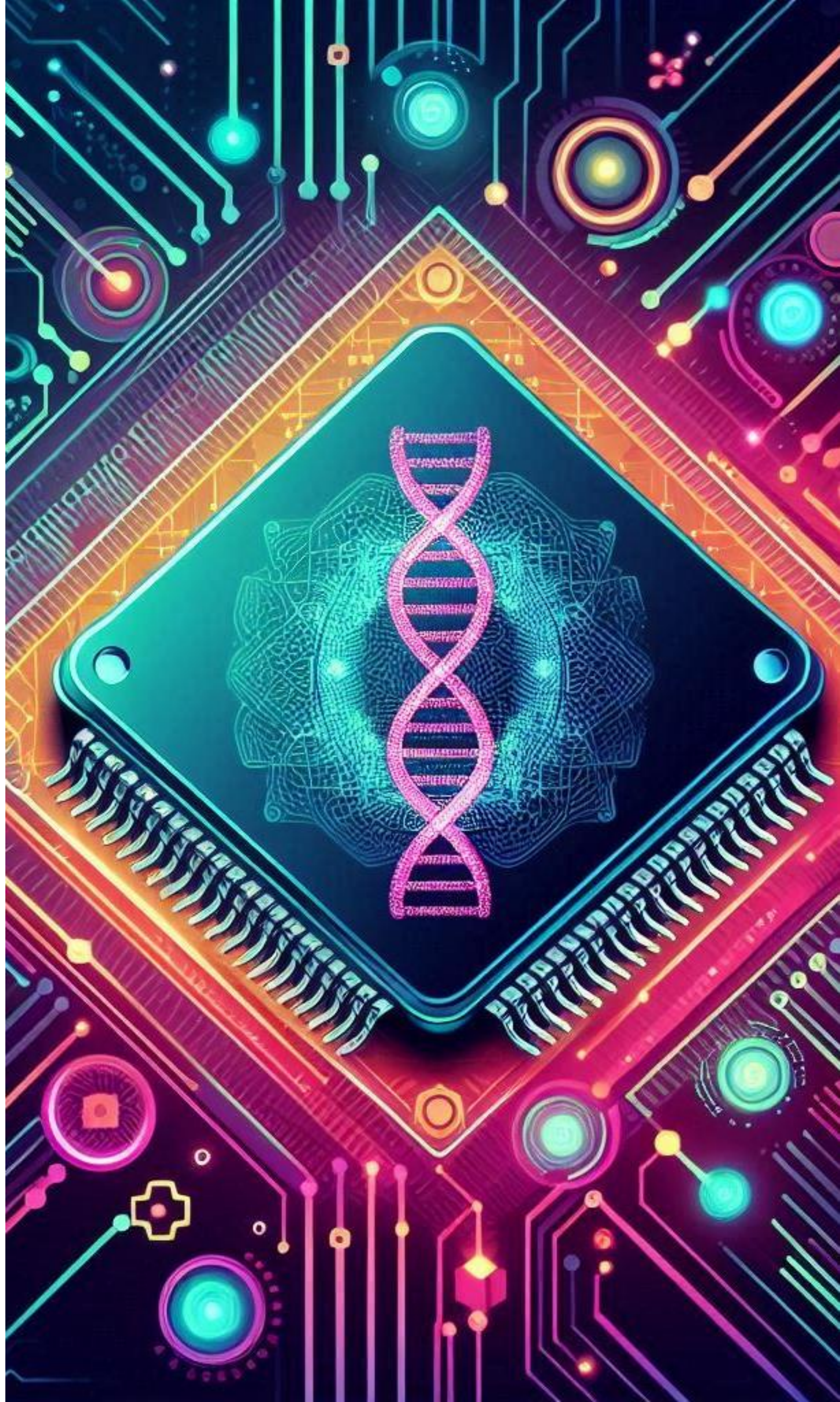
UVM Verification of an Accelerator for Genetic Sequence Alignment

- Students: Dor Lerman, Liran Lavi
- Supervisor: Pavel Gushpan
- Summer 2024
- VLSI lab, Technion

- Design was taken from a previous Project



Agenda

Goals

Functional Verification: Ensure the Smith-Waterman accelerator operates accurately according to algorithm specifications for reliable sequence. alignment

UVM-Based Debugging: Develop a UVM environment to facilitate efficient debugging of the accelerator's Verilog code, enhancing performance and stability.

