Sl No.	Block Name	Top Parameters	Verified Parameters	Convergence	Block Lvl Covergae	Coverage from Top	Comments
1	ecc_add_sub_mod_alter	REG_SIZE: 384	REG_SIZE: 384	Converged	100%	100%	Stopats on add_en_i and sub_i on task ecc_Add_sub_mod_alter
2	ecc_pe_first	RADIX: 32	RADIX: 4	Converged	100%	100%	
3	ecc_pe	RADIX: 32	RADIX: 32	Converged	100%	100%	
4	ecc_pe_final	RADIX: 32	RADIX: 32	Converged	100%	100%	
5	ecc_montgomerymultiplier	REG_SIZE: 384 RADIX: 32	REG_SIZE: 16,48 RADIX: 4	No convergence	100%		Block lvl coverage achieved on 16bit
6	ecc_fau	REG_SIZE: 384 RADIX: 32	REG_SIZE: 384 RADIX: 32	converged	100%		Only glue logic verified and the coverage is for only glue logic
7	ecc_ram_tdp_file	ADDR_WIDTH: 6 DATA_WIDTH: 384	ADDR_WIDTH: 6,3 DATA_WIDTH: 384,32	Converged	100%	89.47%	Implicit if for ena and enb as both were hardcoded
8	ecc_pm_sequnecer			Converged	99.98%	99.98%	Stoptats in the task for addra driven using formal counter, coverage to be run under stimuli engine
9	ecc_pm_ctrl	Secp384_SCA_MONT_COUNT: 576 Secp384_MONT_COUNT: 384 MULT_DELAY: 39 ADD_DELAY: 1	Secp384_SCA_MONT_COUNT: 576,2 Secp384_MONT_COUNT: 384,2 MULT_DELAY: 39,3 ADD_DELAY: 1	No convergence			no pm_seq Deadcode of stalled all the counter set,coverage achieved with top
10	ecc_arith_unit	REG_SIZE: 384 RND_SIZE: 192 RADIX: 32 ADDR_WIDTH: 6 p_prime: 384'hfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffeffffffff0000000000000000ffffffff p_mu: 32'h00000001 q_grouporder: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973 q_mu: 32'he88fdc45	REG_SIZE: 384 RND_SIZE: 192 RADIX: 32 ADDR_WIDTH: 6 p_prime: 384'hfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffeffffffff0000000000000000ffffffff p_mu: 32'h00000001 q_grouporder: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973 q_mu: 32'he88fdc45	converged	100%		
11	ecc_scalar_blinding	REG_SIZE: 384 RND_SIZE: 192 RADIX: 32 GROUP_ORDER: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973	REG_SIZE: 32 RND_SIZE: 16 RADIX: 8 GROUP_ORDER: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973	converged	98.28%		Deadcode and open outputs of an instantiation
12	ecc_hmac_drbg_interface	REG_SIZE: 384 GROUP_ORDER: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973 LFSR_INIT_SEED: 148'h6_04E7_A407_54F1_4487_A021_11AC_D0DF_8C55_57A0	REG_SIZE: 384 GROUP_ORDER: 384'hffffffffffffffffffffffffffffffffffffffffffffffffc7634d81f4372ddf581a0db248b0a77aecec196accc52973 LFSR_INIT_SEED: 148'h6_04E7_A407_54F1_4487_A021_11AC_D0DF_8C55_57A0	No convergence	97.1%		by adding stopat on hmac_drbg instantiation, en and counter nonce,Deadcode
13	ecc_dsa_sequencer			converged	99.26%	99.26%	mplicit ena, as it is constraine
14	ecc_dsa_ctrl			No convergence		100%*	*with waivers of deadcode,kv clients

fv_ecc_dsa_sequencer

Sl No.	Property	Proof status on Top	status block	Description	Additional notes	Can add to top
1	reset_p	hold	hold	When reset_n deasserts or zeroize triggers then the output douta should be equal to zero		Yes
2	const_p	NA	hold	Checks the sequence where some constant values are written into the memory	Driven by an internal counter in formal setup so that every step in the operation just takes a single clock cycle	No
3	keygen_p	NA	hold	Checks the sequence of operations for the keygen operation	Driven by an internal counter in formal setup so that every step in the operation just takes a single clock cycle	No
4	sign_p	NA	hold	Checks the sequence of operations for the signing operation	Driven by an internal counter in formal setup so that every step in the operation just takes a single clock cycle	No
5	verify_p	NA	hold	Checks the sequence of operations for the verifying operation	Driven by an internal counter in formal setup so that every step in the operation just takes a single clock cycle	No
6	when_nop_both_id_addr_0_p	hold	hold	If the first field is DSA_UOP_NOP, the next fields should be NOP_ID and UOP_OPR_DONTCARE, respectively.		yes
7	when_wr_both_id_addr_not_0_p	hold	hold	If the first field is DSA_UOP_WR_CORE, the next fields should include a valid ID and ADDRESS, respectively.		yes
8	when_drbg_both_id_addr_0_p	hold	hold	If the first field is DSA_UOP_HMAC_DRBG, the next fields should be NOP_ID and UOP_OPR_DONTCARE, respectively.		yes
9	when_scalar_both_id_addr_p	unknown	hold	If the first field is DSA_UOP_SCALAR_SCA, the next fields should be SCALAR_G_ID and UOP_OPR_DONTCARE, respectively.		yes
10	when_wr_scalar_both_id_addr_p	unknown	hold	If the first field is DSA_UOP_WR_SCALAR, the next fields should be SCALAR_ID/ SCALAR_G_ID/ SCALAR_G_ID and UOP_OPR_DONTCARE, respectively.		yes
11	when_commands_both_id_addr_0_	unknown	hold	If the first field is DSA_UOP_KEYGEN/ DSA_UOP_KEYGEN/ DSA_UOP_VERIFY0-2, the next fields should be NOP_ID and UOP_OPR_DONTCARE, respectively.		yes
12	when_rd_both_id_addr_not_0_p	unknown	hold	If the first field is DSA_UOP_RD_CORE, the next fields should include a valid ID and ADDRESS, respectively.		yes
13	illegal_addr_p	unknown	hold	If the addra input has illegal addresses then the output douta should be zero		yes

fv_add_sub_alter

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	Validtates when reset_n is not triggered or the zeroize is set, ready is deasserted.	output is not zero when reset occurs as it depends on the combinatorial registers, added a constraint that all inputs would be zero after reset on block level, this constraint should be converted into assertion on top level	Yes
2	sub_p	Unknown	holds	Property to check (a-b)%c, where the case it holds is when we have add_en pulse,sub_i for 3 cycles	For commands constraint is in place in block level, should be convereted into assertion on top level	Yes
3	add_p	Unknown	holds	Property to check (a+b)%c, where the case it holds is when we have add_en pulse,no sub_en_i for 3 cycles	For commands constraint is in place in block level, should be convereted into assertion on top level	Yes
4	no_cmd_p	holds	holds	Property to check the if there isn't any cmd for 2 consecutive cycles then the res would have previous valueand ready should be deasserted	res_o, the output is removed from commitment as the same reason as reset	Yes

