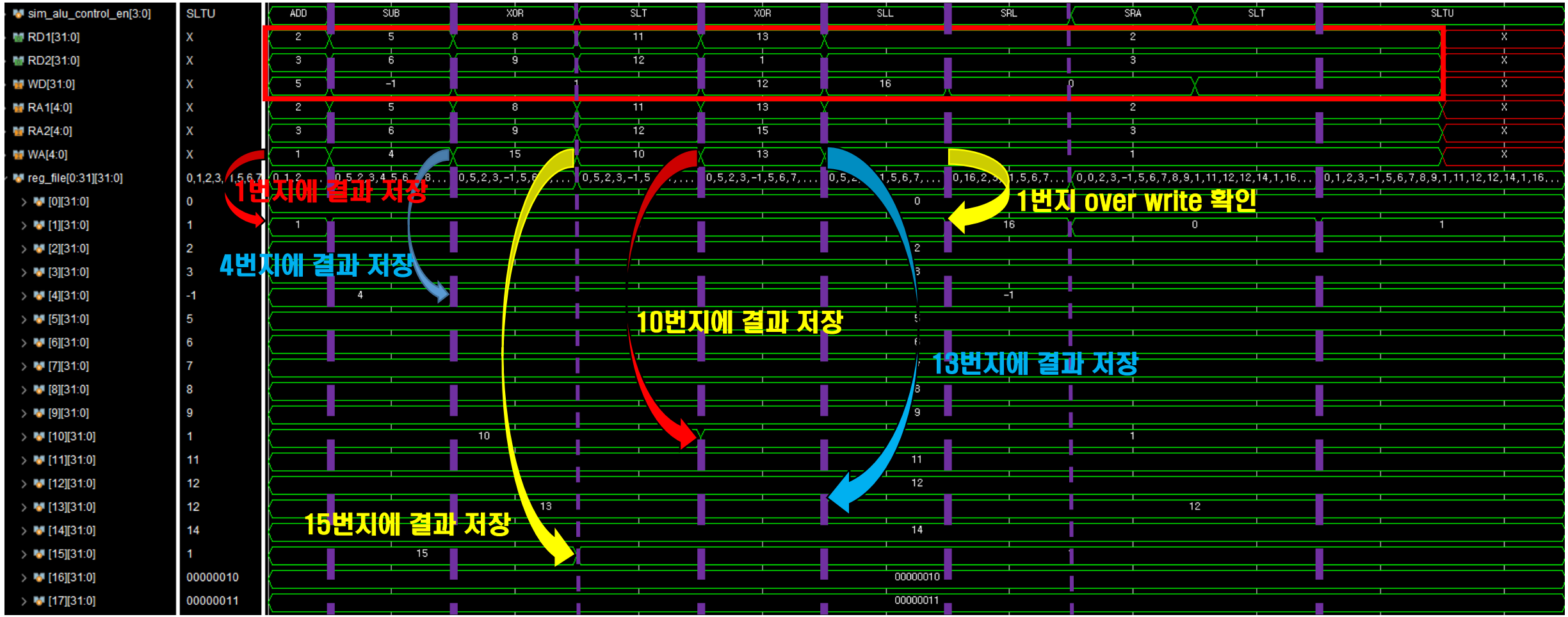
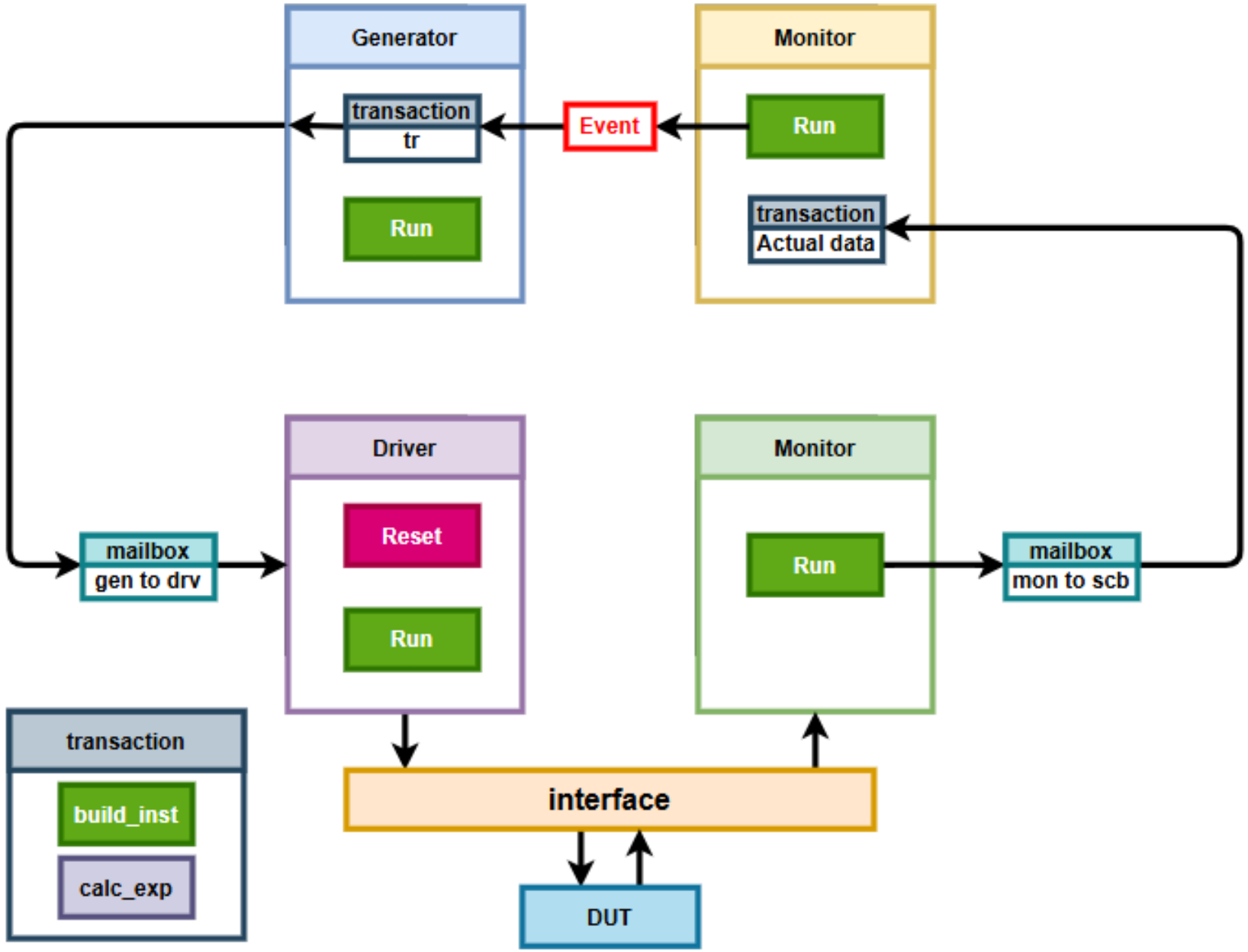