Flexible UVM Environment: Establish a versatile UVM setup adaptable to various testing scenarios for comprehensive validation.

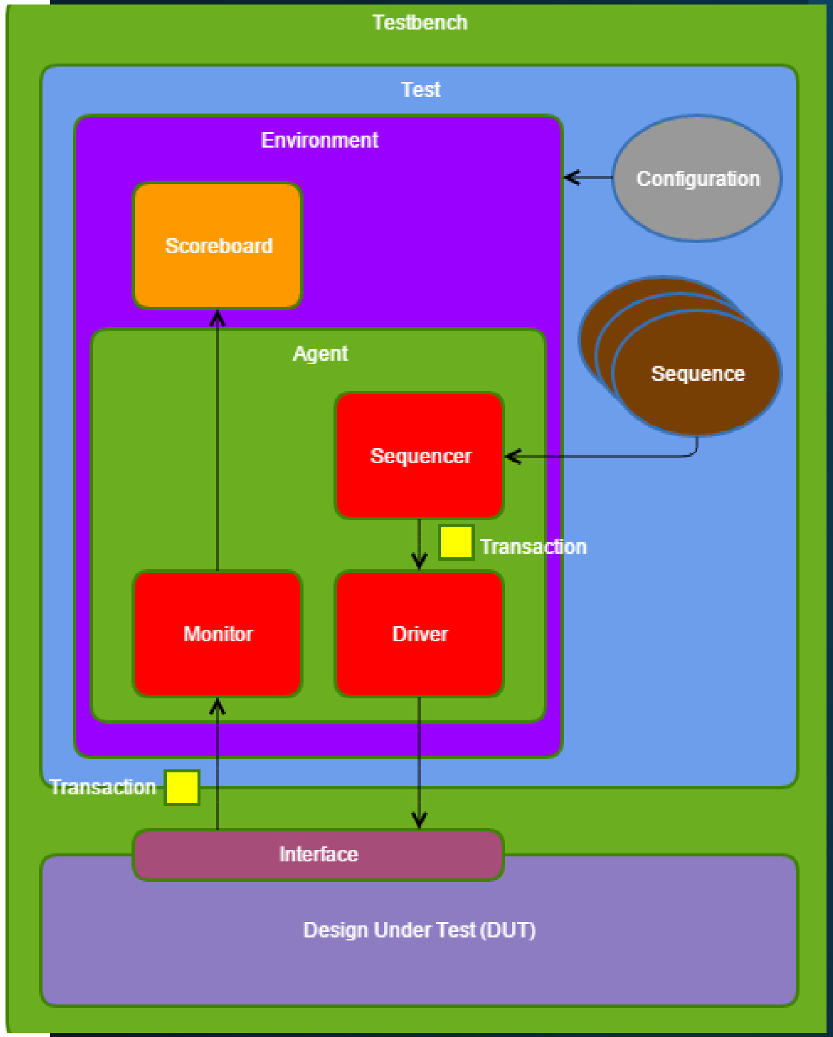
Comprehensive Testing: Aim for high functional and code coverage to identify any design flaws in the accelerator.

What is UVM

UVM (Universal Verification Methodology) is an industry-standard framework for verifying integrated circuit designs.

Using UVM, you can:

- **Generate Randomized Stimuli:** Use UVM to create a variety of test scenarios through constrained randomization.
- **Build Reusable and Modular Testbenches:** Use UVM's OOP structure to design test environments that are easy to reuse and maintain.
- **Perform Functional Coverage Analysis:** Make sure your design is fully tested by checking which features or scenarios have been covered.
- **Debug Efficiently:** Take advantage of UVM's built-in tools to find and fix issues faster.
- **Support Verification at Multiple Levels:** Use UVM to test designs at both small (block) and large (system) scales, making it flexible and scalable.



