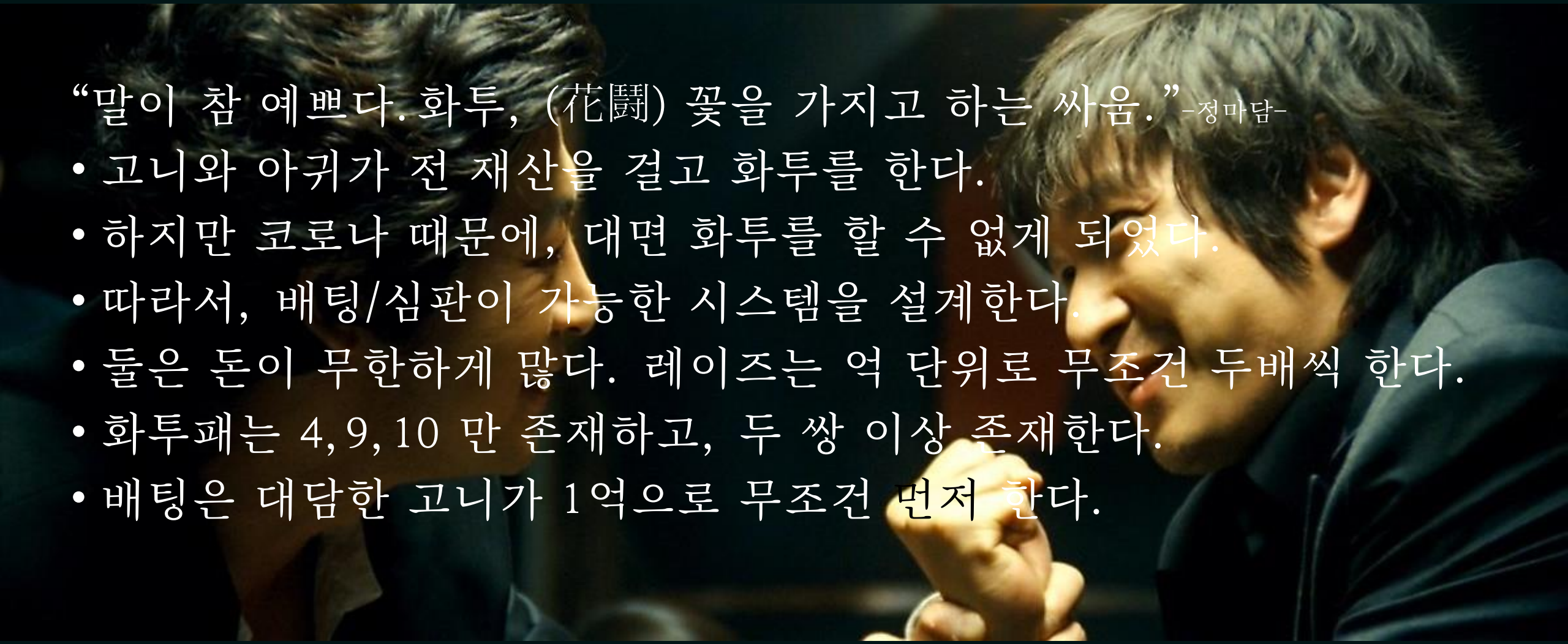
기말 대체 프로젝트

감독 : 디지털시스템 (Digital Logic Design) 박기호 교수님

시나리오 : 18011573 컴퓨터공학과 이장후

dlwkdgn1@naver.com

선정 시스템 및 제한 사항



“말이 참 예쁘다. 화투, (花鬪) 꽃을 가지고 하는 싸움.” -정마담-

- 고니와 아귀가 전 재산을 걸고 화투를 한다.
- 하지만 코로나 때문에, 대면 화투를 할 수 없게 되었다.
- 따라서, 배팅/심판이 가능한 시스템을 설계한다.
- 둘은 돈이 무한하게 많다. 레이즈는 억 단위로 무조건 두배씩 한다.
- 화투패는 4, 9, 10 만 존재하고, 두 쌍 이상 존재한다.
- 배팅은 대담한 고니가 1억으로 무조건 먼저 한다.

선정 이유
교수님의 말씀대로 참신한 것을 해 볼 필요가 있었고, 아무리 생각해도 지난번 제안했던 도어락은 너무 흔했음. 뭘 할까 고민하던 도중 책상 아래에서 화투패 발견.

과제에 임하는 태도
복잡하지 못하면 참신하기라도 하자.
하지만 너무 복잡해져서 힘들었다.

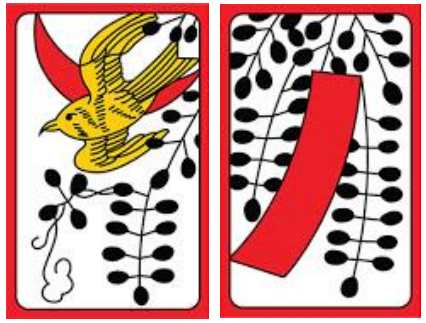
오류 경고
“배팅하다” 를 의미하는 부분이 Quartus 에서 “bet” 대신 “bat” (방망이질) 로 명명됨.

간단한 규칙 설명 (패)

정의한 숫자들

10 (“장” 이라고도 부름.)
Rising edge **3개(11)**

9
Rising edge **2개(10)**



4
Rising edge **1개(01)**

모든 끝은 땡보다 약하다.

동일한 숫자가 나오면 “땡” 이다.
더 높은 숫자의 땡일수록 강하다.

아홉 땡.

숫자가 다르면 숫자합을 더해 10으로 나눈 “끝” 이다.
더 높은 숫자의 끝일수록 강하다.

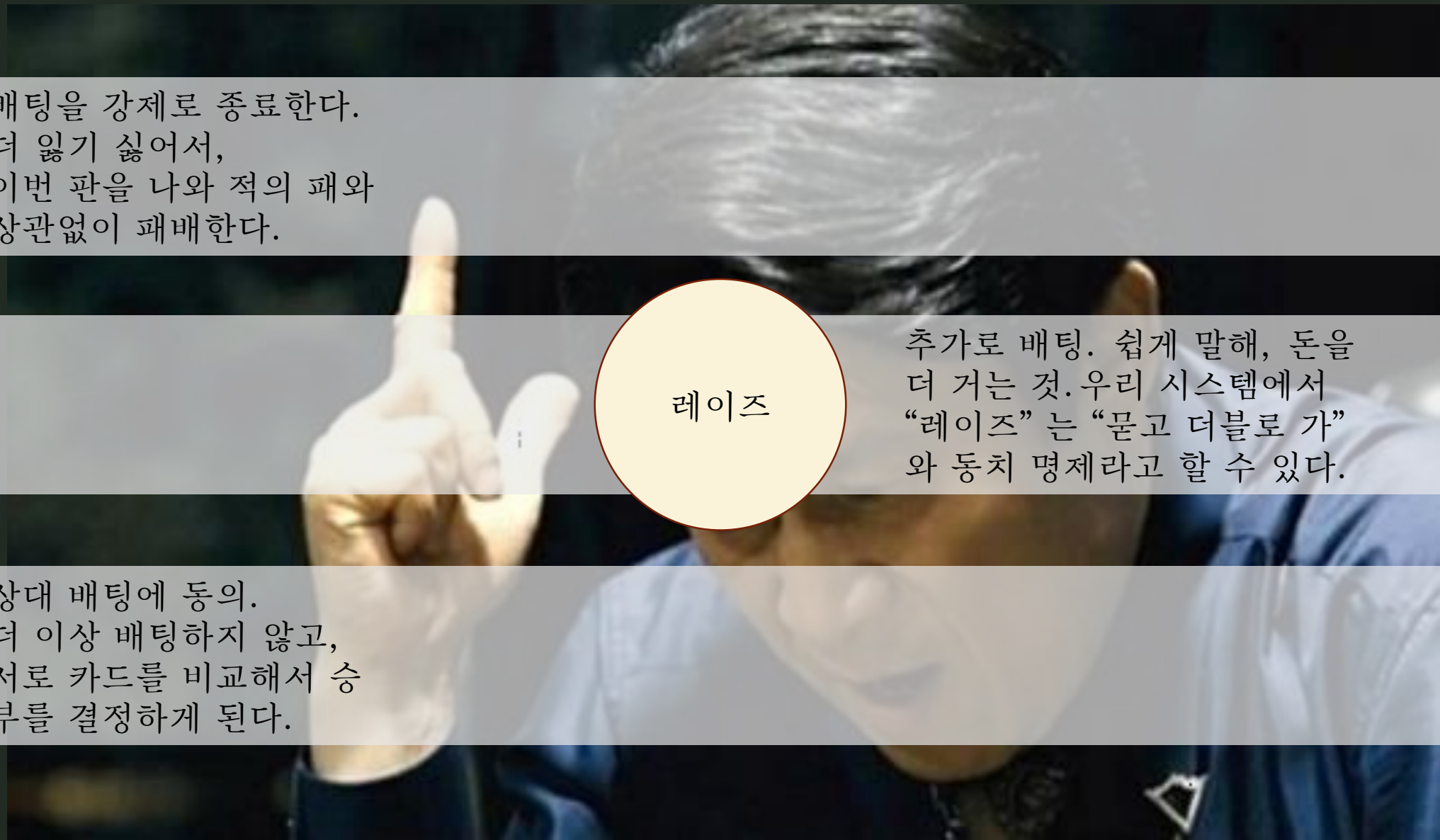
아홉 끝.

예외

끝 중에서도, 4와 9가 동시에 나오면 “무승부” 이다.

사구파토

간단한 규칙 설명 (배팅)



다이

배팅을 강제로 종료한다.
더 잃기 싫어서,
이번 판을 나와 적의 패와
상관없이 패배한다.

