Pranav Eada

SJ DD Onboarding Writeup

Screenshots when running make verify_onboarding:

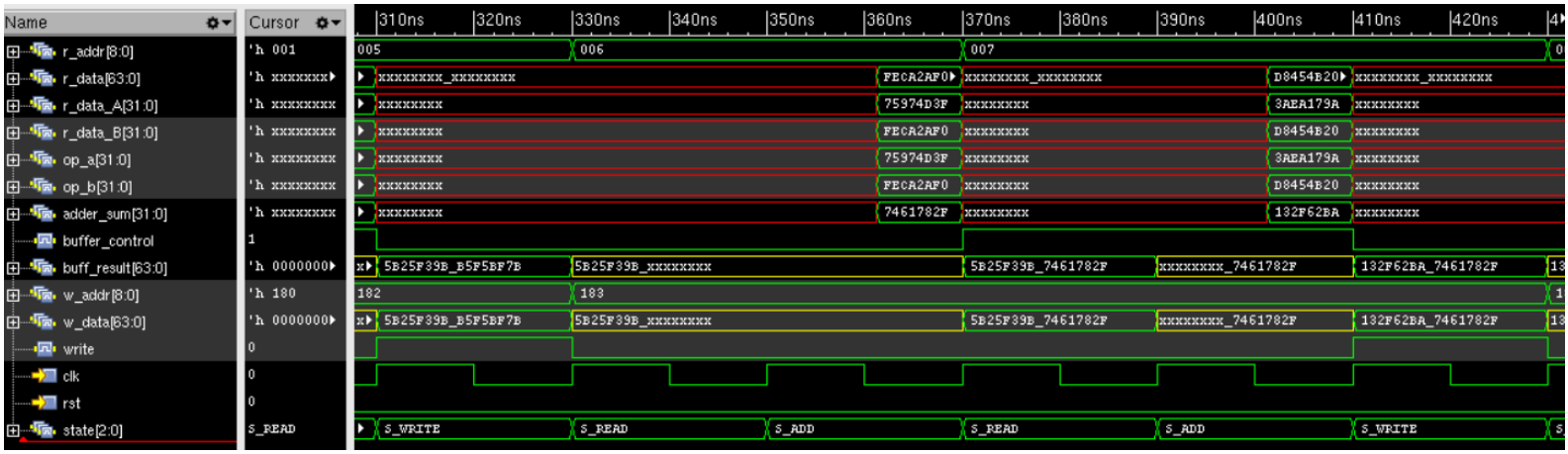
```
peada3@ece-linlabsrv01>make verify_onboarding
make clean
make[1]: Entering directory '/nethome/peada3/Digital-Design-Onboarding/sim/behav'
make[1]: Leaving directory '/nethome/peada3/Digital-Design-Onboarding/sim/behav'
python3.12 ../../scripts/init_mem.py /nethome/peada3/Digital-Design-Onboarding/sim/behav/WORKSPACE
make xrun
make[1]: Entering directory '/nethome/peada3/Digital-Design-Onboarding/sim/behav'
looking in file Include/calculator.include
cd /nethome/peada3/Digital-Design-Onboarding/sim/behav/WORKSPACE && \
xrun -64bit -sv -linedebug -access +rwc -timescale 1ns/10ps +define+SIM=1 +define+INST_FILE=\"\" +testname= +incdir+sym_links -coverage all -li
cqueue -covoverwrite -f sym_links/sim_no_path.include \
     -logfile simulation.log
TOOL:   xrun(64)        25.03-s003: Started on Sep 10, 2025 at 19:20:12 EDT
xrun(64): 25.03-s003: (c) Copyright 1995-2025 Cadence Design Systems, Inc.
xrun: *W,BADPRF: The -linedebug option may have an adverse performance impact.
file: sym_links/calculator_pkg.sv
        package worklib.calculator_pkg:sv
                errors: 0, warnings: 0
file: sym_links/adder32.sv
        module worklib.adder32:sv
                errors: 0, warnings: 0
file: sym_links/controller.sv
        module worklib.controller:sv
                errors: 0, warnings: 0
file: sym_links/full_adder.sv
        module worklib.full_adder:sv
                errors: 0, warnings: 0
file: sym_links/result_buffer.sv
        module worklib.result_buffer:sv
                errors: 0, warnings: 0
file: sym_links/sky130_sram_2kbyte_1rw1r_32x512_8.sv
        module worklib.sky130_sram_2kbyte_1rw1r_32x512_8:sv
                errors: 0, warnings: 0
file: sym_links/tb_calculator.sv
        module worklib.tb_calculator:sv
                errors: 0, warnings: 0
file: sym_links/top_lvl.sv
        module worklib.top_lvl:sv
                errors: 0, warnings: 0
xmvlog: *W,SPDUSD: Include directory sym_links given but not used.
        Total errors/warnings found outside modules and primitives:
                errors: 0, warnings: 1
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Top level design units:
                calculator_pkg
```

```
        Extracting FSMs for coverage:
                worklib.full_adder
                worklib.controller
                        FSM extracted for state register state
                worklib.sky130_sram_2kbyte_1rw1r_32x512_8
                worklib.adder32
                worklib.result_buffer
                worklib.top_lvl
                worklib.tb_calculator
        Total FSMs extracted = 1
        Building instance overlay tables: ........
xmelab: *W,COVFDP: FSM description matching across instance of a module for "COVFDM" warning has been moved from elaboration to simulation dumping. To restore old matching behav
"set_backward_compat -COVFDM_during_elab". Also, it is recommended to use set_parameterized_module_coverage CCF command to avoid coverage loss due to FSM description difference
stances of a module.
xmelab: *W,COVFHT: FSM hold transitions (transitions to the current state) are not extracted for any FSM in default mode.
............ Done
        Enabling instrumentation for coverage types: block expression FSM toggle functional
xmelab: *W,COVDCL: By default expression coverage is scored only for Verilog logical operators (|| and &&) and VHDL logical operators (OR, AND, NOR, and NAND), and is scored onl
ition expressions. To score coverage for other operators and for expressions in other statements, use the "set_expr_coverable_operators" and "set_expr_coverable_statements" cove
iguration file commands with suitable options at elaboration.
        Generating native compiled code:
                worklib.result_buffer:sv <0x64fea7c1>