Sequence Alignment

- Method of arranging sequences of DNA, RNA or proteins
- Aims to identify regions of similarity that may indicate on a relationship between the sequences
- Can be used also for non-biological sequences

Sequence Alignment

TACGCTTG

CTACCTAG

⬇

TAC – CT

TACGCT

Smith Waterman Algorithm

Dynamic Programming algorithm used for local sequence alignment

Identify regions of similarity between the Query sequence and the Database sequence

Method:

1. **Initialization:** build a scoring matrix initialize its first row and column with zeros
2. **Scoring:** Calculate score for each cell based on scoring scheme
3. **Traceback:** Determine the optimal alignment by trace back from the highest-scoring cell in matrix to a cell with a score of zero

Time complexity:

$O(n^2)$

Space complexity:

$O(n^2)$

Initialization

$$H_{i,j} = max \begin{cases} H_{i-1,j-1} + MATCH/MISMTACH \\ H_{i-1,j} + GAP\_PENALTY \\ H_{i,j-1} + GAP\_PENALTY \\ 0 \end{cases}$$

Match	+1
Mismatch	–1
Gap Penalty	–2

S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	0								
T	0								
A	0								
C	0								
C	0								
T	0								
A	0								
G	0								

Scoring

$$H_{i,j} = max \begin{cases} H_{i-1,j-1} + MATCH/MISMTACH \\ H_{i-1,j} + GAP\_PENALTY \\ H_{i,j-1} + GAP\_PENALTY \\ 0 \end{cases}$$

Match	+1
Mismatch	–1
Gap Penalty	–2

S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	0	0							
T	0								
A	0								
C	0								
C	0								
T	0								
A	0								
G	0								

Scoring

$$H_{i,j} = max \begin{cases} H_{i-1,j-1} + MATCH/MISMTACH \\ H_{i-1,j} + GAP\_PENALTY \\ H_{i,j-1} + GAP\_PENALTY \\ 0 \end{cases}$$