레이즈

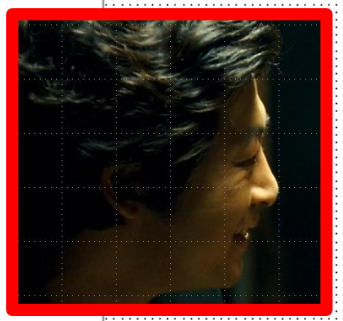
추가로 배팅. 쉽게 말해, 돈을
더 거는 것.우리 시스템에서
“레이즈” 는 “묻고 더블로 가”
와 동치 명제라고 할 수 있다.

콜

상대 배팅에 동의.
더 이상 배팅하지 않고,
서로 카드를 비교해서 승
부를 결정하게 된다.

전체 시스템 outline

고니 책상
goni



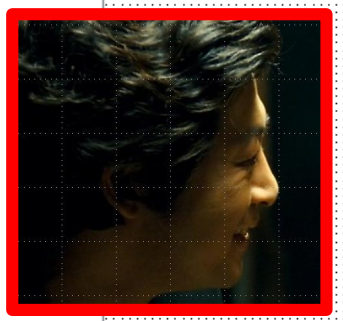
아귀 책상
argwe

배팅기

승패
결과출력

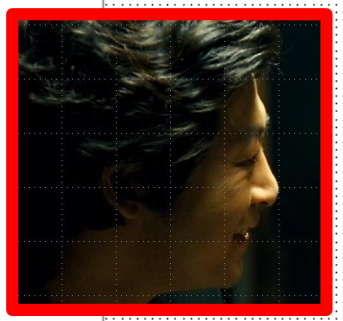
Assignment.bdf | battingsaver.bdf | Compilation Report - Assignment | cardsaver2.bdf | selectfinalwinner.bdf | playercardcomparer.bdf

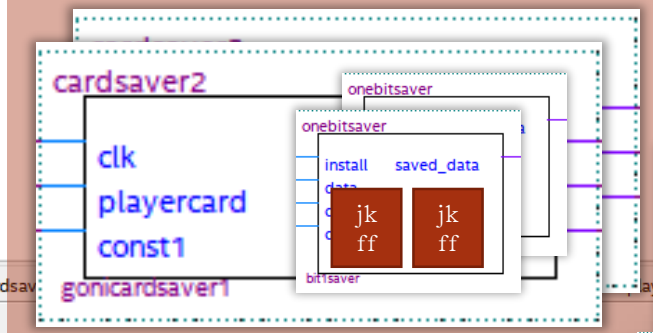
전체 시스템 구조

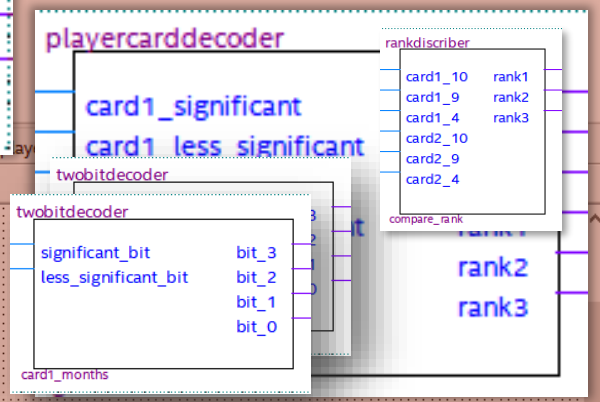


Assignment.bdf
battingsaver.bdf
Compilation Report - Assignment
cardsaver2.bdf
selectfinalwinner.bdf
playercardcomparer.bdf
고니 카드 해독기
goni
아귀 카드 해독기
argwe
카드
비교기
최종
비교기
승패
결과출력
배팅기

전체 시스템 구조




cardsaver2
clk
playercard
const1
gonicardsaver1
onebitsaver
onebitsaver
install
saved_data
jk ff
jk ff
bitsaver


playercarddecoder
card1_significant
card1_less_significant
twobitdecoder
twobitdecoder
significant_bit
less_significant_bit
bit_3
bit_2
bit_1
bit_0
card1_months
rankdiscriber
card1_10
card1_9
card1_4
card2_10
card2_9
card2_4
rank1
rank2
rank3
compare_rank
rank2
rank3


jkff
jkff

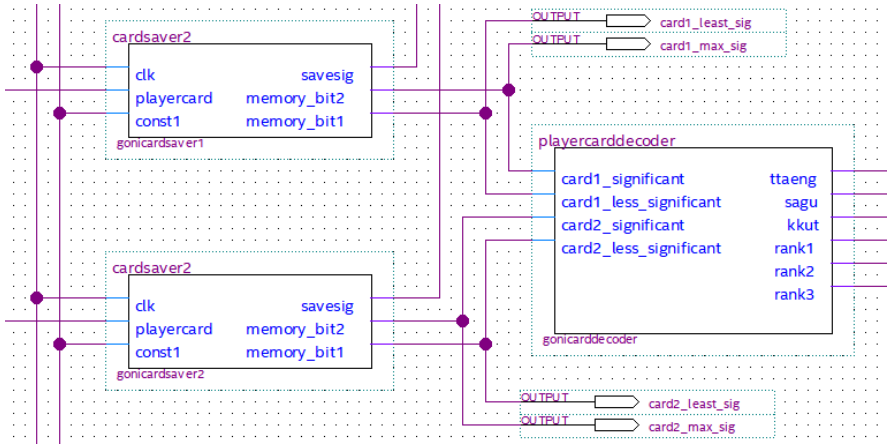
Assignment.bdf
battingsaver.bdf
Compilation Report - Assignment
고니 카드 해독기
goni
아귀 카드 해독기
argwe
카드 비교기
최종 비교기
배팅기

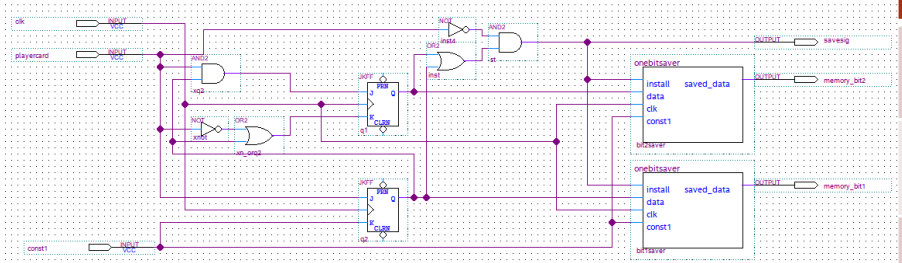
모듈 : 플레이어 카드 해독기 : 설명

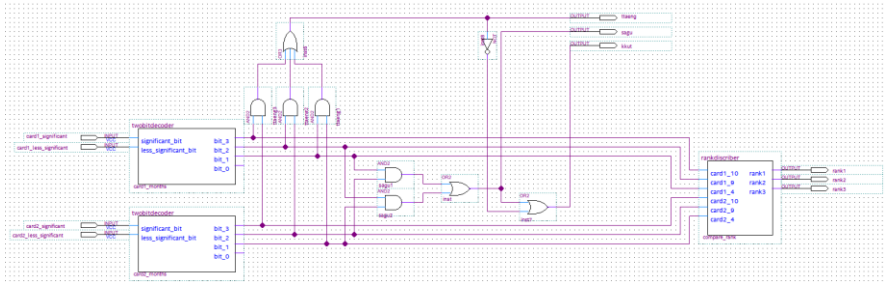
모듈 입력

플레이어가 몇 비트를
연속해서 rising 하는지 입력한다.

입력	설명
Playercard	플레이어 입력
clk	클럭
const1	활용할 수 있는 항상 1 인 신호







모듈 출력

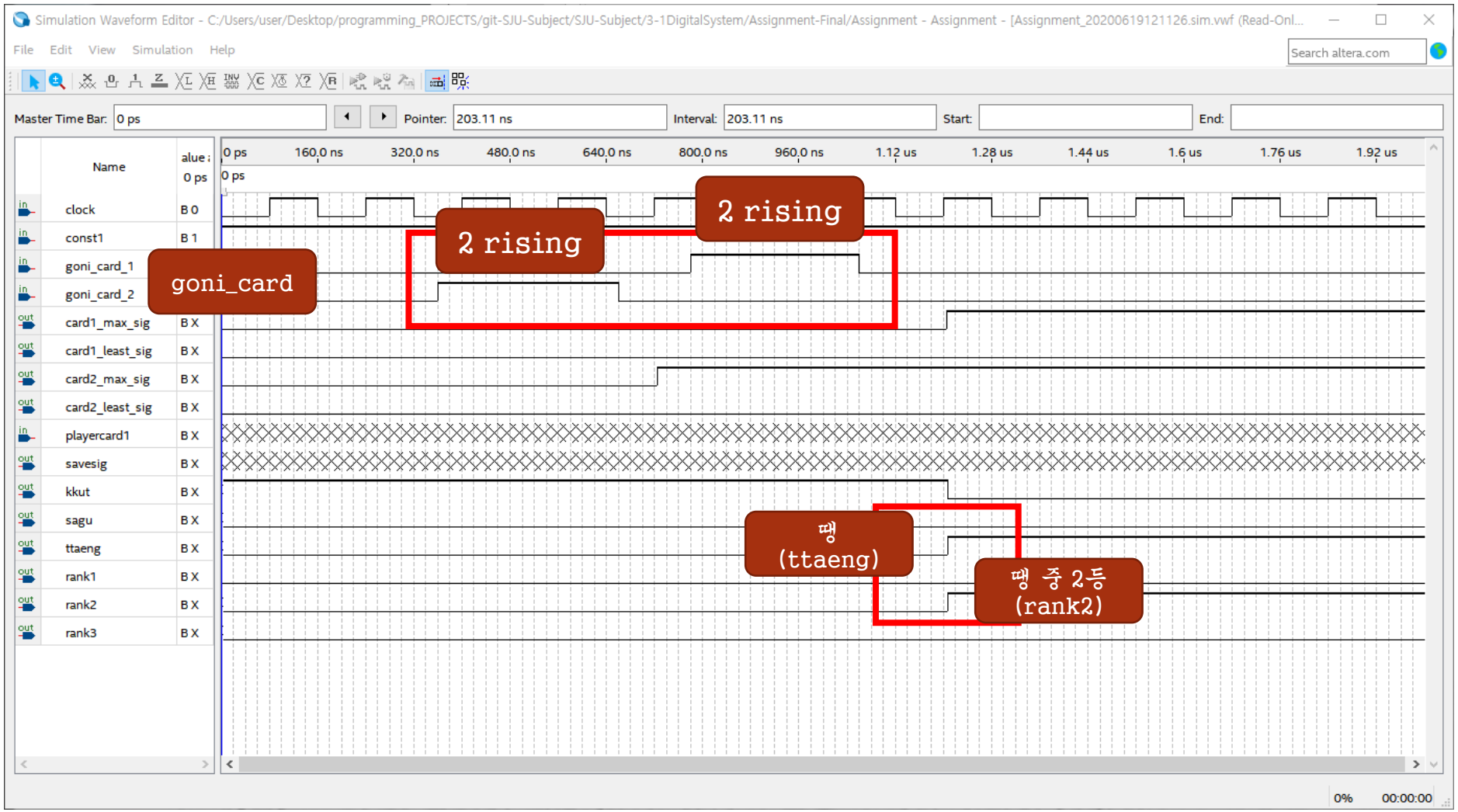
해당 카드가 어떤 카드인지 판단,
두 장을 모두 뽑았는지 확인한다.

