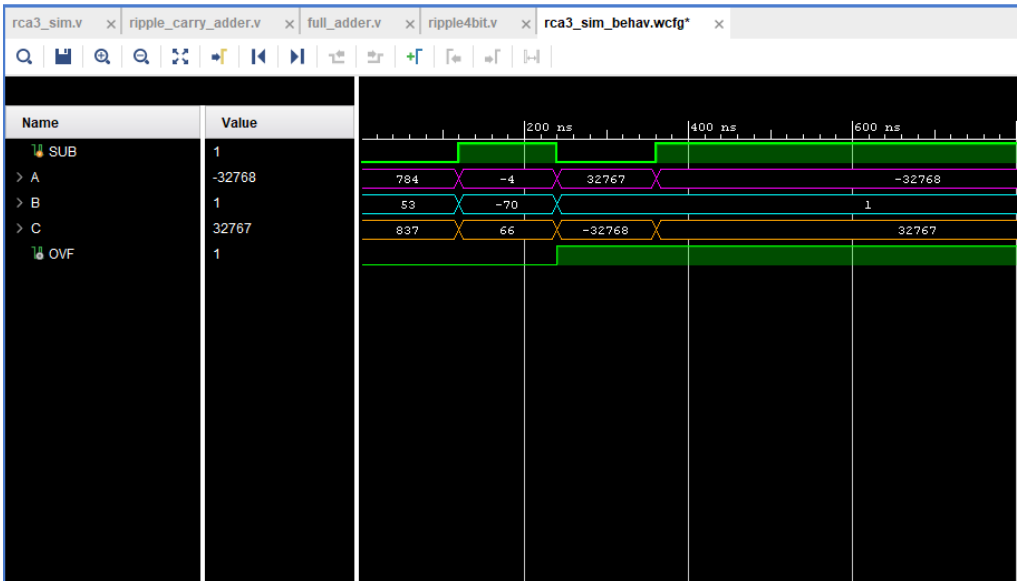
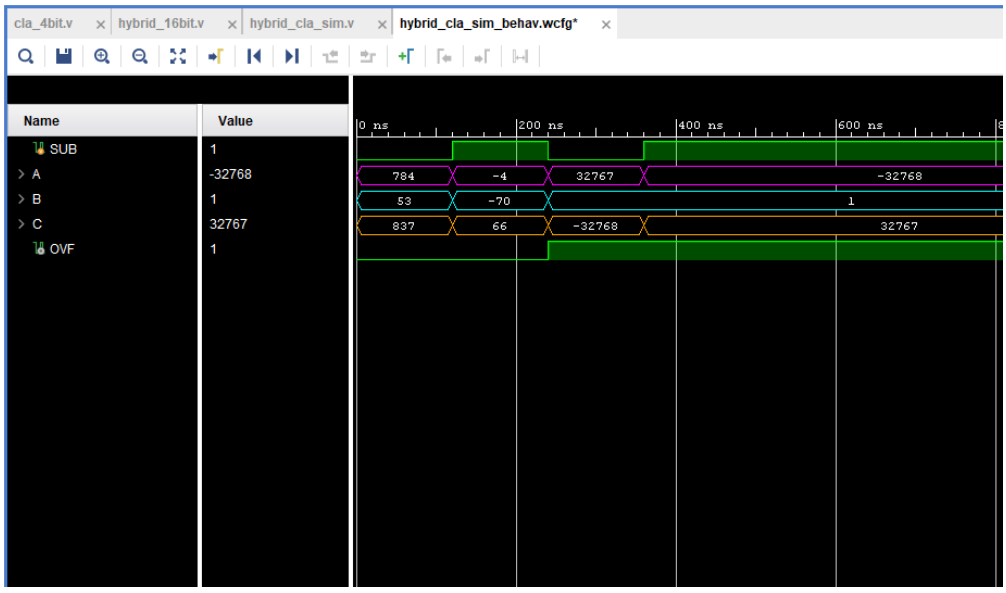
For 16-bit ripple-carry adder-subtractor:

1. Input A = 784, Input B = 53, Operation: Addition → Output C = 837, Overflow = 0
2. Input A = -4, Input B = 70, Operation: Subtraction → Output C = 66, Overflow = 0
3. Input A = 32767, Input B = 1, Operation: Addition → Output C = -32768, Overflow = 1
4. Input A = -32768, Input B = 1, Operation: Subtraction → Output C = 32767, Overflow = 1



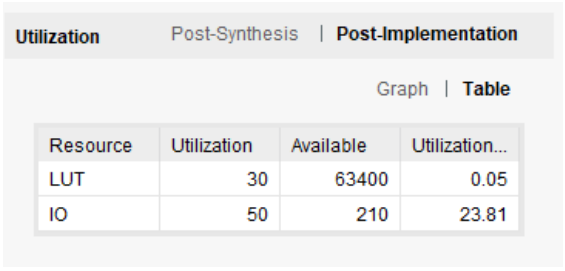
For 16-bit hybrid adder-subtractor using four 4-bit carry lookahead adders:

5. Input A = 784, Input B = 53, Operation: Addition → Output C = 837, Overflow = 0
6. Input A = -4, Input B = -70, Operation: Subtraction → Output C = 66, Overflow = 0
7. Input A = 32767, Input B = 1, Operation: Addition → Output C = -32768, Overflow = 1
8. Input A = -32768, Input B = 1, Operation: Subtraction → Output C = 32767, Overflow = 1



- Ripple-carry adder is better in terms of area, but the hybrid implementation using four 4-bit carry-lookahead adders is faster since there is no need to wait for computing carry_j in order to compute carry_i, where i > j.

- Below, you can see post-implementation project summaries for **16-bit hybrid adder-subtracter using four 4-bit CLA:**

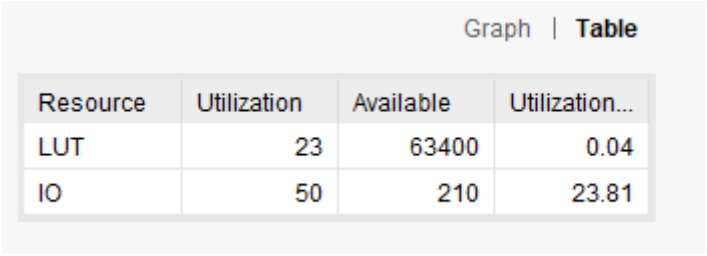


Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization...
LUT	30	63400	0.05
IO	50	210	23.81

- Below, you can see post-implementation project summaries for **16-bit ripple-carry adder using full-adders:**



Graph | **Table**

Resource	Utilization	Available	Utilization...
LUT	23	63400	0.04
IO	50	210	23.81

  - These project summaries indeed verify the fact that ripple-carry adder requires less area.