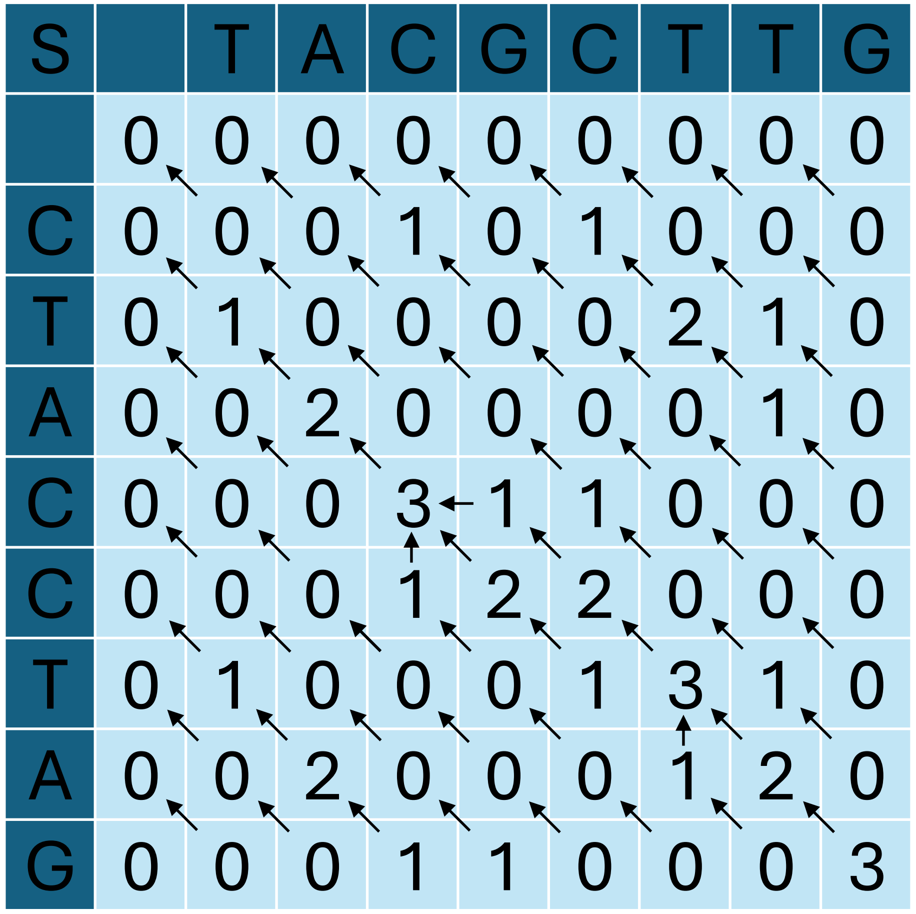
Match	+1
Mismatch	–1
Gap Penalty	–2

S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	0	↖ 0	↖ 0						
T	0	↖ 1							
A	0								
C	0								
C	0								
T	0								
A	0								
G	0								

Scoring

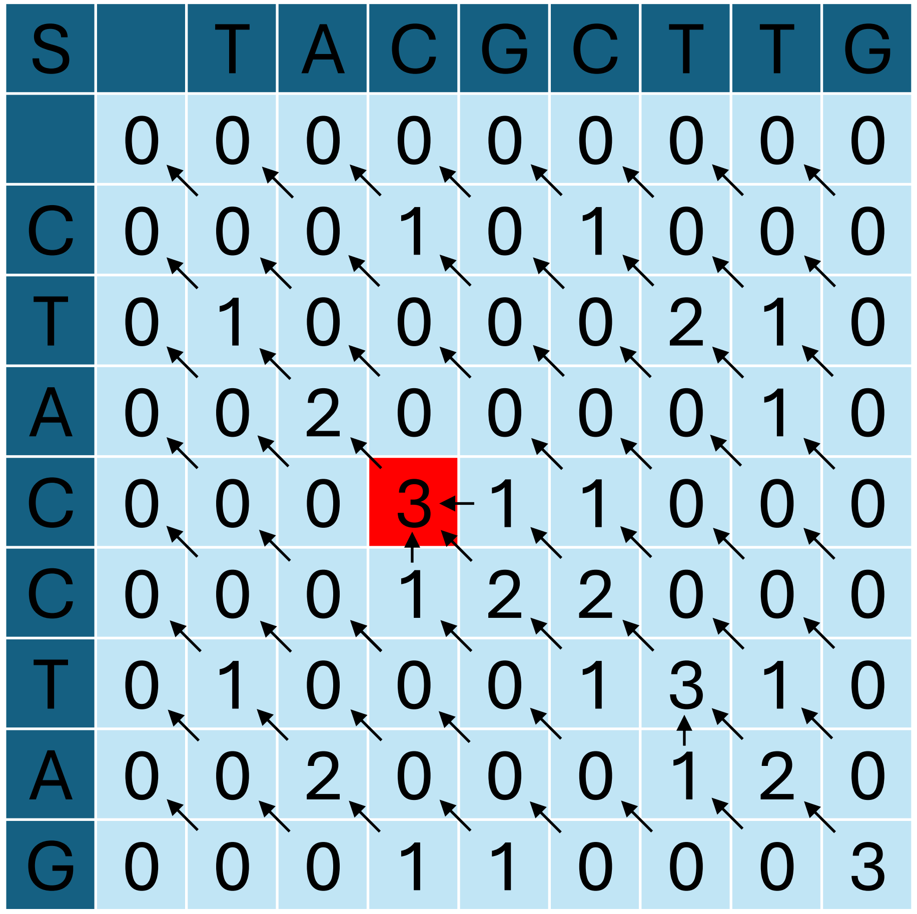
$$H_{i,j} = max \begin{cases} H_{i-1,j-1} + MATCH/MISMTACH \\ H_{i-1,j} + GAP\_PENALTY \\ H_{i,j-1} + GAP\_PENALTY \\ 0 \end{cases}$$