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	When !reset_n or zeroize or start_in then all the outputs are set to zero	All the proof Convergence is achieved on reduced parameter of RADIX=4	Yes
2	aout_p	holds	holds	When its odd and no start then a out takes the previous a_in value		Yes
3	mout_p		holds	When its odd and no start then m out takes the computed value		Yes
4	aout_even_p	holds	holds	When its even and no start then a out takes the 2cyc previous a_in value		Yes
5	mout_even_p	Unknown	holds	When its even and no start then mout takes the previous mout value		Yes
6	cout_odd_p	Unknown	holds	When its odd and no start then cout takes the computed value from previous carry sum and computed m		Yes
7	cout_even_p	Unknown	holds	When its even and no start then cout takes the computed value from previous carry sum and computed m		Yes

fv_pe

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	When !reset_n or zeroize or start_in then all the outputs are set to zero		Yes
2	aout_p	holds	holds	aout when odd takes the previous value of a_in		Yes
3	mout_p	holds	holds	mout when odd takes the previous value of m_in		Yes
4	aout_even_p	holds	holds	aout when even holds the previous value		Yes
5	mout_even_p	holds	holds	mout when even holds the previous value		Yes
6	cout_odd_p	Unknown	holds	cout when odd MSB (a_in*b_in)+(p_in*m_in)+c_out+s_in		Yes
7	cout_even_p	Unknown	holds	cout when even MSB a_in*b_in)+(p_in*m_in)+c_in+s_out		Yes
8	sout_even_p	holds	holds	sout when even LSB a_in*b_in)+(p_in*m_in)+c_in+s_out		Yes
9	sout_odd_p	holds	holds	sout when odd LSB (a_in*b_in)+(p_in*m_in)+c_out+s_in		Yes

fv_pe_final

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	When !reset_n or zeroize or start_in then all the outputs are set to zero		Yes
2	s_out_odd_p	holds	holds	When and no start_in then s_out is equal to addition of c_out, s_in, multiplication of a_in, b_in and multiplication of p_in,m_in		Yes
3	s_out_noodd_p	holds	holds	When and no start_in then s_out is equal to addition of c_in, s_out, multiplication of a_in, b_in and multiplication of p_in,m_in		Yes
4	c_out_odd_p	holds	holds	When and no start_in then c_out is equal to most significant radix bits addition of c_out, s_in, multiplication of a_in, b_in and multiplication of p_in,m_in		Yes
5	c_out_noodd_p	Unknown	holds	When and no start_in then c_out is equal to most signification radix bits addition of c_in, s_out, multiplication of a_in, b_in and multiplication of p_in,m_in		Yes

fv_montmultiplier and fv_montmultiplier_glue

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	when !reset_n or zeroize then the output is zero and ready is asserted		Yes
2	mult_p	NA	No convergence	Complete proof for the whole design encapsulated output =(opa*opb*R_INV)%prime, where R_INV is calculated based on R = 2^((ceil(bits(n_i)/RADIX)+1)*RADIX) R^(-1) = R invmod n_i;	The proof check is done only with 16,48 bit versions. For top parameters it is unlinkely to compute as the multipliaction crosses the boundary of 2**10. And constraint is in place for the command start to trigger only after the mult_delay+2 here it is 41 or would say whenever the ready is high. To enable 16bit version define macro FOR48 if not 48 bit version is automatically selected.	No
3	no_ready_p	Unknown	holds	ready signal deasserts once start is triggered and stays deasserted for required number of clock cycles.		No

Following proofs are white box checks on full version

4	compare_p	Unknown	holds	Once the start triggers then after a delay of 28 cycles i.e DLY_CONCAT because to push the whole operand a sequentially would require 26 cycles and 2 extra cycles for valid computation to receive on first box from next cycle on as each block of PE works parallely on 2 RADIX elements, next computed value is again on first_box later the sequence continues until the last block i.e the carry bits are stored and then reduced by prime	Black boxed the ecc_mult_dsp	Yes
5	when_odd_a_array_shifts_p	Unknown	holds	a_reg shifts by RADIX, when odd and no start	Black boxed the ecc_mult_dsp	Yes
6	when_even_a_array_stable_p	Unknown	holds	a_reg stable if no odd	Black boxed the ecc_mult_dsp	Yes
7	reg_set_start_p	Unknown	holds	reg's set once start is triggered	Black boxed the ecc_mult_dsp	Yes
8	reg_no_start_p	Unknown	holds	reg's stay stable if no start cmd	Black boxed the ecc_mult_dsp	Yes
9	when_odd_b_p_array_p	Unknown	holds	b_array and p_array are 64bits takes MSB 32 if odd from b_reg and p_reg	Black boxed the ecc_mult_dsp	Yes
10	when_even_b_p_array_p	Unknown	holds	b_array and p_array are 64bits takes LSB 32 if even from b_reg and p_reg	Black boxed the ecc_mult_dsp	Yes
11	s_in_routing_p	Unknown	holds	connections for the s_in's of the block	Black boxed the ecc_mult_dsp	Yes
12	a_in_routing_p	Unknown	holds	connections for the a_in's of the pe blocks	Black boxed the ecc_mult_dsp	Yes
13	m_c_in_routing_p	Unknown	holds	connections for the m_in's,c_in's of the pe blocks	Black boxed the ecc_mult_dsp	Yes