                        streams:   5, words:  2968
                worklib.top_lvl:sv <0x2955175c>
                        streams:  33, words: 12168
                worklib.tb_calculator:sv <0x4c3a69c3>
                        streams:  12, words: 15528
                worklib.full_adder:sv <0x047a4f99>
                        streams:   1, words:   863
                worklib.adder32:sv <0x02735639>
                        streams:   2, words:  3510
                worklib.controller:sv <0x4c8fba68>
                        streams:  10, words: 17581
                worklib.sky130_sram_2kbyte_1rw1r_32x512_8:sv <0x0b7b34ed>
                        streams:   8, words: 17168
xmelab: *W,COVNOEN: By default, toggle coverage is not supported for systemverilog enumerated nets and variables. To enable toggle coverage of these objects, specify the 'set_t
ing -sv_enum' command in the coverage configuration file.
  reg [DATA_WIDTH-1:0]    mem [0:RAM_DEPTH-1];
                                |
xmelab: *W,COVMDD (./sym_links/sky130_sram_2kbyte_1rw1r_32x512_8.sv,52|28): Toggle coverage for bit, logic, reg, wire, enum and struct multi-dimensional static arrays and vector
supported by default. To enable toggle coverage for enum multi-dimensional static arrays specify 'set_toggle_scoring -sv_enum enable_mda' and for other multi-dimensional static
pecify 'set_toggle_scoring -sv_mda [<max_bit_base2_exponent>] [-sv_mda_of_struct]' ccf command in the coverage configuration file.
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                                  Instances  Unique
```

```
                                  Instances  Unique
                Modules:                  8       7
                Verilog packages:         1       1
                Registers:               65      53
                Scalar wires:            85       -
                Vectored wires:          37       -
                Always blocks:           13       8
                Initial blocks:           2       2
                Parallel blocks:          1       1
                Cont. assignments:       36       5
                Pseudo assignments:      38       -
                Simulation timescale:  10ps
        Writing initial simulation snapshot: worklib.full_adder:sv
Loading snapshot worklib.full_adder:sv ..................... Done
xcelium> source /tools/software/cadence/xcelium/latest/tools/xcelium/files/xmsimrc
xcelium> run

-------------Beginning Simulation!-------------

Time:                    0
-------------Initializing Signals---------------

Time:                 1000

-------------Finished Simulation!----------------

Time:              1293000
Simulation complete via $finish(1) at time 12930 NS + 0
./sym_links/tb_calculator.sv:52         $finish;
xcelium> exit

coverage setup:
  workdir  :  ./cov_work
  dutinst  :  full_adder(full_adder)
  dutinst  :  tb_calculator(tb_calculator)
  scope    :  scope
  testname :  test

coverage files:
  model(design data) :  ./cov_work/scope/icc_75c54b2d_00000000.ucm
  data               :  ./cov_work/scope/test/icc_75c54b2d_00000000.ucd
TOOL:   xrun(64)        25.03-s003: Exiting on Sep 10, 2025 at 19:20:13 EDT  (total: 00:00:01)
make[1]: Leaving directory '/nethome/peada3/Digital-Design-Onboarding/sim/behav'
python3.12 ../../scripts/check_onboarding.py /nethome/peada3/Digital-Design-Onboarding/sim/behav/WORKSPACE/memory_post_state_lower.txt /nethome/peada3/Digital-Design-Onboarding/
/WORKSPACE/sim_memory_post_state_lower.txt
PASSED: RTL simulation output matches expected results.
python3.12 ../../scripts/check_onboarding.py /nethome/peada3/Digital-Design-Onboarding/sim/behav/WORKSPACE/memory_post_state_upper.txt /nethome/peada3/Digital-Design-Onboarding/
/WORKSPACE/sim_memory_post_state_upper.txt
PASSED: RTL simulation output matches expected results.
```