Match	+1
Mismatch	–1
Gap Penalty	–2



S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	1	0	0	0
T	0	1	0	0	0	0	2	1	0
A	0	0	2	0	0	0	0	1	0
C	0	0	0	3	←1	1	0	0	0
C	0	0	0	↑1	2	2	0	0	0
T	0	1	0	0	0	1	3	1	0
A	0	0	2	0	0	0	↑1	2	0
G	0	0	0	1	1	0	0	0	3

Traceback



S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	1	0	0	0
T	0	1	0	0	0	0	2	1	0
A	0	0	2	0	0	0	0	1	0
C	0	0	0	3	1	1	0	0	0
C	0	0	0	1	2	2	0	0	0
T	0	1	0	0	0	1	3	1	0
A	0	0	2	0	0	0	1	2	0
G	0	0	0	1	1	0	0	0	3

Match	+1
Mismatch	–1
Gap Penalty	–2

Traceback

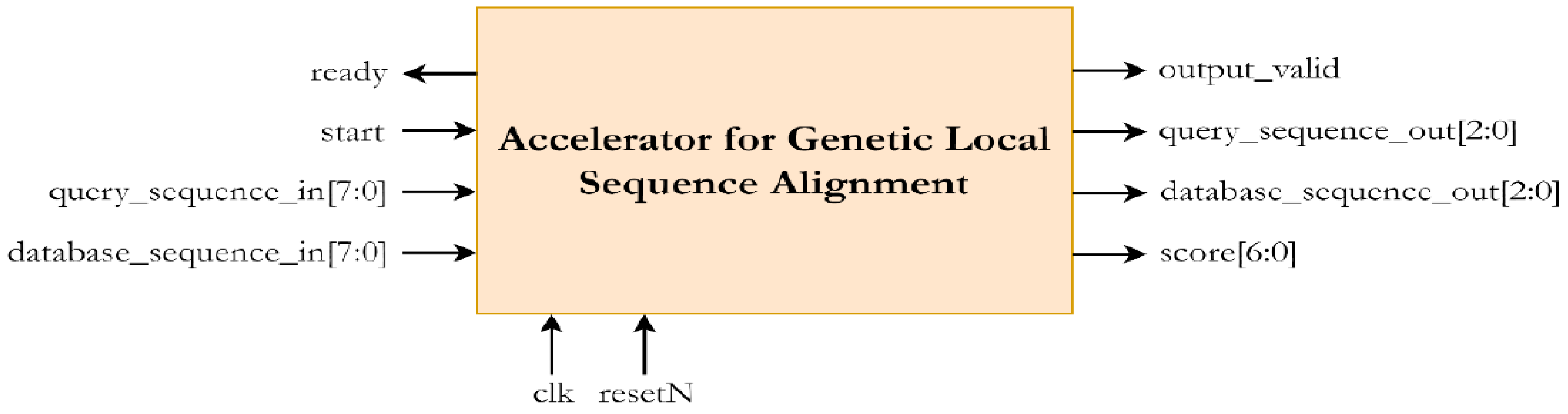
TAC
TAC

S		T	A	C	G	C	T	T	G
	0	0	0	0	0	0	0	0	0
C	**0**	0	0	1	0	1	0	0	0
T	0	**1**	0	0	0	0	2	1	0
A	0	0	**2**	0	0	0	0	1	0
C	0	0	0	**3**	1	1	0	0	0
C	0	0	0	1	2	2	0	0	0
T	0	1	0	0	0	1	3	1	0
A	0	0	2	0	0	0	1	2	0
G	0	0	0	1	1	0	0	0	3