fv_ecc_fau

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	Validtates when reset_n is not triggered or the zeroize is set, outputs are deasserted.	add_res_o is not taken into commitments because being a combinatorial output could have some value in there. This at block level, the same commitment can be enabled at top level where the inputs are set to zero from top, using macro TOP.	Yes
2	mult_pulse_p	holds	holds	When ever mult_en_i is triggered, it would just generate one pulse		Yes
3	add_pulse_p	unknown	holds	When ever add_en_i is triggered, it would just generate one pulse		Yes
4	no_add_edge_p	holds	NA	Once edge triggered from next cycle on it stays out until there is an another add cmd	Holds on block lvl with until_with not with s_until_with as consequent part is PI	Yes
5	no_mult_edge_p	holds	NA	Once edge triggered from next cycle on it stays out until there is an another mult cmd	Holds on block lvl with until_with not with s_until_with as consequent part is PI	Yes
6	garbage_ready_p	holds	holds	When add and mult ready		Yes
7	outputs_p	holds	holds	Primary outputs connected to primary outputs of submodules		Yes
8	inputs_p	holds	holds	Primary inputs connected to primary inputs of submodules		Yes
9	data_out_prime_p	unknown	NA	Whenever a ready signal is asserted then the resulted value should be less than prime	Need to define macro TOP, so that this could be loaded at top level. This proof isn't used at block level as the subcomponents or msubmodules are black boxed	Yes

fv_ecc_ram_tdp_file

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	Validtates when reset_n is not triggered or the zeroize is set, outputs are deasserted.		Yes
2	sym_reset_mem_p	holds	holds	memory to be resetted on the reset sequence	Symbolic check is done, as the symbolic could take anyvalue of the address length and check for the cex	Yes
3	write_p	Unknown	holds	write on the respective port and any command on the other port, then memory should have the port data in		Yes
4	read_p	holds	holds	read on the respective and any command on the other port, then the data out should have the data from the memory of given port addr		Yes
5	no_enable_p	NA	holds	no cmd on the respective port then the out data and the memory of the respective symbolic addr stays the same		No
6	no_enable_ab_p	NA	holds	no cmd on both ports then the out data and the memory of the respective symbolic addr stays the same		No

fv_ecc_pm_sequencer

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	Validtates when reset_n is not triggered or the zeroize is set, outputs are deasserted.		Yes
2	initial_p	NA	holds	validates the sequence for first two steps		No
3	pm_init_G_p	NA	holds	Base point randamization and storing in R1		No
4	pm_init_S_p	NA	holds	Initialize R0 with Zero		No
5	point_add_p	NA	holds	validates the point addition sequence		No
6	point_dbl_p	NA	holds	validates the point double sequence		No
7	conv_p	NA	holds	validates the conversion sequence		No
8	sign0_p	NA	holds	validates the initial sign sequence (d + r (privKey-d)) and ((h-d) + r.d)		No
9	sign1_p	NA	holds	validates the final sequence of sign		No
10	verify0_p0_p	NA	holds	validates Verify part0 to convert inputs to the mont domain		No
11	verify0_p1_p	NA	holds	verify part1 use the generated s^-1 for the intermediate computations and convert them into normal domain so in next step they could be used in the point multiplication.		No
12	verify1_st_p	NA	holds	validates the sequenceverify1 (h*s^-1)*G		No
13	verify2_init_pk_p	NA	holds	validates the sequence verify2 Initialise with PubKey(PK) for (r*s^-1)*PK computation It is stored R1, but R1_z is initialised with one_mont		No
14	verify2_pointadd_p	NA	holds	validates the point addition of the verifying sequence		No
15	inv_modp_p	NA	holds	Validates the inverse modulo of p, sequence	Generated using the python script	No
16	inv_modq_p	NA	holds	Validates the inverse modulo of q, sequence	Generated using the python script	No
17	chk_pk_p	NA	holds	Validates the sequence to check the provided pubKey is on the curve or not.		No
18	store_not_same_addr_p	Unknown	holds	When storing both the address should not be the same		Yes
19	when_nop_both_addr_0_p	holds	holds	If it starts with UOP_NOP, the next both fields should be UOP_OPR_DONTCARE		Yes

20	when_add_addrb_0_p	Unknown	holds	If it starts with UOP_ST_ADD_p/UOP_ST_ADD_q, the last field should be UOP_OPR_DONTCARE.		Yes
21	when_mult_addra_0_p	holds	holds	If it starts with UOP_ST_MUL_p/UOP_ST_MUL_q the middle field should be UOP_OPR_DONTCARE		Yes
22	when_do_add_mul_sub_addr_not_zero_p	holds	holds	If it starts with UOP_DO_MUL_p/UOP_DO_MUL_q/ UOP_DO_ADD_p/ UOP_DO_ADD_q/ UOP_DO_SUB_p/UOP_DO_SUB_q, the next both fields shouldn't be UOP_OPR_DONTCARE.	Need to exclude PM_INIT_S and PM_INIT_S+4 because const_zero is same as dontcare	Yes
23	illicit_addra_p		holds	Illegal address should result in zero		Yes
			These sequences can be verified from ctrl on top, but individual sequences verification is not possible without considering the delays as 1			

fv_ecc_pm_ctrl

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	Validtates when reset_n is not triggered or the zeroize is set, outputs are deasserted.		Yes
2	check_point_p	Unknown	holds	Validates once the check public key cmd is set then the sequence is triggered and finally ends in NOP	Checked with mult delay 3 and add_delay 1, can be added to top if delays are as choosen	Yes
3	pm_init_g_s_to_pm_init_s_p	Unknown	holds	validates once the cmd is set initally it would traverse through PM_INIT_G_S and PM_INIT_S		Yes
4	pm_init_s_to_pa_s_p	Unknown	holds	validates if cmd is ongoing then it would traverse from PM_INIT_S till PA_S		Yes
5	pa_s_to_pd_s_p	Unknown	holds	validates if cmd is ongoing then it would traverse from PA_S till PD_S		Yes
6	pd_s_to_pd_e_p	Unknown	holds	validates if cmd sequence is ongoing then it would traverse from PD_S to PD_E		
7	pd_e_to_invs_p	Unknown	holds	validates if cmd is ongoing then it would traverse from PD_E to INV_S when the mont_cntr is zero		Yes
8	pd_e_to_pa_s_p	Unknown	holds	validates if cmd is ongoing then it would traverse from PD_E to PA_S when the mont_cntr is not zero		Yes
9	invs_to_inve_p	Unknown	holds	validates if cmd is ongoing then it would traverse from INV_S to INV_E		Yes
10	inve_to_convs_p	Unknown	holds	validates if cmd is ongoing then it would traverse from INV_E to CONV_S		Yes
11	convs_to_nop	Unknown	holds	validates if cmd is ongoing then it would traverse from CONV_S to NOP		Yes
12	counter_keygen_a_liveness_p	Unknown	No convergence	Once keygen command triggered then eventually it should endup in any of the mentioned sequence steps	This is done symbollically by taking a counter_keygen_a as a variable which would take gereater than PM_INIT_S and less than CONV_E	Yes