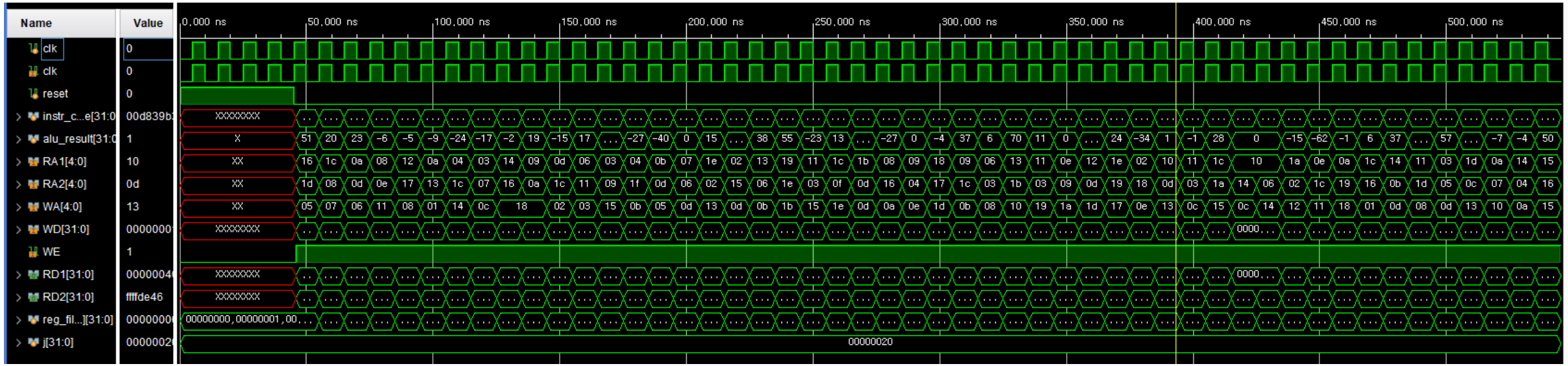
R