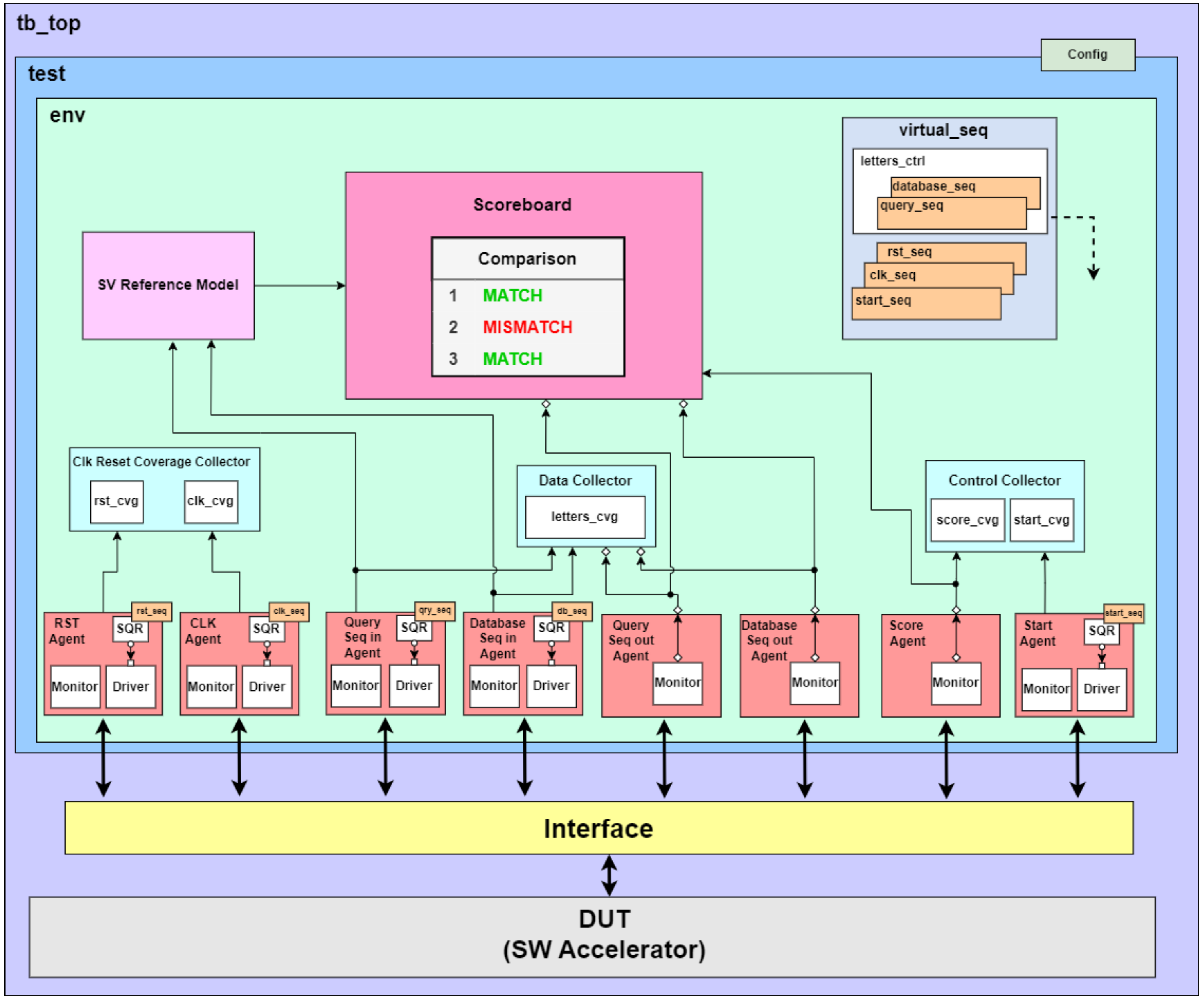
Match	+1
Mismatch	–1
Gap Penalty	–2

Sequence Alignment DUT

An accelerator that implements the Smith Waterman Algorithm


ready
start
query_sequence_in[7:0]
database_sequence_in[7:0]
Accelerator for Genetic Local Sequence Alignment
output_valid
query_sequence_out[2:0]
database_sequence_out[2:0]
score[6:0]
clk
resetN