출력	설명
Ttaeng	두 장의 카드가 땡인 경우 set
Sagu	두 장의 카드가 49파토인경우
Kkut	두 장의 카드조합이 끝인 경우
Rank1	카드 조합 각 레벨에서 1티어인경우 set
Rank2	카드 조합 각 레벨에서 2티어인경우 set
Rank3	카드 조합 각 레벨에서 3티어인경우 set

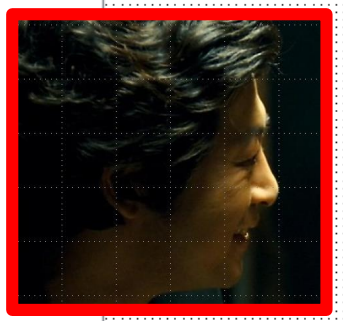
모듈 : 플레이어 카드 해독기 : 예시 (1)

이 카드 해독기는, 아무리 긴 시간 간격을 두고 뜸을 들이고 뽑아도 동작하도록 설계함.

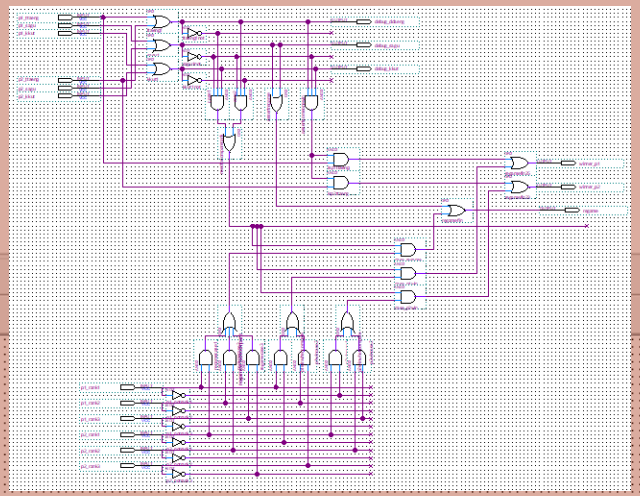
- 아홉땡.



전체 시스템 구조




playercardcomparer
p1_ttaeng
p1_sagu
p1_kkut
p2_ttaeng
p2_sagu
p2_kkut
p1_rank1
p1_rank2
p1_rank3
p2_rank1
p2_rank2
p2_rank3
debug_ddaeng
debug_sagu
debug_kkut
winner_p1
winner_p2
regame
cardcompare



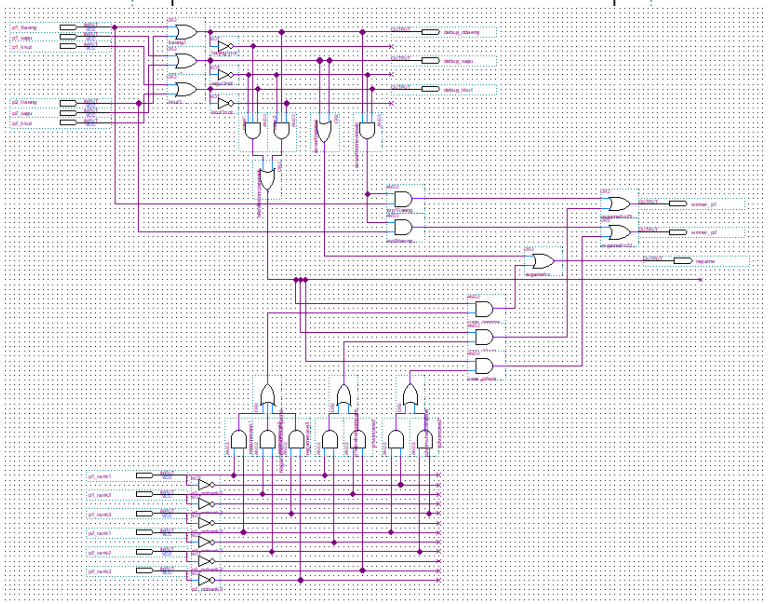
Assignment.bdf
battingsaver.bdf
Compilation Report - Assignment
고니 카드 해독기
goni
아귀 카드 해독기
argwe
카드
비교기
최종
비교기
배팅기

모듈 : 두 플레이어 카드 비교기 : 설명

모듈 입력

두 플레이어의 해독기의 output 을 입력받는다.

입력	설명
P1 (고니, goni) 와 관련된 값들	끝인가? 땡인가? 사구인가? 몇 티어인가?
P2 (아귀, argwe) 와 관련된 값들	



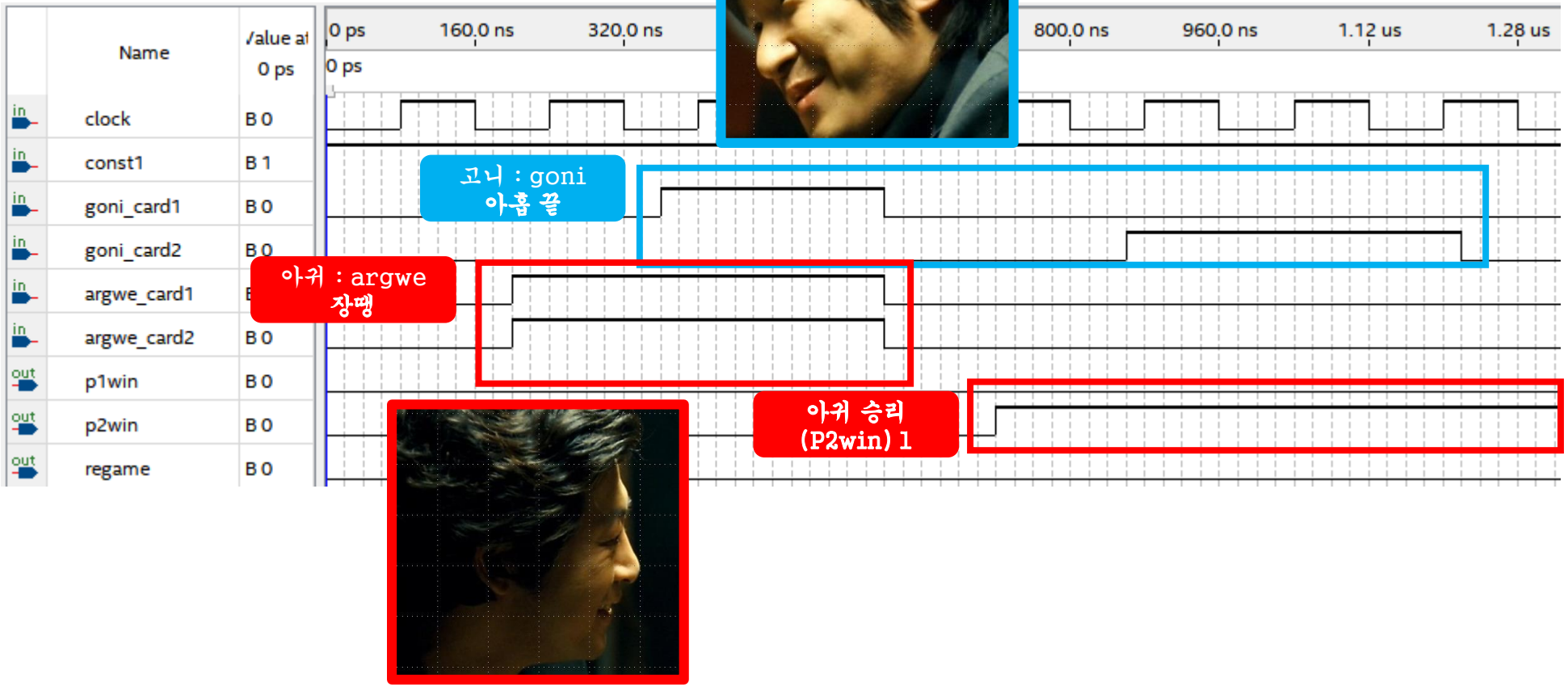
모듈 출력

두 플레이어의 패만을 고려할 때 어떤 결과가 나올지를 출력한다.