13	counter_keygen_b_liveness_p	Unknown	No convergence	Once keygen command triggered then eventually it should endup in any of the mentioned sequence steps	This is done symbollically by taking a counter_keygen_a as a variable which would take greater than previous symbolicand less than CONV_E	Yes
14	keygen_order_check_a	Unknown	holds	This proves that the symbolic step of counter b should be only trigger once the counter a is done, as counter a is less than counter b sequence. This ensures that none of the steps jump over and follow the order		Yes
15	signing_stage3_2_p	Unknown	holds	validates if signing sequence is ongoing then it would traverse from CONV_S till INVq_S		
16	signing_stage4_1_p	Unknown	holds	validates if signing sequence is ongoing then it would traverse from INVq_S till INVq_E		Yes
17	signing_stage4_p	Unknown	holds	validates if signing sequence is ongoing then it would traverse from INVq_E till NOP		Yes
18	counter_sign_a_liveness_p	Unknown	No convergence	Once signing command triggered then eventually it should endup in any of the mentioned sequence steps		Yes
19	counter_sign_b_liveness_p	Unknown	No convergence	Once signing command triggered then eventually it should endup in any of the mentioned sequence steps		Yes
20	order_check_sign_a	Unknown	holds	This proves that the symbolic step of counter b should be only trigger once the counter a is done, as counter a is less than counter b sequence. This ensures that none of the steps jump over and follow the order		Yes
21						
22	verify_part0_stage0_p	Unknown	holds	Validates once the VER_PART0_CMD is triggered then it would traverse itll INVq_S		Yes
23	verify_part0_stage1_1_p	Unknown	holds	validates if the verify part0 is ongoing then it would traverse from INVq_s to INVq_E		Yes
24	verify_part0_stage1_p	Unknown	holds	validates if the verify part0 is ongoing then finally it would end in NOP		Yes
25	counter_ver_p0_a_liveness_p	Unknown	No convergence	validates that the prog_cntr eventually take the symbolic value once the cmd triggered	Symbolic variable in the range [INVq_S, INVq_E],	Yes

26	counter_ver_p0_b_liveness_p	Unknown	holds	validates that the prog_cntr eventually take the symbolic value once the cmd triggered	Symbolic variable in the range [VER0_P0_S, VER0_P0_E]v	Yes
27	counter_ver_p0_c_liveness_p	Unknown	No convergence	validates that the prog_cntr eventually take the symbolic value once the cmd triggered	Symbolic variable in the range [VER0_P1_S, VER0_P1_E]	Yes
28	counter_intrnl_ver0_p0_s_a	Unknown	holds	validates that the VERO_P0_S sequence order is correct		
29	counter_staging0_ver_p0_p	Unknown	holds	This proves that the symbolic step of counter a should be only trigger once the counter b is done, as counter b is less than counter a sequence. This ensures that none of the steps jump over and follow the order		Yes
30	counter_staging1_ver_p0_p	Unknown	No convergence	This proves that the symbolic step of counter c should be only trigger once the counter b and counter a is done, as counter b and counter a is less than counter c sequence. This ensures that none of the steps jump over and follow the order		Yes
31	verify_part1_stage0_p	Unknown	holds	validates once the verify part1 is set initally it would traverse through PM_INIT_G_S and PM_INIT_S		Yes
32	verify_part1_stage2_p	Unknown	holds	validates if verify part1 sequence is ongoing then it would traverse from PD_E to NOP when the mont_cntr is zero		Yes
33	counter_ver_p1_a_liveness_p	Unknown	No convergence	Once verify part1 command triggered then eventually it should endup in any of the mentioned sequence steps	This is done symbollically by taking a counter_keygen_a as a variable which would take gereater than PM_INIT_G_S and less than CONV_E	Yes
34	counter_ver_p1_b_liveness_p	Unknown	No convergence	Once verify part1 command triggered then eventually it should endup in any of the mentioned sequence steps	This is done symbollically by taking a counter_keygen_a as a variable which would take greater than previous symbolicand less than PD_E	Yes
35	counter_order_check_ver_p1_a	Unknown	holds	This proves that the symbolic step of counter b should be only trigger once the counter a is done, as counter a is less than counter b sequence. This ensures that none of the steps jump over and follow the order		Yes

36	verify_part2_stage0_p	Unknown	holds	validates once the verify part2 is set initally it would traverse through VER1_ST_S and PM_INIT_S		Yes
37	verify_part2_stage2_p	Unknown	holds	validates if verify part2 sequence is ongoing then it would traverse from PD_E to VER2_PA_S when the mont_cntr is zero		Yes
38	verify_part2_stage3_p	Unknown	holds	validates if verify part2 sequence is ongoing then it would traverse from INV_S till INV_E		No
39	verify_part2_stage3_p	Unknown	holds	validates if verify part2 sequence is ongoing then it would traverse from VER2_PA_S till INV_S		No
40	counter_ver_p2_a_liveness_p	Unknown	holds	Once verify part2 command triggered then eventually it should endup in any of the mentioned sequence steps	Symbolic variable in the range [VER1_ST_S, PM_INIT_PK_E],	Yes
41	counter_ver_p2_b_liveness_p	Unknown	No convergence	Once verify part2 command triggered then eventually it should endup in any of the mentioned sequence steps	Symbolic variable in the range [PM_INIT_S, PD_E]	Yes
42	counter_ver_p2_c_liveness_p	Unknown	No convergence	Once verify part2 command triggered then eventually it should endup in any of the mentioned sequence steps	Symbolic variable in the range [VER2_PA_S, VER2_PA_E]	Yes
43	counter_ver_p2_d_liveness_p	Unknown	No convergence	Once verify part2 command triggered then eventually it should endup in any of the mentioned sequence steps	Symbolic variable in the range [INV_S, CONV_E]	Yes
44	counter_intrnl_ver_p2_a	Unknown	No convergence	This proves that the symbolic step of counter a_1 should be only trigger once the counter a is done, as counter a is less than counter a_1 sequence. This ensures that none of the steps jump over and follow the order		
45	counter_staging0_ver_p2_p	Unknown	holds	This proves that the symbolic step of counter a should be only trigger once the counter a is done, as counter a is less than counter b sequence. This ensures that none of the steps jump over and follow the order		Yes