3 + 2 = 5 | 5 − 6 = −1 | 1000 ^ 1001 = 0001 | 11 < 12 ? 1 | 1101 ^ 0001 = 1100 | 0010 << 3 = 1_0000 | 0010 >> 3 = 0000 | 2 >>> 3 = 0 | 2 < 3 ? = 0



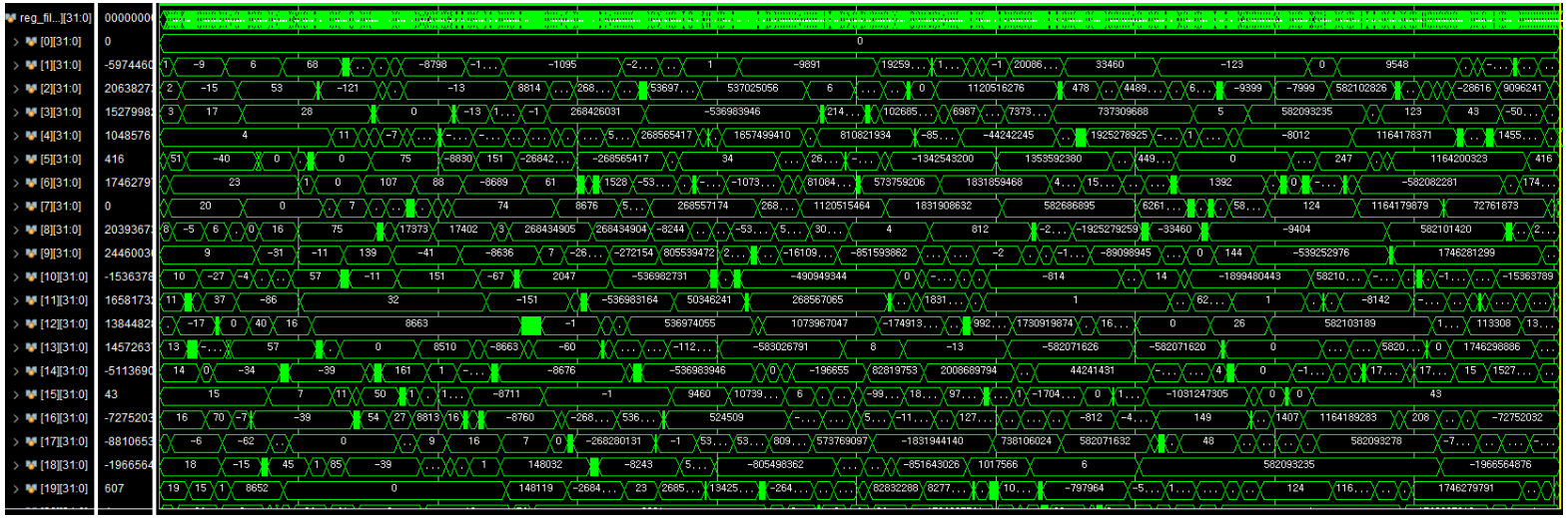

Generator
transaction
tr
Run
Event
Monitor
Run
transaction
Actual data
Driver
Reset
Run
mailbox
gen to drv
Monitor
Run
mailbox
mon to scb
transaction
build_inst
calc_exp
interface
DUT