출력	설명
winner_p1	두 카드를 비교한 결과 P1 이 이기는 경우
Winner_p2	두 카드를 비교한 결과 P2 가 이기는 경우
regame	게임을 다시 해야 하는 경우

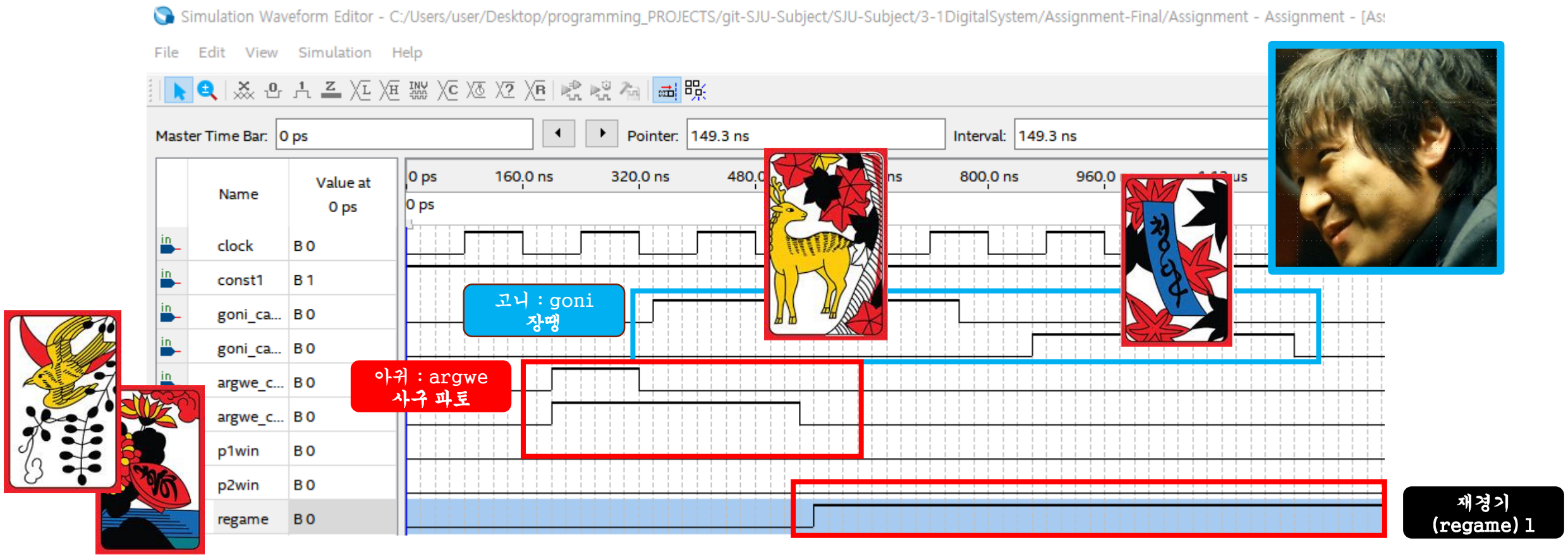
모듈 : 두 플레이어 카드 비교기 : 예시

- 예제 : 패가 잘 안 따라주는 고니



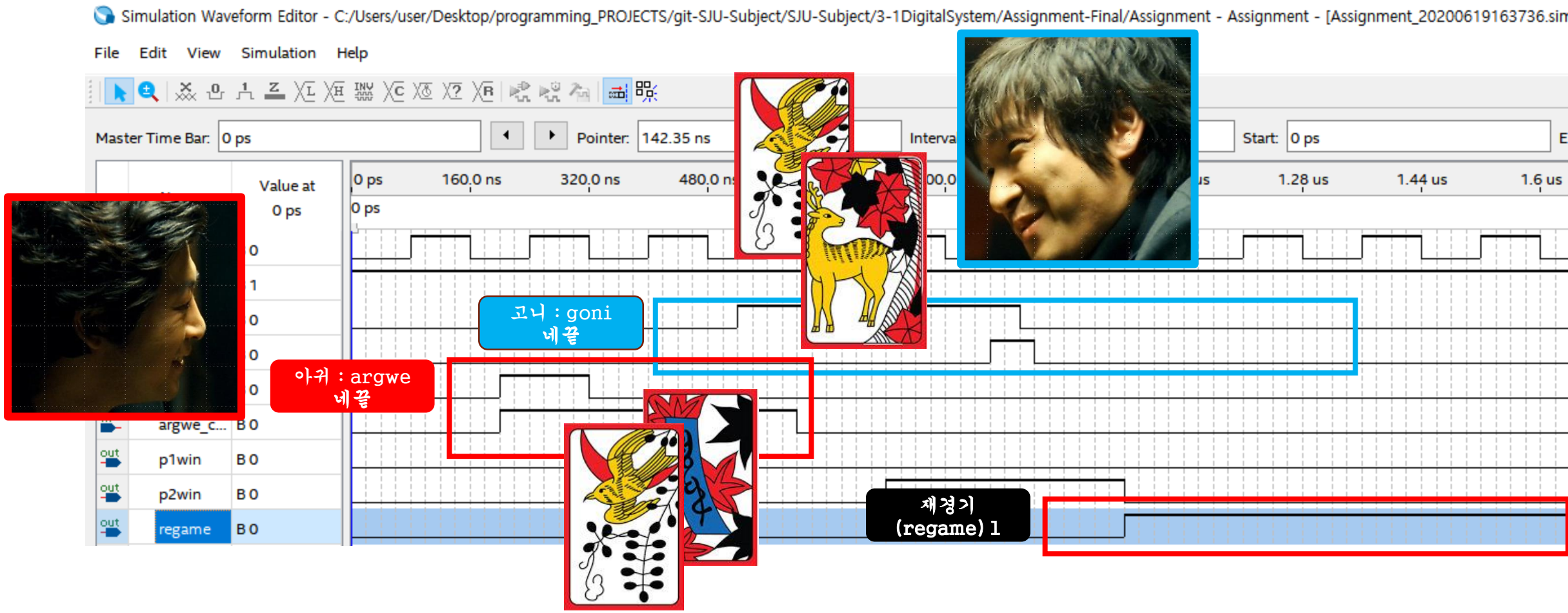
모듈 : 두 플레이어 카드 비교기 : 예시

- 예제 : 오랫만에 장땡을 잡은 고니

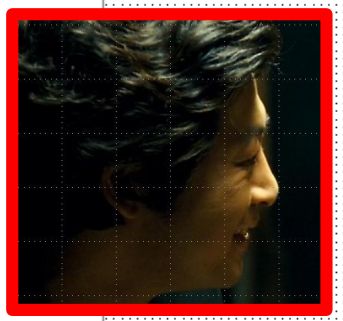


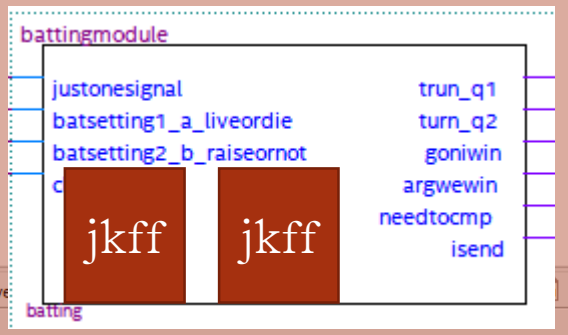
모듈 : 두 플레이어 카드 비교기 : 예시

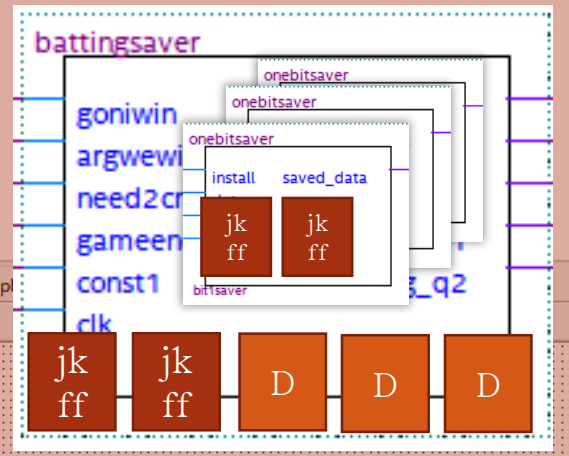
- 예제 : 숙적



전체 시스템 구조







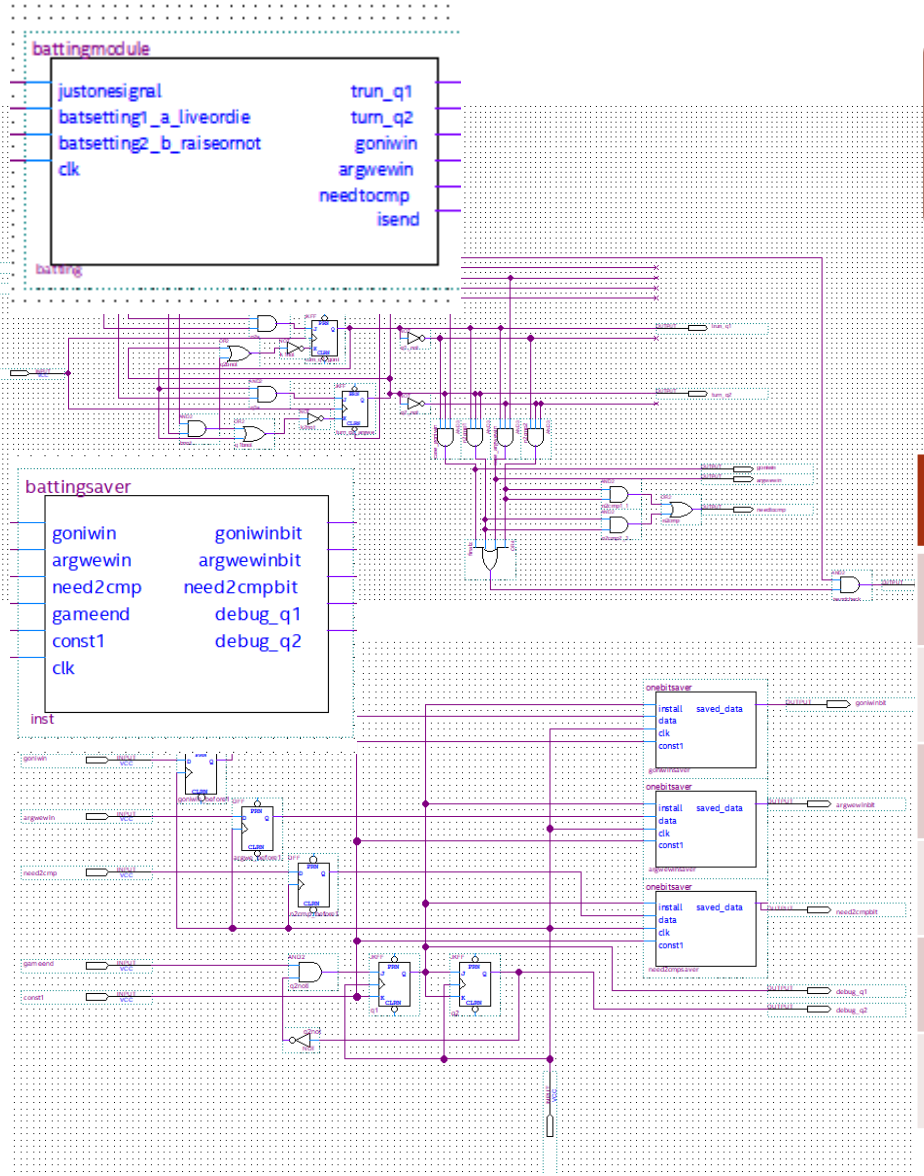
이 모듈들에서 플립플롭이 특히 많은 이유는, 배팅 상황을 기록하는 도중 의미있는 정보들이 나오는데 그것이 휘발성이라 상태를 통해 보존할 필요가 있었음.