46	counter_staging1_ver_p2_p	Unknown	No convergence	This proves that the symbolic step of counter c should be only trigger once the counter b and counter a is done, as counter b and counter a is less than counter c sequence. This ensures that none of the steps jump over and follow the order		Yes
47	counter_staging2_ver_p2_p	Unknown	No convergence	This proves that the symbolic step of counter d should be only trigger once the counterc, counter b and counter a is done, as counter c,counter b and counter a is less than counter c sequence. This ensures that none of the steps jump over and follow the order		Yes
48	busy_p	holds	holds	If the prog_cntr is not NOP then the pm_ctrl is busy		Yes
49	no_busy_p	holds	holds	If the prog_cntr is NOP then the pm_ctrl is not busy		Yes
50	req_digit_p	holds	holds	req_digit_o is set only when the mont_cntr is not zero in PD_E step or the very first rotation i.e in PM_INIT_E		Yes
51	no_req_digit_p	holds	holds	req_digit_o is zero when the mont_cntr is zero in PD_E step or it is not the very first rotation i.e in PM_INIT_E		Yes
52	mont_multiplies_req_digit_o_p	Unknown	holds	req_digit_o trigger mont_cntr times		Yes
53	opcode_add_p	holds	holds	when output of pm_sequencer instruction is add I would be on primary output after 3 cycles and stays stable for the delay time +2		Yes
54	opcode_mul_p	holds	holds	when output of pm_sequencer instruction is multiplication I would be on primary output after 3 cycles and stays stable for the delay time +2		Yes
55	opcode_no_compute_p	holds	holds	When the add or mult are set then the next cycle output of pm_seq instruction is stored so it not lost during the exceution of add ormult and the next instr_o would be stored value		Yes

56	addr_when_add_sub_p		holds	when output of pm_sequencer instruction is add and the addresses would be on primary output after 3 cycles and stays stable for the delay time +2		Yes
57	addr_when_mult_p	holds	holds	when output of pm_sequencer instruction is multiplication and the addresses would be on primary output after 3 cycles and stays stable for the delay time +2	Can add to Top only if MULT delay is reduced to 3	Yes
58	addr_when_no_cmd_p	holds	holds	When the add or mult are set then the next cycle output of pm_seq instruction is stored so it not lost during the exceution of add or mult and the next instr_o would be stored value	Constraint on digit_i stability can be changed as assertion on the top	Yes
59	instr_o_when_cmds_2cycls_p	Unknown	holds	inst_o stays stable for the next cycle when the cmd is triggered		
60	instr_o_when_no_cmds_2cycls_p	Unknown	holds	inst_o stays stable for the next cycle when the cmd is triggered and no stall		
61	prog_cntr_stable_p	Unknown	holds	when stalled the counter stays stable		Yes
62	prog_cntr_change_p	Unknown	holds	When stalled and the delay counters are set to zero then the prog_cntr changes		Yes
63	prog_cntr_change_1_p	Unknown	holds	When prog_cntr is not in NOP and there is no stall then the cntr changes		Yes
64	no_cmd_prog_nop_p	holds	holds	When no command is at the input then the prog_cntr will hold on to NOP if it in NOP already		Yes

fv_ecc_arith_unit

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	hold	hold	During reset or zeroize all the internal regand outputs are set to zero	At block level pm_ctrl is blackboxed, so the busy_o port is free port could take any value so for that an additional constraint is in place but should be removed when loaded form the top	Yes
2	dout_p		hold	data_o takes the value of the opb_s, which is from the ram module when rd_reg_i is set		Yes
3	no_dout_p	hold	hold	data_o sets to zero when rd_reg_i is not set		Yes
4	busy_mux	hold	hold	If busy then the mux has the inputs for ram as addr and mult_we from pm_ctrl and din is the result of multiplication		Yes
5	not_busy_addr_mux_p	hold	hold	If not busy then ram addrb takes addr_i		Yes
6	not_busy_web_mux_p	hold	hold	If not busy and the previous cycle there isn't wr_op_sel then web takes wr_en_i		Yes
7	not_busy_dinb_mux_p	Unknown	hold	if not busy and in the previous clcok tick there is wr_en_i and no wr_op_sel then dinb takes data_i		Yes
8	prime_selection_as_q_p	Unknown	hold	If mod_q_sel is set then the prime should be selected as group order		Yes
9	prime_selection_as_p_p	hold	hold	If mod_q_sel is not set then the prime should be selected as the p (prime)		Yes
10	req_digit_p	Unknown	hold	If req_digit is set and no wr_en then the secret_key should be shifted		Yes
11	no_req_digit_p	Unknown	hold	If req_digit isn't set then and no wr_En then the secret_key should hold the previous value		Yes
12	wr_en_op_sel_p	Unknown	hold	If wr_en is set and the wr_op_Sel is set then secret_key should take the value of data_i and reg_dinb_r should hold the previous value		Yes
13	wr_en_no_op_sel_p	Unknown	hold	If wr_en is set and the wr_op_Sel isn't set then secret_key shouldhold the previous value and reg_dinb_r should take the data_i		Yes
14	no_wr_en_i_p	hold	hold	If no wr_En then reg_dinb_r should hold the previous value		Yes
15	reg_addr_r_p	hold	hold	reg_addr_r always takes the addr_i		Yes
16	digit_in_p	hold	hold	digit_in is always equal to the MSB bit of the secretkey, which when shifted becomes secret_key[0]		Yes

fv_hmac_drbg_interface

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	reset property, when !reset_n or zeroize all the o/p are zero		Yes
2	idle_to_lfsr_p	unknown	holds	State transitioning from idle to lfsr if en and hmac_drbg_ready	only if counter cut	Yes
3	idle_wait_p	unknown	holds	If en and hmac_drbg isn't ready then stays back in idle state		Yes
4	lfsr_to_lambda_p	unknown	holds	State transition from lfsr to lambda when hmac_drbg_valid is set		Yes
5	lfsr_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
6	lambda_to_scalar_rnd_p	unknown	holds	State transition from lamda to scalar_rnd when hmac_drbg_valid is set	only if counter cut	Yes
7	lambda_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
8	scalar_rnd_to_rnd_done_p	unknown	holds	State transition from scalar_rnd to rnd_done when hmac_drbg_valid is set		Yes
9	scalar_rnd_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
10	rnd_done_to_masking_rnd_p	unknown	holds	if in rnd_done state then if keygen_sign is set then state changes to masking_rnd	only if counter cut	Yes
11	rnd_done_to_keygen_p	unknown	holds	if in rnd_done state then if keygen_sign isn't set then state changes to keygen	only if counter cut	Yes
12	masking_rnd_to_sign_p	unknown	holds	State transition from masking_rnd to sign when hmac_drbg_valid is set	only if counter cut	Yes
13	masking_rnd_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
14	keygen_to_done_p	unknown	holds	State transition from keygen to done when hmac_drbg_valid is set		Yes
15	keygen_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
16	sign_to_done_p	unknown	holds	State transition from sign to done when hmac_drbg_valid is set		Yes
17	sign_wait_p	unknown	holds	If hmac_drbg_valid isn't set then stays back in same state		Yes
18	done_to_idle_p	unknown	holds	State transition from done to idle		Yes
19	counter_reg_p	unknown	holds	counter_reg is checked if it adds 1 after everycycle		
20	counter_nonce_reg_p	unknown	holds	counter_nonce_reg has counter_nonce onece en is triggered.		
21	counter_nonce_reg_stable_p	unknown	holds	counter_nonce_reg stable if no en		
22	done_pulse_p	unknown	holds	done_edge is a pulse from the hmac_drbg_valid		
23	ready_liveliness_p	unknown	holds	eventually ready==1, once the fsm triggered		

