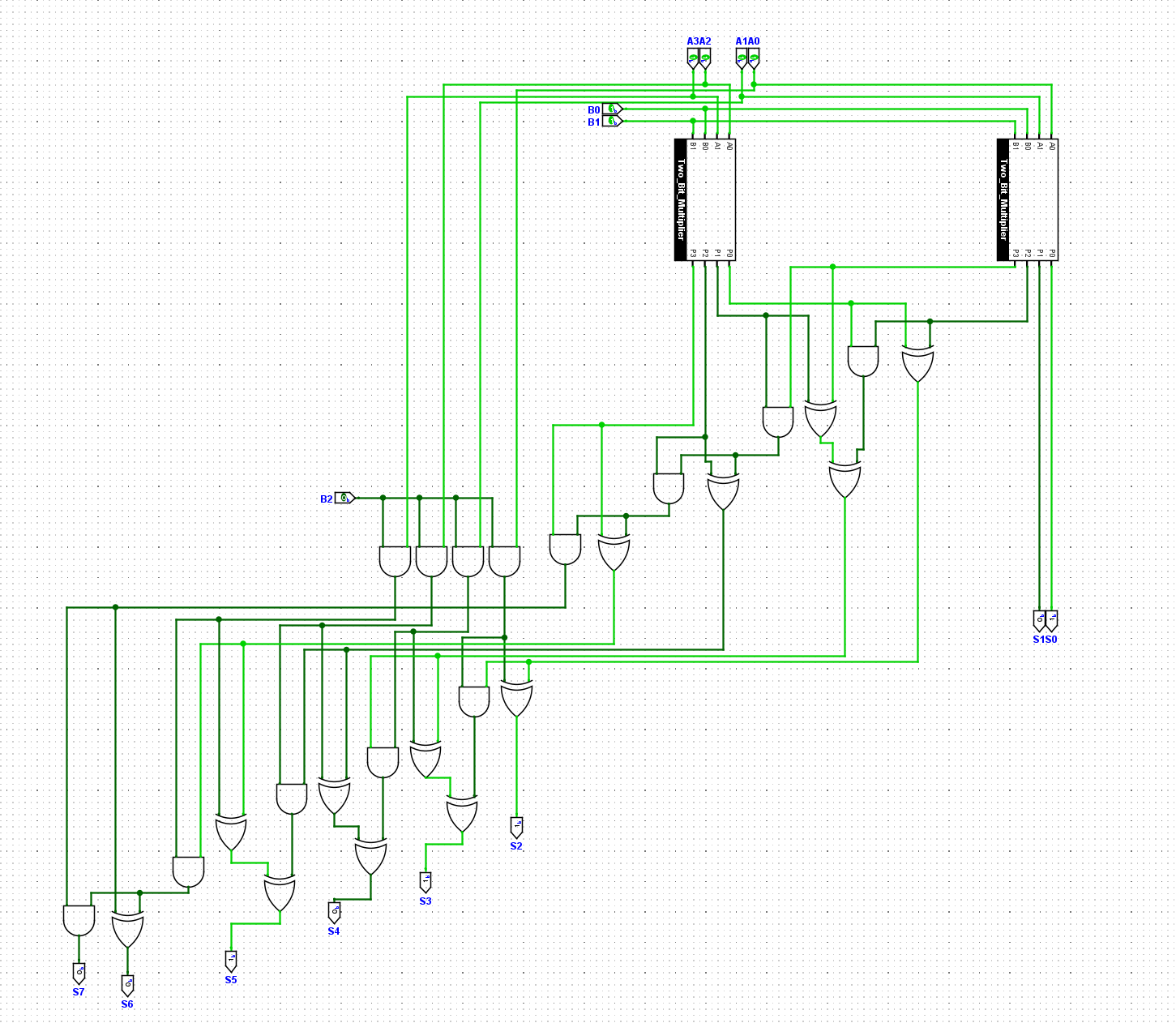
**[5월 30일 중간고사 설명]**

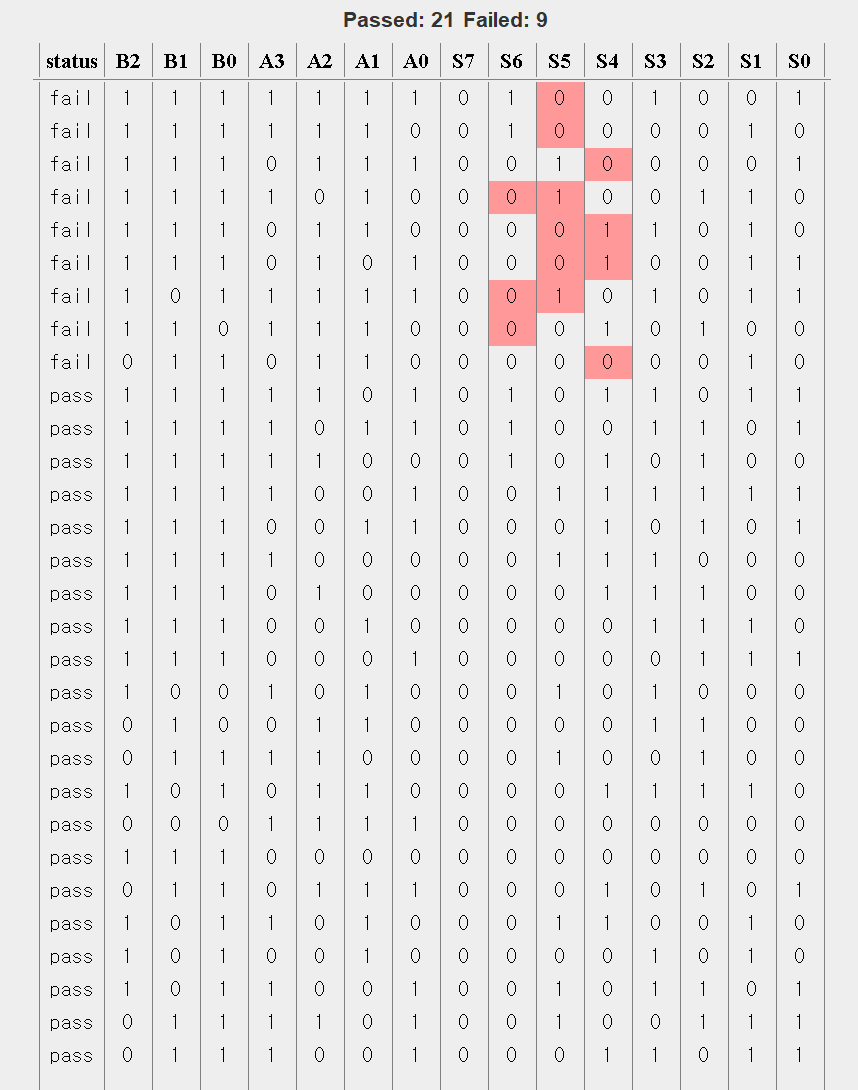
**(a)번에서 구현한 two bit multiplier를 이용하기 위해서는 각각의 비트 연산에서 발생하는 carry와 sum을 통해 출력값 [S7 S6 S5 S4 S3 S2 S1 S0] (S7: MSB S0: LSB) 을 표현해야 한다고 생각했습니다. 우선 2개의 two bit multiplier는 B2B1B0 중 B1B0와 A3A2A1A0가 연산되는 것을 표현하여 S0와 S1이 출력되는 것을 보실 수 있습니다. 그리고 이후의 연산과정은 각각의 비트연산이 AND 게이트와 Xor 게이트를 통해 Sum과 carry가 발생하는 것을 표현했습니다.**

2 bit multiplier 와 Adder를 기본 회로인 Xor 게이트와 And 게이트로 구현했습니다.

B0B1을 먼저 2bit multiplier를 이용해 sum을 구하고 이후에

B3와 A0A1A2A3를 각각 곱한 것과 회로를 이용해 합을 표현하려고 하였습니다.





\*\*\* **제출물**: **답안지**와 **Logisim-evolution 파일** (circ 파일)을 online으로 hisnet에 제출. ((a)에서의 설계 회로는 (b)에서 sub-circuit에 포함되도록 하여서 circ 파일로 제출. Test vector는 제출하지 않아도 됨. 다만 test 결과는 capture하여 답안지에 포함시켜야 함.)

\*\*\* 설계 시에 **이용할 수 있는 소자는 기본 logic gate들**이다. (NOT, AND, NAND, OR, NOR, XOR, XNOR) 입출력을 위한 component들은 당연히 사용할 수 있다. (설계 시에 자신이 필요한 sub-circuit를 적절하게 구현하여 사용하는 것이 설계를 간단하게 하는 방법일 것이다.)