모듈 : 두 플레이어 배팅/저장기 : 설명

모듈 입력

두 플레이어가 배팅하는 상황을 상정하고, 입력을 받는다.

입력	설명
Live_or_die	고니가 항상 먼저 시작한다는 제한 조건 하에, 1 rising bit 일 때를 기준으로 번갈아가며 배팅함.
Raise_or_not	
justonesignal	배팅을 시작하기 전까지 1을 유지시켜줘서 초기상태를 유지하도록 하는 bit



모듈 출력

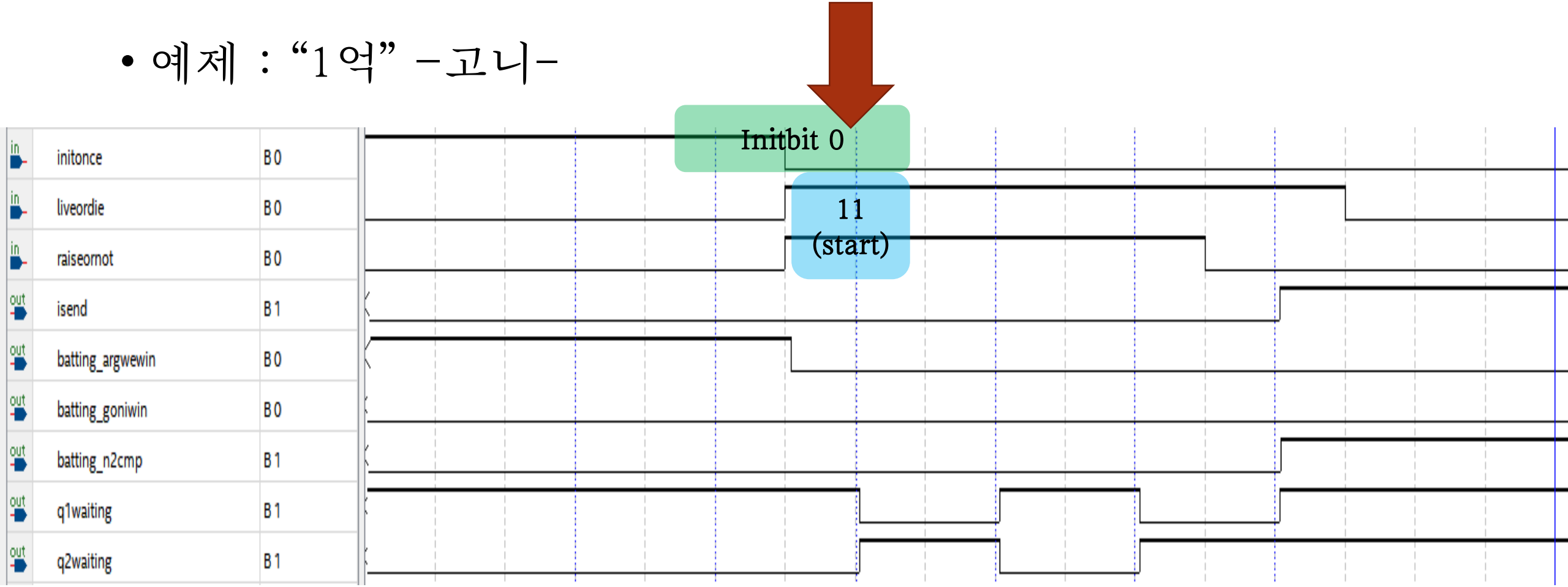
배팅 상황에 대해서 종합적으로 출력한다.

goniwin	고니가 이길 때 set
argwewin	아귀가 이길 때 set
need2cmp	단순히 배팅만 봐서는 “콜” 등으로 인해 패를 보아야 하는 상황
Isend	배팅이 끝나면 set 을 유지
q1waiting	아귀가 고니의 배팅을 기다리고 있는 상태
q2waiting	고니가 아귀의 배팅을 기다리고 있는 상태

모듈 : 두 플레이어 배팅기 : 예시

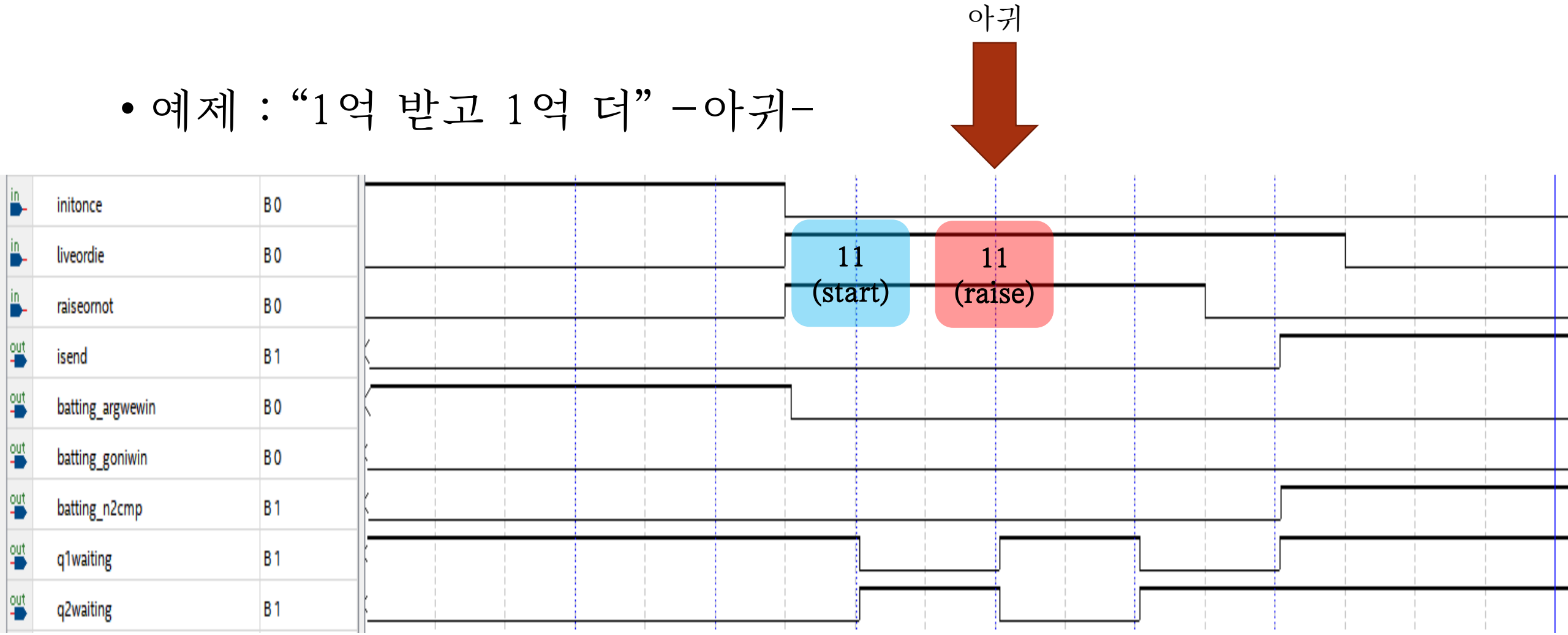
시작, 고니. 시작은 initbit 0, 11로 합니다.