fv_sha512_masked

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p		holds	Checks that all the resgiters are resetted and the state is idle, with the ready to high		
2	done_to_idle_p		holds	Checks the necessary registers, outputs holds the values when state transits from Done to idle		
3	idle_to_ctrl_rnd_p		holds	Checks if the state is in idle, if there is an init_cmd or next_cmd, state transists to CTRL_RND and checks if the state registers are unchanged and round counter remains zero		
4	ctrl_rnd_to_ctrl_rnd_p		holds	State transition remains CTRL_RND as long as the round_counter values is less than 9 and checks the necessary registers, masking register holds corrcet value		
5	ctrl_rnd_to_sha_rounds_224_p		holds	Checks if the state is in ctrl_rnd ,the mode choosen as 224,the init is triggered then the registers should be initialised with the respective constants of 224		
6	ctrl_rnd_to_sha_rounds_256_p		holds	hecks if the state is in ctrl_rnd ,the mode choosen as 256,the init is triggered then the registers should be initialised with the respective constants of 256		
7	ctrl_rnd_to_sha_round_512_p		holds	Checks if the state is in ctrl_rnd ,the mode choosen as 512,the init is triggered then the registers should be initialised with the respective constants of 512		
8	ctrl_rnd_to_sha_round_384_p		holds	Checks if the state is in ctrl_rnd ,the mode choosen is neither 512,256 nor 224,the init is triggered then the registers should be initialised with the respective constants of default, which covers 384 mode also		
9	ctrl_rnd_to_sha_rounds_next_p		holds	Checks if the state is in ctrl_rnd and there is no init signal and the next signal asserts then the register holds the past values		
10	sha_rounds_to_done_p		holds	Checks if the rounds are done then the registers are updated correctly		
11	sha_rounds_to_sha_rounds_before_16_p		holds	Checks if the the rounds less than 16 then the necessary registers are updated correctly and the round increments		
12	sha_rounds_to_sha_rounds_after_16_p		holds	Checks if the rounds are greater than 16 and less than 80 then the respective registers are updated correctly and the round increments		
13	idle_wait_p		holds	Checks if there isn't either init or next signal triggered in idle state then the state stays in idle and holds the past values and the core is ready		
					lfsr_inst.rnd signal is cut	

fv_scalar_blinding

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	hold	hold	when reset busy_o and data_o is zero		Yes
2	input_read_to_input_read_a	unknown	hold	If not busy and en is set then the computation is carried out	Constraints in place can be converted as assertions on top, convergence achieved on REGSIZE=24,RNDSIZE=12,RADIX=2 params, which are aligned in the same ratio as original params	Yes
3	input_read_wait_a	unknown	hold	If not busy and not enabled then it stays in not busy		Yes
4	input_read_next_a	unknown	hold	If not busy and enabled then busy is set		Yes

fv_dsa_ctrl

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top
1	reset_p	holds	holds	reset property, when reset all the o/p are zero		Yes
2	zeroize_p	holds	holds	when hwif_out.ECC_CTRL.ZEROIZE.value or debugUnlock_or_scan_mode_switch is set that is zeroize and the hw_clr of the respective outs should be triggered		Yes
3	no_zeroize_p	holds	holds	when hwif_out.ECC_CTRL.ZEROIZE.value or debugUnlock_or_scan_mode_switch is not set that is zeroize and the hw_clr of the respective outs should not be triggered		Yes
	no_zeroize_seed_clr_p	holds	holds	when no zeroize, then seed hwclr is equivalent to the reset step of the kv read data		
4	store_const_after_reset_p	holds	holds	After reset, the prog_cntr initially increases till 11 which are used for storing constants in the memory		Yes
5	store_const_end_p	holds	holds	Once the prog_cntr reaches 11 it is repeated for by cycle_count and ends up in the DSA_NOP		Yes
	stable_const_mem_p	unknown	holds	Once the constants are stored in the memory before the ecc_ready, they stay stable.		
6	counter_const_a_liveness_p	unknown	No Convergence	Using the symbolics as counter a and b, where counter a is triggered when prog_cntr is equal to symbolic counter a and counter b is triggered when prog_cntr is equal to symbolic counter b, once the cmd sequence starts then eventually counter_a should be triggered		Yes
7	counter_const_b_liveness_p	unknown	No Convergence	once the cmd sequence starts then eventually counter_b should be triggered		Yes
8	counter_integrity_const_p	unknown	holds	counter_b is only triggred if counter_a has already been triggered, this ensures the order is followed as the counter_b is greater than the counter_a		Yes
9	keygen_sequence_p	N/A	holds		Need to define a Macro REDUCED_PM_CTRL to use it with the pm_ctrl is abstract version, when all the sequences just have two steps like PM_INIT_S and PM_INIT_E in PM_INIT sequence	No
10	sequence_valid_p	unknown	No Convergence	checks that once any of the cmds are triggerd then eventually we will have the valid		Yes
11	valid_set_end_seq_p	unknown	holds	At the last step of the cmd sequence, the valid signal should be asserted		
12	no_valid_p	holds	holds	If the prog_cntr isn't in last step of the sequences and the DSA_NOP then valid should be deasserted		