The two red checkmarks indicate contents of lower and upper SRAM memory blocks match the expected results when doing RTL simulation.

Waveforms demonstrating a full adding operation



R_addr first loads the value from address 006 and splits it into r_data_A (lower 32 bits) and r_data_B (upper 32 bits). Then, these values are summed using op_a and op_b signals (which basically store r_data_A and r_data_B respectively). The sum is stored in adder_sum. The summing operation for address 006 was 0x75974D3F + 0xFECA2AF0 = 0x7461782F. Then when buffer control goes high (and also on the rising clock edge), this value is stored in the lower 32 bits of buff_result and w_data. Then, the loading and summation happens for the value at address 007 and this time addr_sum stores the sum 0x132F62BA. During this, the upper 32 bits in w_addr and buff_result are cleared at the next rising clock edge to make space for the new summed value. Then, at the rising clock edge and when write goes high, the new sum is written to the upper 32bits of w_data and buff_result and written back into SRAM memory. At the bottom, the states are also transitioning from Read -> Add (lower 32 bits), back to Read->Add (for upper 32 bits) and then Write (to store and write back value into SRAM memory)Hence, these signals and waveforms demonstrate accurate functionality of the different controller modules.

Explanation of States and Correctness

Our calculator design is functionally correct because the controller state machine, adder, result buffer, and SRAM modules work together to implement a full add operation: load → add lower → add upper → combine → store.

- S_IDLE initializes the read and write pointers and prepares the buffer select.
- S_READ asserts the read signal and fetches 64-bit operands from memory (split across SRAM A and B).

- S_ADD feeds the operands into the 32-bit ripple-carry adder. On the first pass, the lower 32-bit sum is written into the result buffer; on the second pass, the upper 32-bit sum is written. The buffer control signal ensures each half is placed in the correct location.
- S_WRITE asserts the write signal and writes the full 64-bit result from the buffer back into memory at the designated address. Both SRAM A (lower half) and SRAM B (upper half) are updated in parallel.
- S_END halts the controller once all input pairs have been processed.

The adder32 module is implemented as a 32-bit ripple-carry adder, chaining full adders to compute each sum. The result_buffer correctly collects lower and upper halves into a single 64-bit word. By sequencing through these states, the calculator successfully performs operand loading, arithmetic addition, result combination, and memory storage. This demonstrates that the datapath and FSM operate correctly as a calculator.