R 검증



Register file 변화 확인

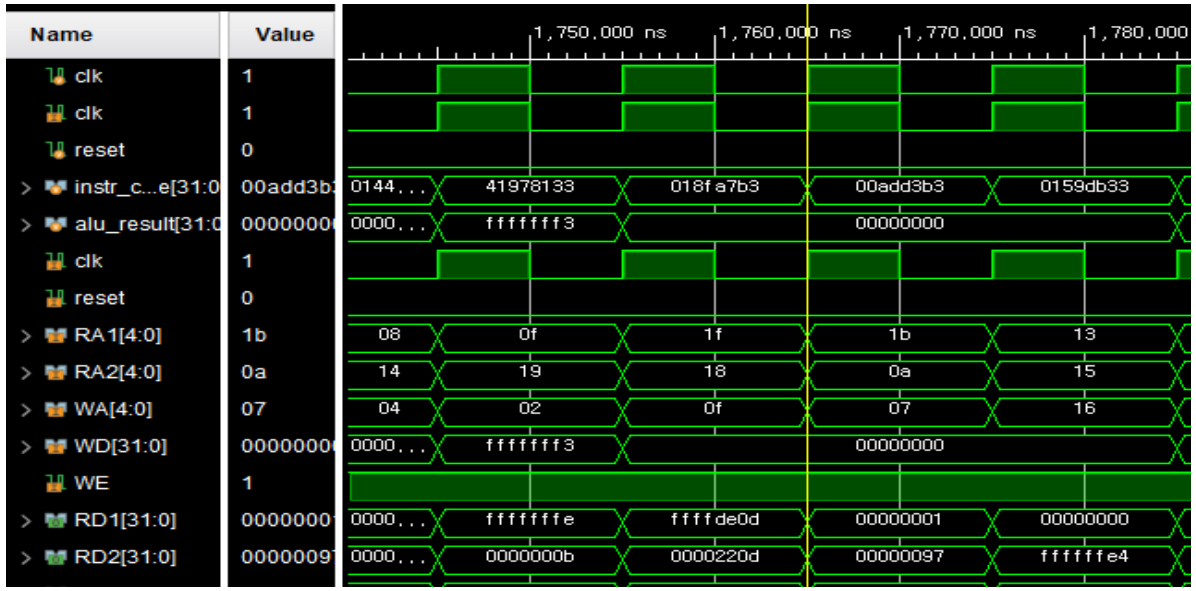
PASS=1000 FAIL=0



트러블슈팅

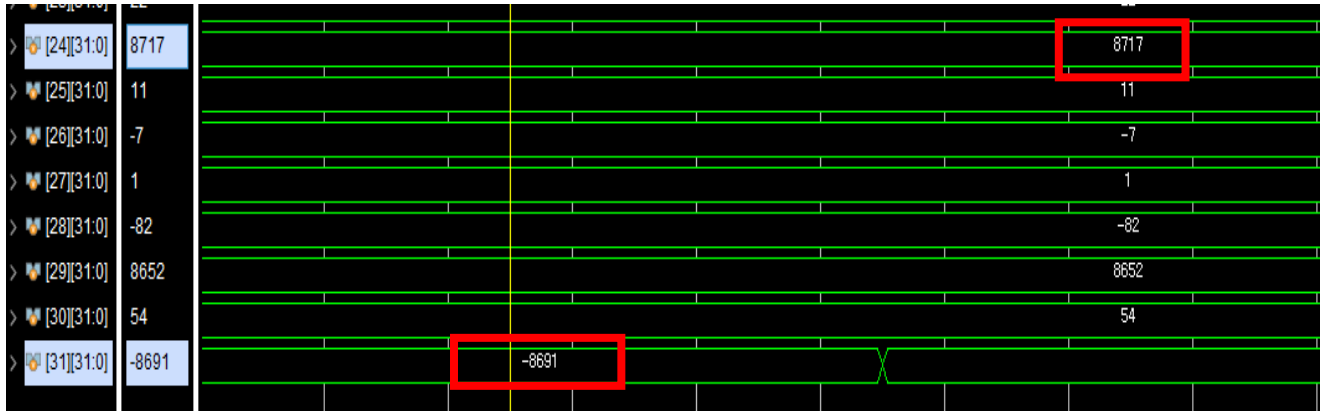
```
[GEN] instr=018fa7b3 rs1=31 rs2=24 rd=15 funct3=2 funct7b5=0
[1765000 ns] [SCB] FAIL  ALU=0  EXP=1
```