- 예제 : “1억” –고니–



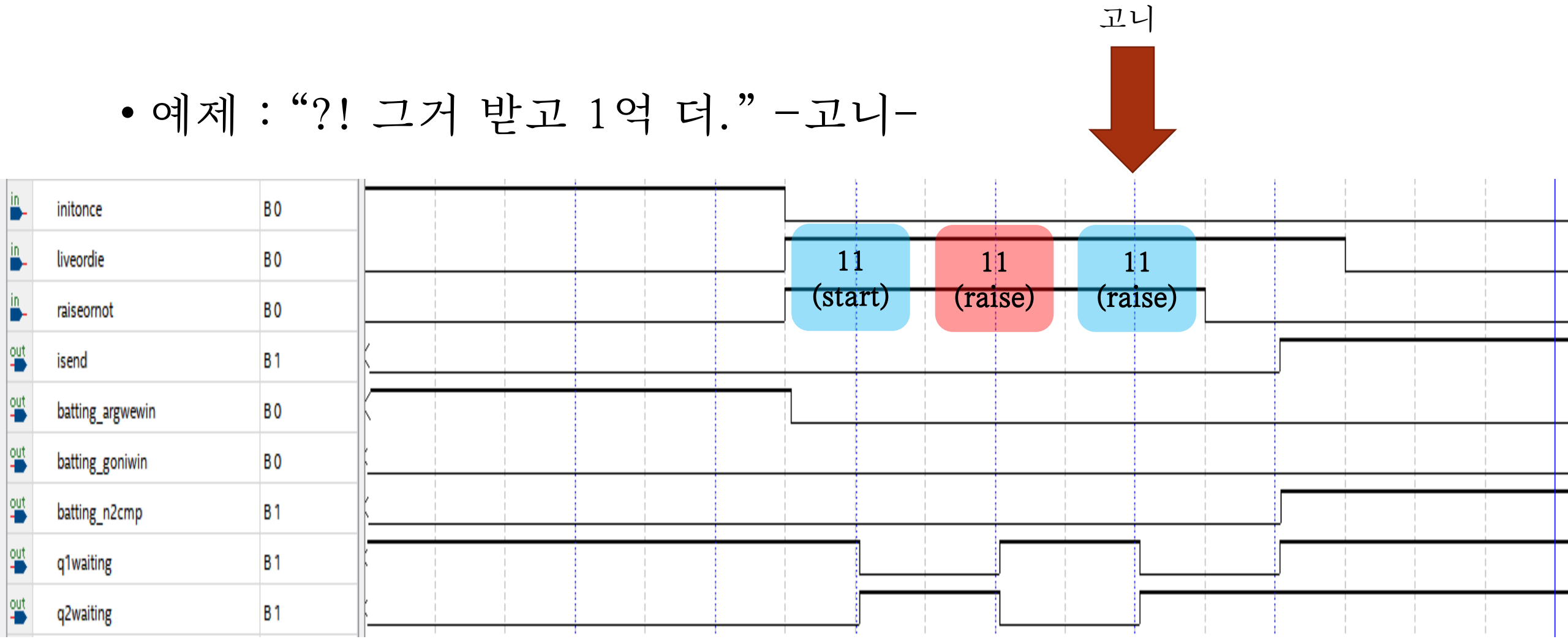
모듈 : 두 플레이어 배팅기 : 예시

- 예제 : “1억 받고 1억 더” –아귀–



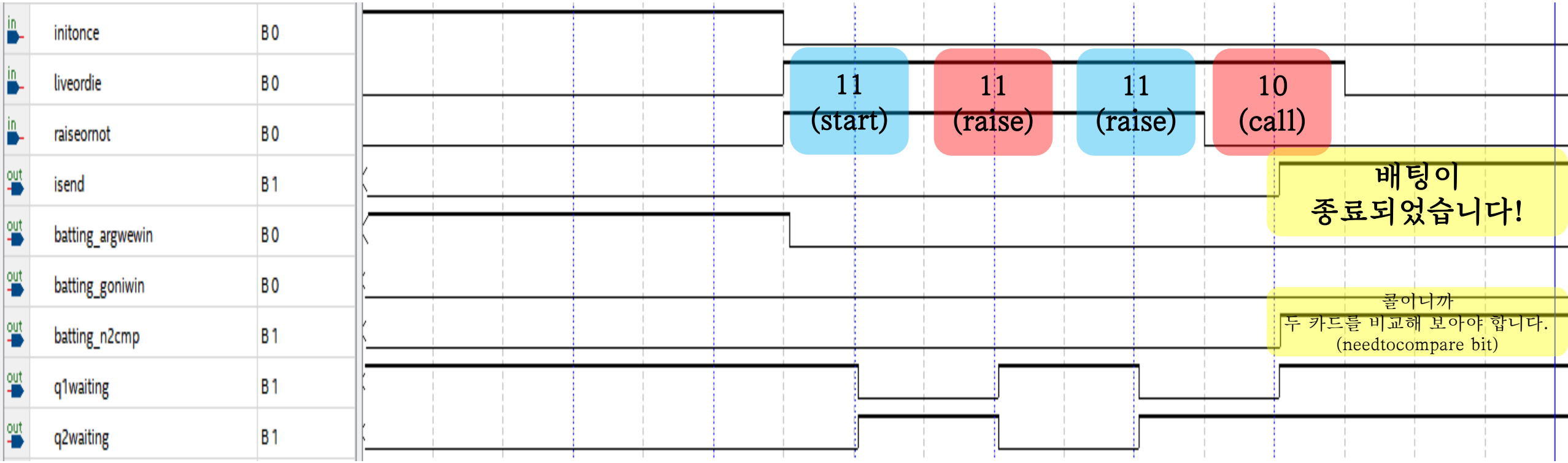
모듈 : 두 플레이어 배팅기 : 예시

- 예제 : “?! 그거 받고 1억 더.” –고니–

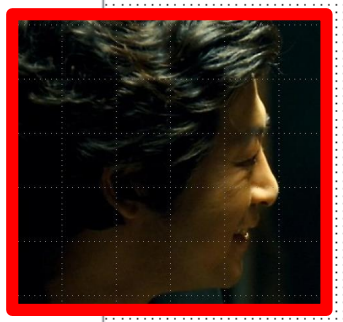


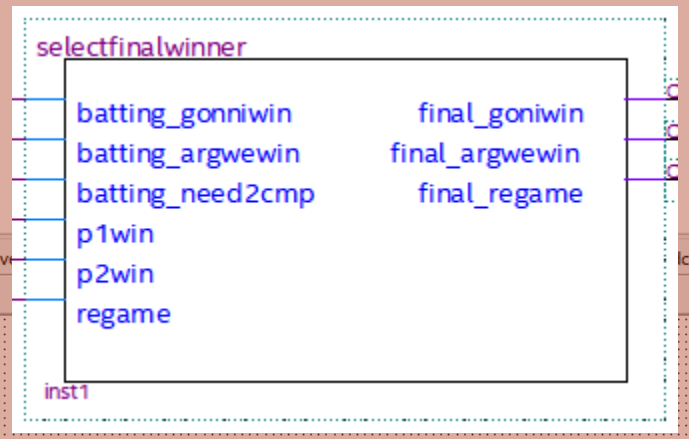
모듈 : 두 플레이어 배팅기 : 예시

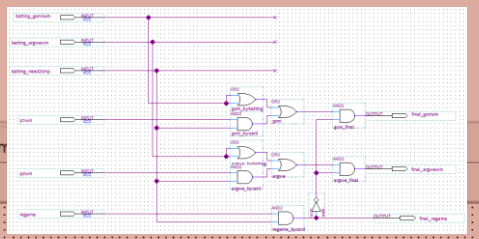
- 예제 : “콜. 까볼까?” –아귀–



전체 시스템 구조




selectfinalwinner
batting_gonniwin
batting_argwewin
batting_need2cmp
p1win
p2win
regame
final_goniwin
final_argwewin
final_regame
inst1



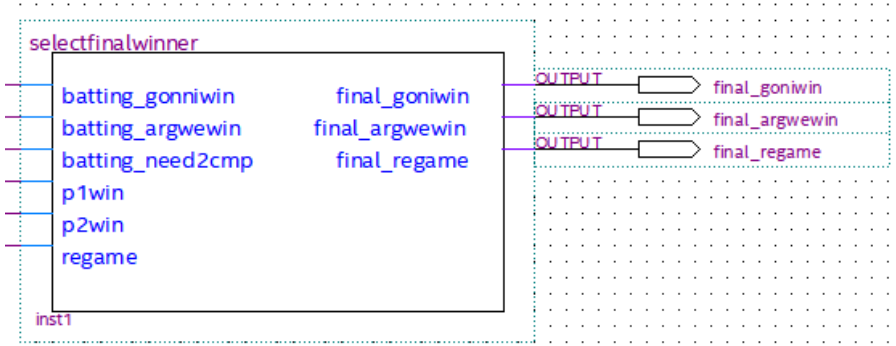
고니 카드 해독기
goni
아귀 카드 해독기
argwe
카드
비교기
최종
비교기
배팅기

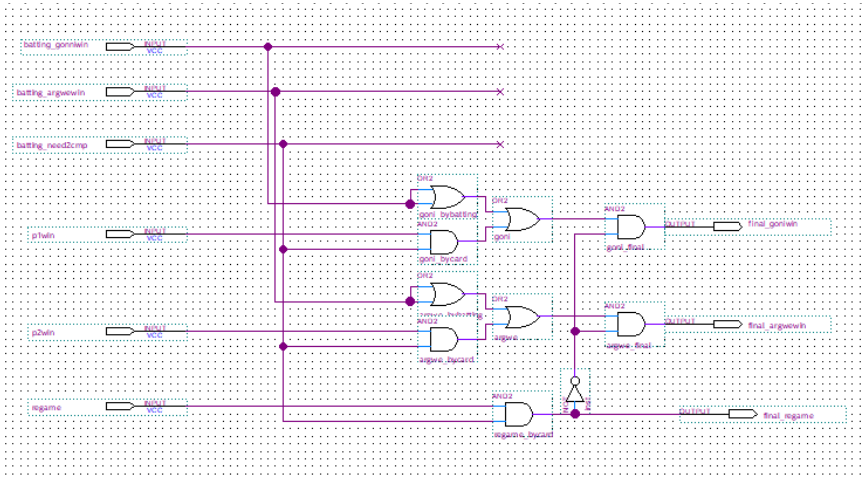
모듈 : 최종 출력 : 모든 경우 대변

모듈 입력

두 플레이어의 패, 배팅 상황을 종합적으로 입력

입력	설명
배팅에 대한 정보 Prefix : batting	배팅에서 고니가 이겼나? 아귀가 이겼나? 카드비교 해봐야 아나?
패에 대한 정보 No prefix	패의 정보를 바탕으로 만 산출된 결과로써, 카드 비교가 필요할 때 이용하기 위해서 넣어 줌.





