My Test Plan

Test Name	Sequence Name	Test Objective	Sequence Steps	Coverage
store_fetch	vseq_store_fetch	Verify that store and fetch command are working properly for all registers	01. store random value to register n 02. fetch value from all registers one by one 03. check that register n has the correct value 04. check that other registers were not affected	cmd(9) x r1 x data_in cmd(10) x d1
walking_1s_0s	vseq_walking_1s_0s	Verify all the bits for all the registers	01. store value = walking_1s to register n 02. fetch value from all the registers one by one 03. check that register n has the correct value 04. repeat storing walking 1s for all registers 05. store value = walking_0s to register n 06. fetch value from all the registers one by one 07. check that register n has the correct value 08. repeat storing walking 0s for all registers	cmd(9) x r1 x data_in cmd(10) x d1
arith_cmds	vseq_arith_cmds	Verify arithmetic cmds: add, subtract, shift left, shift right	01. store random value to reg_op1(d1) and reg_op2(d2) 02. send arith cmd to work on reg_op1, reg_op2 and save result in reg_out 03. fetch value from reg_out 04. check that result is as expected	d1 x d2 x cmd op1 x op2 x cmd out_resp x out_data
branch	vseq_branch	Verify branch cmds: branch if zero, branch if equal	01. fetch value from reg n, say value = x 02. store random value to reg_op1 and reg_op2 03. send branch cmd to work on reg_op1, reg_op2 04. send store cmd: value = x=1 to reg n 05. fetch value frmo reg n 06. check that branch worked as expected. --- i.e., if branch happened, value = x else value = x+1	cmd(12) x d1 x (op1 == 0 | op1 != 0) cmd(13) x d1 x d2 x (op1 == op2 | op1 != op2)
overflow	vseq_overflow	Verify addition with overflow	01. randomize op1 and op2 so that their sum is greater than 32'hFFFFFFFF 02. store op1 to reg_op1(d1) and op2 to reg_op2(d2) 03. send add cmd to work on reg_op1, reg_op2 and save result in reg_out 04. fetch value from reg_out 05. check that result is as expected 06. check that out_resp == 2	d1 x d2 x cmd op1 x op2 x cmd out_resp(2) x out_data
underflow	vseq_underflow	Verify subtraction with underflow	01. randomize the value of op2 to be greater than op1 02. store op1 to reg_op1(d1) and op2 to reg_op2(d2) 03. send sub cmd to work on reg_op1, reg_op2 and save result in reg_out 04. fetch value from reg_out 05. check that result is as expected 06. check that out_resp == 2	d1 x d2 x cmd op1 x op2 x cmd out_resp(2) x out_data
reset	vseq_reset	Verify that calc works normal after reset	01. run arith_cmds seq 02. apply reset 03. run arith_cmds seq 04. apply reset 05. run branch seq 06. apply reset 07. run branch seq	~
ordering_rules	vseq_ordering_rules	Verify that ordering rules are followed. Reference calc3 spec for details.	01. send add cmd to work on reg_op2, reg_op2 and save result in reg_op1 02. store random value to reg_op1 03. store random value to reg_op2 04. send arith cmd to work on reg_op1, reg_op2 and save result in reg_op1 05. fetch value from reg_op1 06. check that fetched value in step5 is expected (according to step4)	~
ral	vseq_ral	To store and fetch values to/from registers using RAL	01. write random value to reg n 02. read value from reg n 03. check that value written and read are same	cmd(9) x r1 x data_in cmd(10) x d1