SLT연산 시 Error 발생
-8691 < 8717
ALU = 0 ?

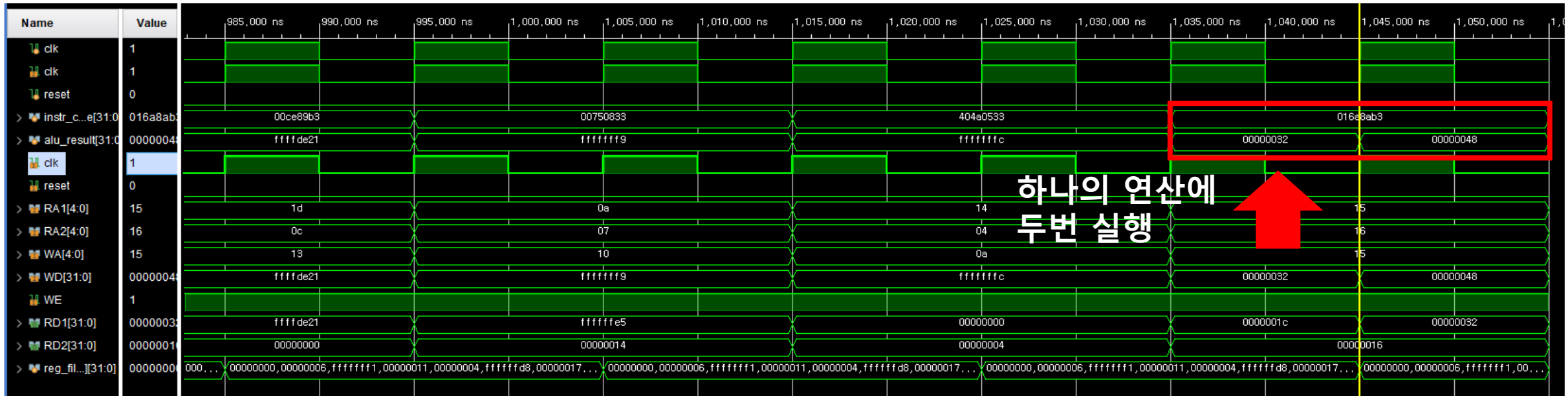



```
4'b0010: begin  //
    alu_result = ($signed(RS1) < $signed(RS2)) ? 1 : 0;
end
```

Unsigned비교 이기에
8717보다 큰 값으로 인식하여 0으로 생성
Signed연산으로 수정




```
[GEN] instr=018fa7b3 rs1=31 rs2=24 rd=15 funct3=2 funct7b5=0
[1765000 ns] [SCB] PASS  ALU=1  EXP=1
```




```
task run();
    @(posedge cif.clk);
    forever begin
        transaction tr;
        gen2drv.get(tr);
        cif.instr_code = tr.instr_code;
        drv2mon.put(tr);
    end
endtask
```

명령어 연산은 clk의 사용하지 않기에 한클럭의 기다리는 과정에서 예상치 못한 딜레이 발생

Clk이 아닌 #1 지연을 통해 해결

```
task run();
    #1;
    forever begin
        transaction tr;
        gen2drv.get(tr);
        cif.instr_code = tr.instr_code;
        drv2mon.put(tr);
    end
endtask
```