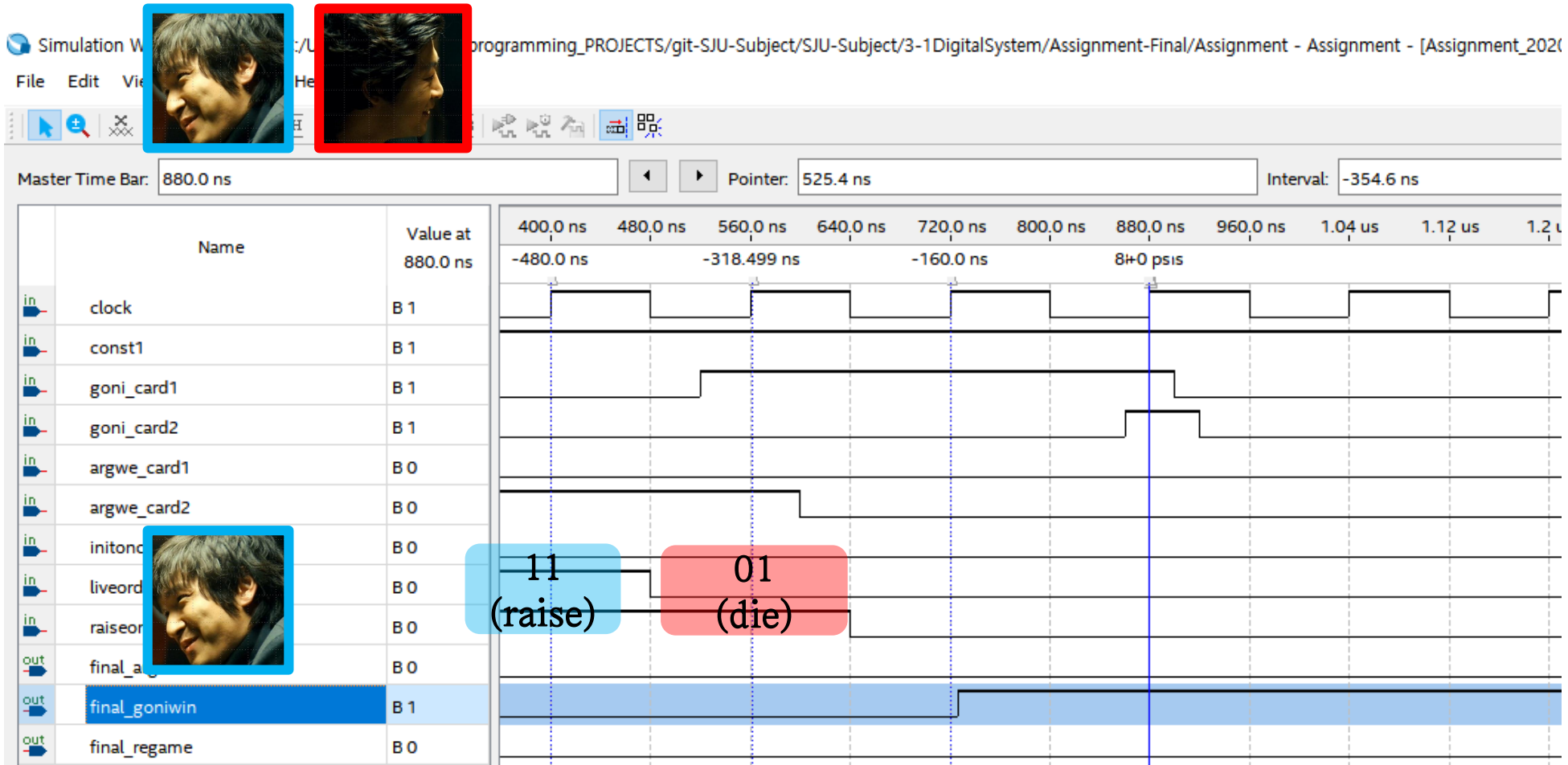
모듈 출력

모든 것을 고려했을 때 최종적 승리한 플레이어를 선택함

출력	설명
Final_goniwin	고니가 이길 때 set
Final_argwewin	아귀가 이길 때 set
Final_regame	까봤는데 둘의 패가 같거나, 까봤는데 한명 이상이 사구일 때 set

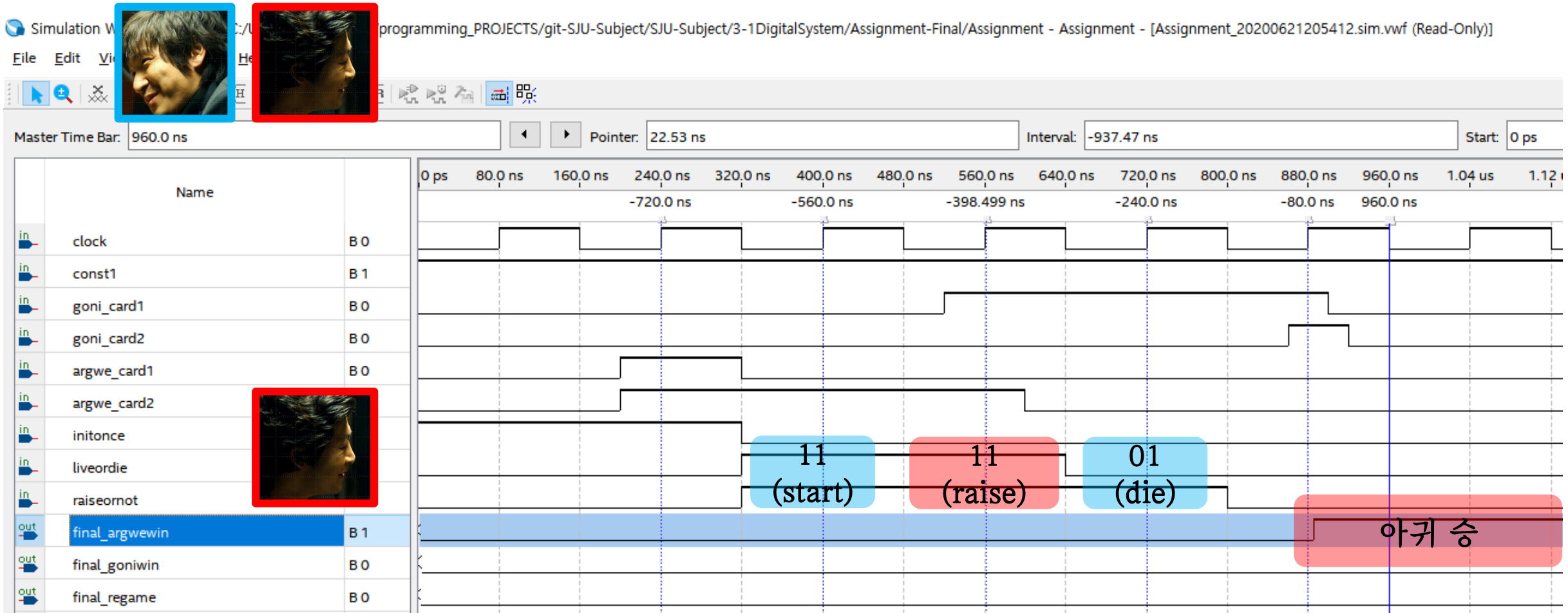
시나리오 : 끝 동점, 아귀 “뒤져버렸어”

- 4, 10 | 4, 10 | ... rasie & argwe_die



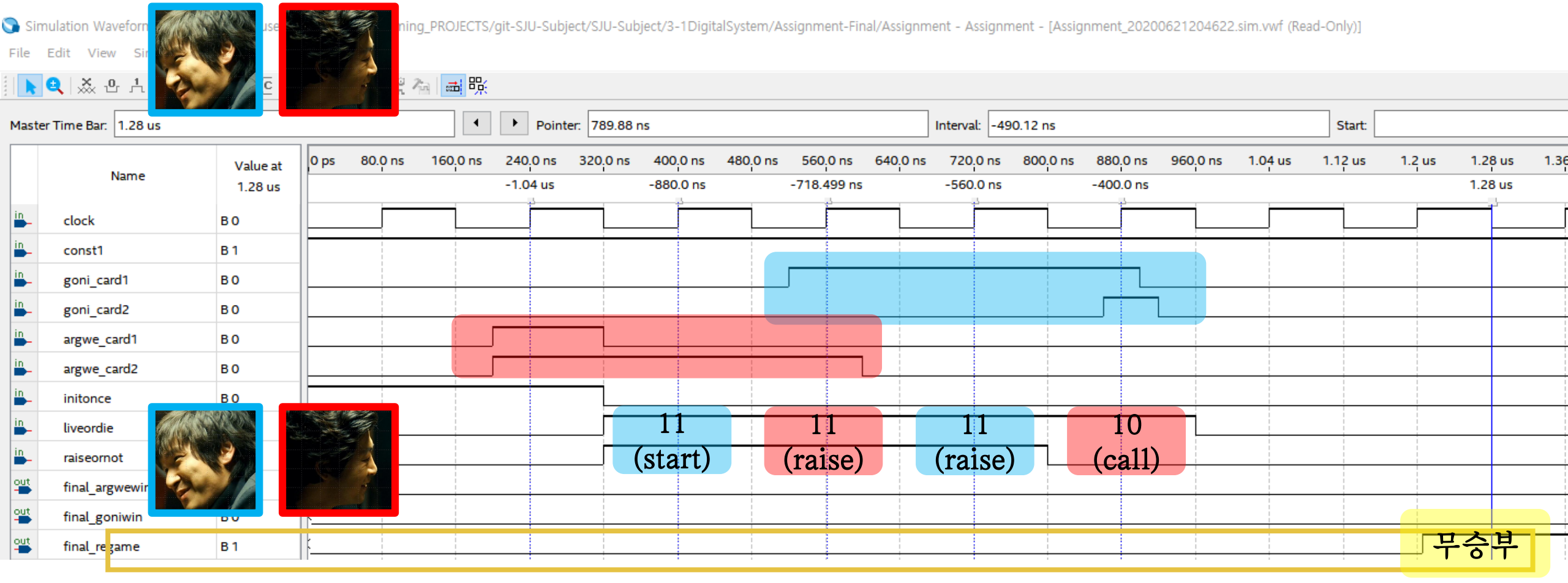
시나리오 : 끝 동점, 고니 “다이”

- 4, 10 | 4, 10 | gonistart & raise & raise & goni_die



시나리오 : 끝 동점, “까볼까?”