13	counter_keygen_a_liveness_p	unknown	No Convergence	Using the symbolics as counter a and b, where counter a is triggered when prog_cntr is equal to symbolic counter a and counter b is triggered when prog_cntr is equal to symbolic counter b, once the cmd sequence starts then eventually counter_a should be triggered		Yes
14	counter_keygen_b_liveness_p	unknown	No Convergence	once the cmd sequence starts then eventually counter_b should be triggered		Yes
15	counter_keygen_integrity_p	unknown	No Convergence	counter_b is only triggred if counter_a has already been triggered, this ensures the order is followed as the counter_b is greater than the counter_a		
16	counter_sign_a_liveness_p	unknown	No Convergence	Using the symbolics as counter a and b, where counter a is triggered when prog_cntr is equal to symbolic counter a and counter b is triggered when prog_cntr is equal to symbolic counter b, once the cmd sequence starts then eventually counter_a should be triggered		Yes
17	counter_sign_b_liveness_p	unknown	No Convergence	once the cmd sequence starts then eventually counter_b should be triggered		Yes
18	counter_integrity_sign_p	unknown	No Convergence	counter_b is only triggred if counter_a has already been triggered, this ensures the order is followed as the counter_b is greater than the counter_a		Yes
19	counter_verify_a_liveness_p	unknown	No Convergence	Using the symbolics as counter a and b, where counter a is triggered when prog_cntr is equal to symbolic counter a and counter b is triggered when prog_cntr is equal to symbolic counter b, once the cmd sequence starts then eventually counter_a should be triggered		Yes
20	counter_verify_b_liveness_p	unknown	No Convergence	once the cmd sequence starts then eventually counter_b should be triggered		Yes
21	counter_integrity_verify_p	unknown	No Convergence	counter_b is only triggred if counter_a has already been triggered, this ensures the order is followed as the counter_b is greater than the counter_a		Yes
22	output_directconnection_input_p	holds	holds	Primary outputs connect to primary inputs		Yes
23	output_connectedto_submodules_p	holds	holds	Primary outputs connected to the submodule outputs, here it is read_reg		Yes
24	pcr_sign_mode_p	holds	holds	If pcr sign is set then msg we and privkey_in we should be ! zeroize and privkey_in should take data from pcr_Signing		Yes
25	no_pcr_sign_mode_p	holds	holds	If pcr sign isn't enabled then for msg no we and privkey_in is dependent on keyvault write_en		
26	privkey_out_we_p	unknown	No Convergence	If privkey_out i.e reading from the reg after the privkey computation and if the seed is not from the keyvault then privkey_out we is equal to !zeroize		Yes

27	no_privkey_out_we_p	holds	holds	If no privkey_out or keyvault is choosen as desitination then privkey_out we is deasserted		Yes
28	seed_we_p	holds	holds	if keyvault write is enabled and offset is equal to each word the seed we is equal to !zeroize		Yes
29	no_seed_we_p	holds	holds	if keyvault write is not enabled and offset is not equal to each word then seed we is deasserted		Yes
30	rd_core_we_p	unknown	No Convergence	Rest we are triggered with rd_core opcode and cycle_cnt=0		Yes
31	kv_privkey_read_ctrl_reg_p	holds	holds	keyvault privkey read ctrl reg is connected to primary input kv_rd_pkey_ctrl		Yes
32	kv_seed_read_ctrl_reg_p	holds	holds	keyvault seed read ctrl reg is connected to primary input kv_rd_seed_ctrl		Yes
33	kv_write_ctrl_reg_p	holds	holds	keyvault write ctrl reg is connected to primary input kv_wr_pkey_ctrl		Yes
34	kv_read_data_present_p	unknown	holds	kv_read data present stays asserted until privkey is generated if read_en of kv is set		
35	no_kv_read_data_present_p	unknown	holds	Once kv_read_data_present is set then it deasserts if privkey is ready to be read and no new read_en		
36	kv_reg_p	unknown	No Convergence	If privkey is ready to read and keyvault is choosen as destination then kv_reg will have the privkey		Yes
37	stable_kv_reg_p	unknown	holds	If privkey is ready to read and keyvault is choosen as destination then kv_reg will have the privkey		
38	primaryout_connected_to_kvout_p	holds	holds	Primary outputs directly connected to kv(submodule) outputs		Yes
39	ready_p	holds	holds	If pm_ctrl is not busy and prog_cntr is equal to DSA_NOP then ecc_ready to accept cmds		Yes
40	no_ready_p	holds	holds	If pm_ctrl is busy or prog_cntr is in cmd execution or memory write steps then ecc isn't ready		Yes
41	error_sign_p	unknown	holds	During sign subroutine if input privkey is zero or >= Group_order or while writing the outputs s and r are equal to zero error is triggered		Yes
42	error_keygen_p	unknown	holds	If keygen subroutine is triggered then input cannot have pcr_sign, this results in error		
43	error_verify_p	unknown	holds	During verifying subroutine if r and s inputs are equal to zero or greater than group order or the pubkey is greater than prime or If the cmd is just set as verify after reset then pcr_sign cannot be set along this results in error		Yes
44	notif_interrupt_p	holds	holds	Once valid signal is set then interrupt is triggered as a pulse		
45	subcomponent_busy_p	holds	holds	If pm_ctrl busy or sca_blinding modules are busy or if hmac is not ready then subcomponent_busy is asserted		Yes

46	no_subcomponent_busy_p	holds	holds	If none of the subcomponents are busy then subcomponent_busy is deasserted		Yes
47	hmac_init_p	holds	holds	If hmac_drbg is enabled from seq then if no error flag and cyc_cnt=3 hmac_init is triggered until the prog_cntr advances		Yes
48	no_hmac_init_less3cyc_p	holds	holds	If hmac_drbg en is set and the cntr is stable then hmac_init isn't set		Yes
49	no_hmac_init_p	holds	holds	If subcomponent busy or prog_cntr is DSA_NOP or there isn't hmac_drbg_en then no hmac_init		
50	hmac_mode_p	holds	holds	If sign subroutine is set the no subcomponent busy then hmac_mode is set		Yes
51	continue_hmac_mode_p	holds	holds	Once hmac_mode is stays stable until keygen subroutine triggers		Yes
52	no_hmac_mode_p	holds	holds	If keygen subroutine is set then no hmac_mode		Yes
53	continue_no_hmac_mode_p	holds	holds	Once hmac_mode deasserted stays stable until new sign subroutine is triggered		Yes
54	hmac_input_reg_p	holds	holds	Hmac inputs directly connected to the primary inputs		Yes
55	scalar_en_p	unknown	holds	If sca_en is enabled from seq then if no error flag and cyc_cnt=3 scalar_en is triggered until the prog_cntr advances		Yes
56	no_scalar_en_less3cyc_p	unknown	holds	If sca_en is set and the cntr is stable then scalar_en isn't set		
57	no_scalar_en_p	unknown	holds	If subcomponent busy or prog_cntr is DSA_NOP or there isn't sca_en then no scalar_en		Yes
58	scalar_in_reg_p	unknown	No Convergence	Input for scalar_blinding is from scalar_g_reg		Yes
59	stable_scalar_in_reg_p	unknown	holds			
60	pm_cmd_reg_p	unknown	holds	If no error and prog_cntr is not equal to DSA_NOP and none of the subcomponents are busy and cyc cnt ==3 then pm_ctrl cmd is equal to dsa_seq opcode pm_cmd		Yes
61	no_pm_cmd_reg_p	holds	holds	If error is triggered or prog_cntr is equal to DSA_NOP or subcomponents are busy then pm_cmd ==0		Yes
62	pm_cmd_when_cyc3less_p	unknown	holds	If cmd is acceptable but cyc cnt is less than 3 then pm_cmd stays stable		
63	wr_core_addr_eventually_in_memory_p	unknown	No Convergence	Eventually whatever address provided from the sequencer should end up in memory interface		Yes
64	write_reg_p	holds	holds	When write core command is enabled then the write_reg should be equal to the formal suite write_reg, which is an auxilary logic for write_reg		Yes
65	write_reg_no_cmd_p	holds	holds	When neither wr_core nor wr_scalar command then write_reg should be equal to zero		Yes