- 4, 10 | 4, 10 | gonistart & raise & call



시나리오 : 끝 싸움, “까불까?”

- 10, 4 | 10, 9 | gonistart & raise & raise & call

Simulation Waveform ... _PROJECTS/git-SJU-Subject/SJU-Subject/3-1DigitalSystem/Assignment-Final/Assignment - Assignment - [Assignment_20200622011859.sim.vwf (Read-Only)]

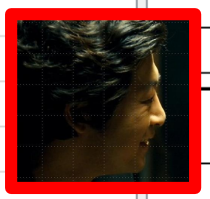
File Edit View Sim

Master Time Bar: 1.44 us | Pointer: 111.97 ns | Interval: -1.33 us | Start: 0 ps

Name	Value at 1.44 us
clock	B 0
const1	B 1
goni_card1	B 0
goni_card2	B 0
argwe_card1	B 0
argwe_card2	B 0
initonce	B 0
liveordie	B 0
raiseornot	B 0
final_argwewin	B 1
final_goniwin	B 0
final_regame	B 0

0 ps, 160.0 ns, 320.0 ns, 480.0 ns, 640.0 ns, 800.0 ns, 960.0 ns, 1.12 us, 1.28 us, 1.44 us

-1.2 us, -1.04 us, -878.499 ns, -720.0 ns, -560.0 ns, -400.0 ns, 1.44 us

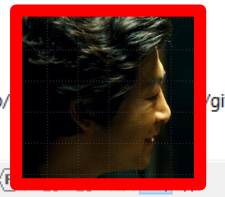


11 (start) 11 (raise) 11 (raise) 10 (call)

아귀 승

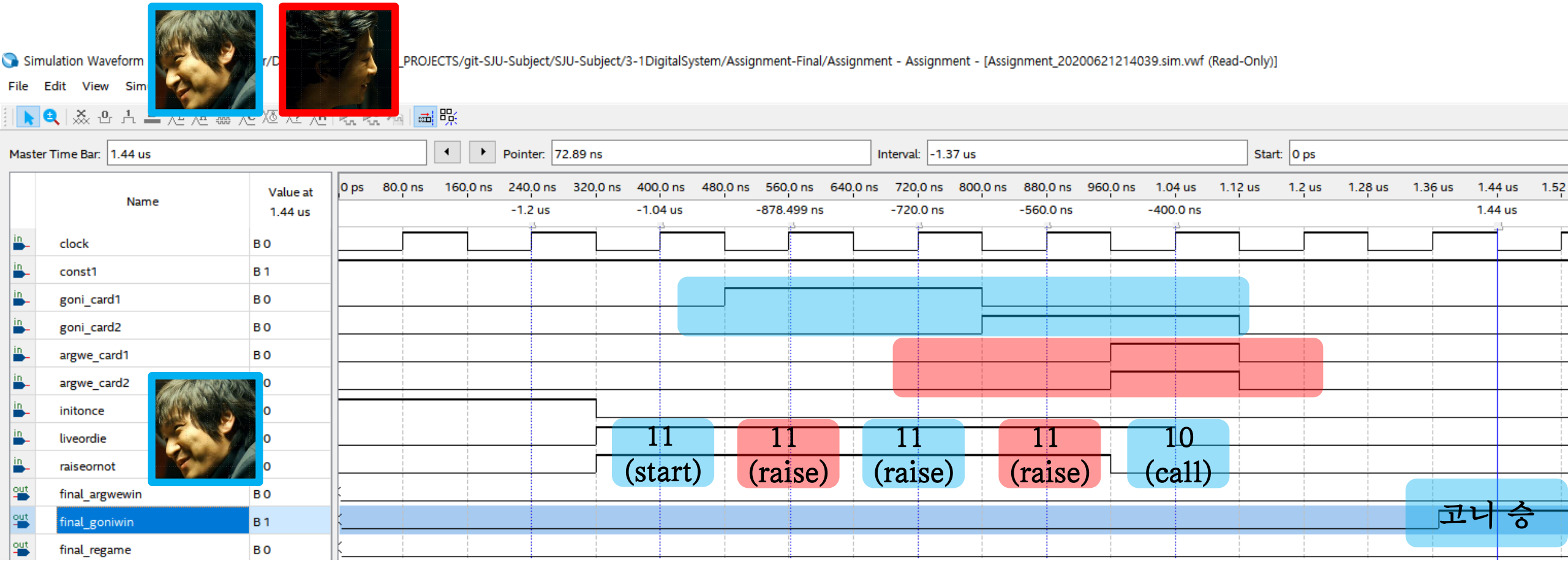
시나리오 : 땡 vs 끝, “까볼까?”

- 1, 1 | 4, 10 | gonistart & raise & raise & raise & call

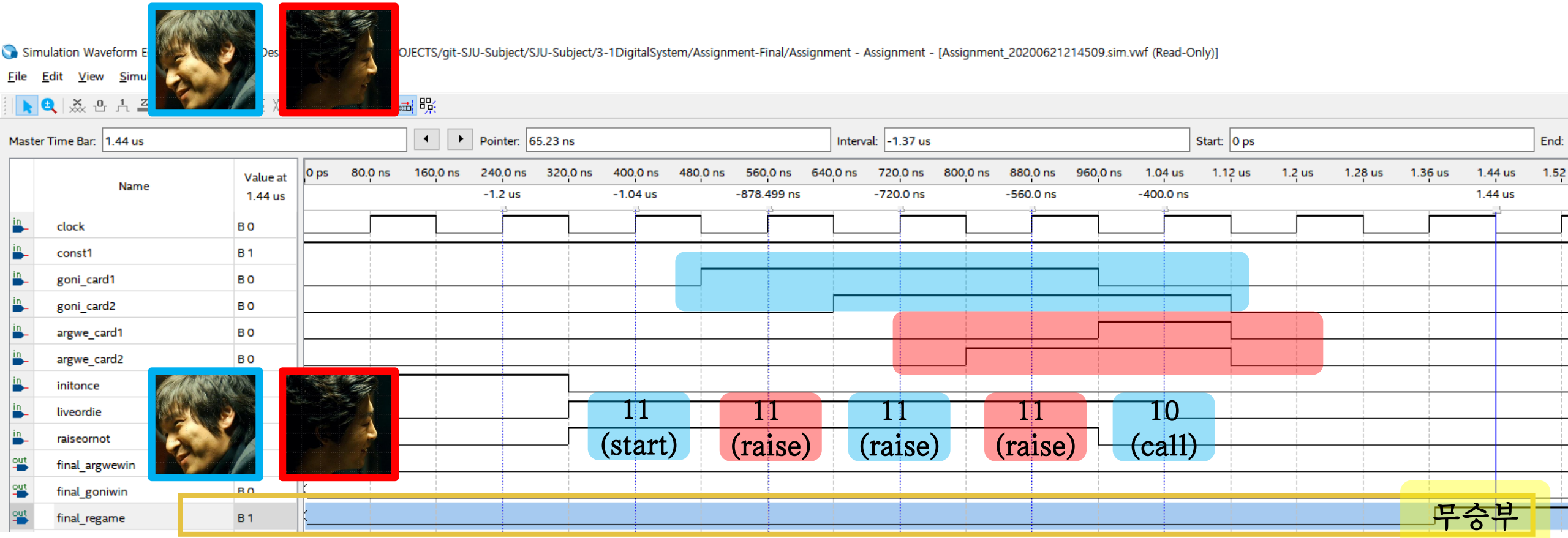
시나리오 : 땡 vs 땡, “까볼까?”

- 9, 9 | 1, 1 | gonistart & raise & raise & raise & call



시나리오 : 장땡 vs 49파토, “까볼까?”

- 10, 10 | 4, 9 | gonistart & raise & raise & raise & call



기타

- 굳이 그럴 필요는 없지만, 카드를 4, 9, 10 이외에 추가하기 용이하도록 설계함.
  - Eg : 3개 카드로 나올 수 있는 모든 경우의 수를 고려하여 하나의 함수식으로 간소화시키는 대신 ROM 과 같은 구조를 사용하여 설계함.
- 시스템에서 카드를 드로우하기까지 뜸을 들일 수 있도록 하고, 최종 결과 확정 전까지 해당 signal 에 아무런 변동이 없도록 만드는 것이 매우 어려웠음.
- 구성 요건에 맞추어 플립플롭들의 input 을 만들어내는 것도 어려웠지만, 더욱 어려운 것은 “필요한 상태를 정확히 정의하는 일” 이었고, 실제로도 이 단계에 훨씬 더 많은 시간이 필요했음.

느낀 점

- 순차회로에서 타이밍 시뮬레이션이 얼마나 중요하고 좋은 기능인지 느끼게 됨.
- 경우가 너무 많아서 어느정도 수준이 넘어가면, 정말 잘 정의되어 있는 모듈이 아니라면 생각을 하기 어렵고 생각조차 하기 싫게 될 수 있다는 것을 알게 됨.
- 쓸데없는 클럭들 노이즈를 모두 없앨 수 있도록 설계하는 것은 매우 생각하기 어려움.
- 간단한 시스템일 줄 알았지만, 생각보다 많이 복잡함.
- 손가락이 10개여서 감사함.