66	write_reg_sca_p	unknown	No Convergence	When write scalar command is enabled then the write_reg should be equal to the formal suite write_reg, which is an auxilary logic for write_reg		Yes
67	we_p	unknown	No Convergence	When rd_core and cyclecnt=0 then the write enables of the respective outputs should be triggered		Yes
68	no_rd_we_p	holds	holds	When no rd core or the cyclecnt!=0 then all the write enable flags for registsr should be set to zero		Yes
69	read_reg_r_p	unknown	N/A	when reading r value it shouldn't have illegal values as zero and group order		
70	read_reg_s_p	unknown	N/A	when reading s value it shouldn't have illegal values as zero and group order		
71	read_reg_ver_r_p	unknown	N/A	when reading verify_r value it shouldn't have illegal values as zero and group order		
72	hmac_drbg_in_scalar_G_reg_p	unknown	No Convergence	Scalar_G_reg an internal register when not in verification sequence then should have hmac_drbg_result nd less than group order		Yes
73	read_reg_scalar_G_reg_p	unknown	No Convergence	when in verifying sequence and if scalar_G write enable flag is set then it should have the read_reg value and should be in the range of zero and group order		Yes
74	no_read_reg_scalar_G_reg_p	unknown	No Convergence	When in verifying sequence and the flag isn't set for write enable then it should hold the value		Yes
75	we_scalar_pk_reg_p	unknown	No Convergence	When scalar_PK write enable flag is set then it should take the value from read_Reg and should in the range of zero and group order		Yes
76	no_we_scalar_pk_reg_p	holds	holds	If write enable flag isn't set then it shall hold the value stable		Yes
77	pk_chk_reg_p	unknown	No Convergence	When pk_chk write enable flag is set then it should take the value from read_reg		Yes
78	no_pk_chk_reg_p	holds	holds	If write enable flag isn't set then it shall hold the value stable		Yes

fv_hmac_drbg

Sl No.	Property	Proof status on Top	Proof status block level	Description	Additional notes	Can add to top	
1	reset_p		holds	Checks that all the resgiters are resetted and the state is idle, with the ready to high			
2	done_to_idle_p		holds	Checks if tag register outputs correct value and valid is high, when transition from done to idle states	Fails on Top, until unless nonce is stable from the top		
3	k10_to_k11_a		holds	Checks if the state is in K11, the hmac_core is called with hmac_next_cmd is set to high and remaining nonce as the block_msg and key remains the same			
4	K11_to_v1_p		holds	Checks if the state is in V1, the hmac_core is called with hmac_init_cmd is set to high and block_msg as the previous block_register and key as updated tag from previous hmac computation			
5	v1_to_k20_p		holds	Checks if the state is in K20, the hmac_core is initalized with hmac_init_cmd set, hmac_key remains the same and block_msg as the newly computed hmac_tag appened with count_register incremented and inputs entropy and nonce			
6	k20_to_k21_p		holds	Checks if the state is in K21, the hmac_core is called with hmac_next_cmd is set to high and remaining nonce as the block_msg and key remains the same			
7	k21_to_v2_p		holds	Checks if the state is in V2, the hmac_core is called with hmac_init_cmd is set to high and block_msg as the previous block_register and key as updated tag from previous hmac computation			
8	v2_to_t_p		holds	Checks if the state is in T, reads the computed tag from the hmac_core and appends it with previous stored tag_register in order to check if tag condition is holding or not			
9	t_to_k3_p		holds	Checks if the state is in K3, where the computated tag from hmac_core failes the required tag condition and goes another round of randomness, by initalizing hmac_core and setting hmac_key as same and block_msg as previous tag appended with zeroes			
10	t_to_done_p		holds	Checks if the is in done state, where the computated tag from hmac_core passes the required tag condition			
11	k3_to_v3_p		holds	if in rnd_done state then if keygen_sign isn't set then state changes to keygen			
12	v3_to_t_p		holds	Checks if the state is in T, reads the computed tag from the hmac_core and appends it with previous stored tag_register in order to check if tag condition is holding or not			
13	idle_to_k10_p		holds	Checks if the state reached is K10, checks for initalization of drbg and HMAC_CORE is ready. Count_register is set to zero, hmac_key is set with init key and hmac_block is appened with init block_register and inputs nonce and entropy			
14	idle_to_k10_1_p		holds	Checks if the state reached is K10, checks for next_cmd and HMAC_CORE is ready. Count_register is incremented, hmac_key is set with previously computed and hmac_block is appened with previously computed hmac_tag and inputs nonce and entropy			
15	k10_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state k10, it remains in same state and certain registers holds past values and ready is low.			
16	k20_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state k20, it remains in same state and certain registers holds past values and ready is low.			
17	v1_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state v1, it remains in same state and certain registers holds past values and ready is low.			
18	k11_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state k11, it remains in same state and certain registers holds past values and ready is low.			
19	k21_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state k21, it remains in same state and certain registers holds past values and ready is low.			
20	v2_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state v2, it remains in same state and certain registers holds past values and ready is low.			
21	t_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state t, it remains in same state and certain registers holds past values and ready is low.			
22	k3_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state k3, it remains in same state and certain registers holds past values and ready is low.			
23	v3_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state v3, it remains in same state and certain registers holds past values and ready is low.			
24	idle_wait_p		holds	Checks if there is valid hmac_tag from hmac_core while in state idle, it remains in same state and certain registers holds past values and while ready is high and previous computed drbg is valid.			