TB Architecture



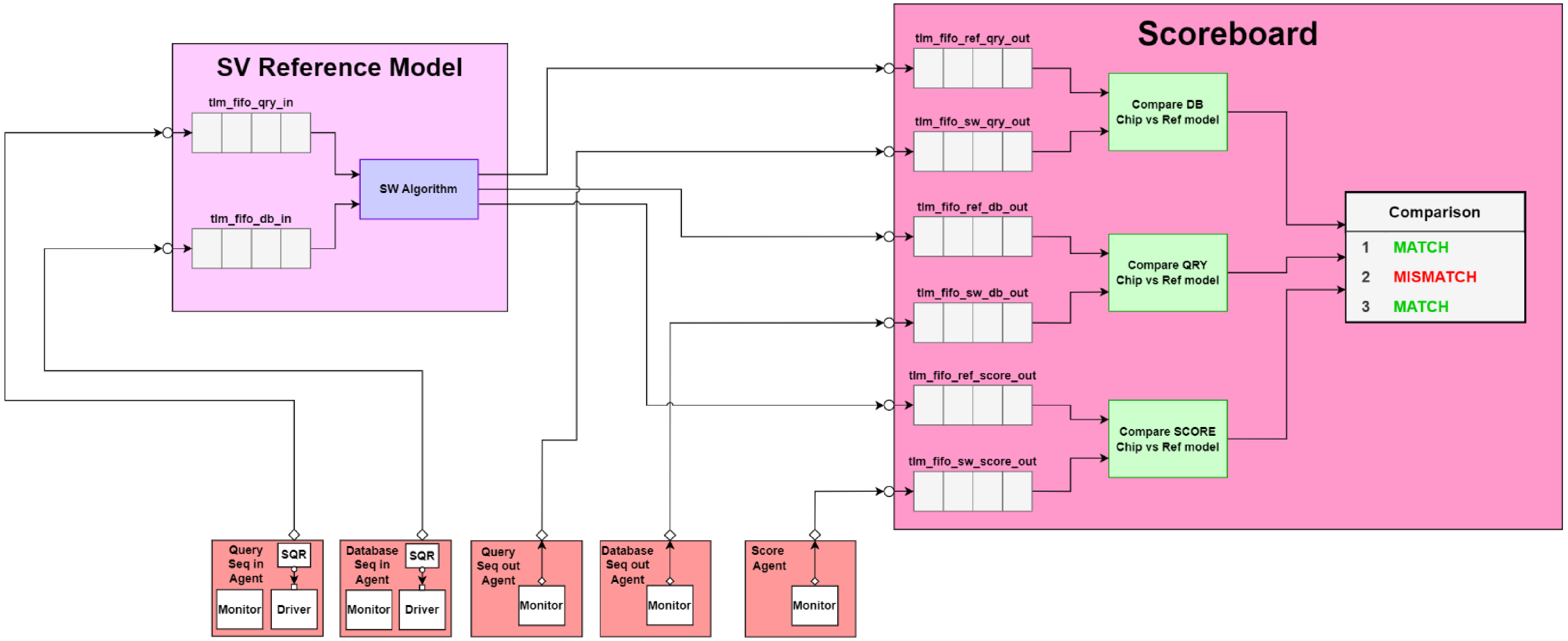
Reference Model

The reference model implements the **Smith Waterman Algorithm,** which is used for local sequence alignment.

The input to the model is **2 sequences , 32 letters** each.

The output of the model is as the output of the chip – the **aligned sequences** and the **maximum score achieved.**

Reference model and Scoreboard Interaction



Test Plan

Test	Description
base_test	A base test class that other tests inherit from. It generates a clock with a 50% duty cycle and provides a single reset signal with a fixed latency. The start signal is triggered after a fixed delay, and packets are inserted randomly.
Clk_test	Inherits from base_test and includes the following configurations: duty cycle, random stopping, and varying periods. This test performs the same behavior as base_test with randomly inserted sequence pairs, while also providing control over the clock signal (clk).
Rst_test	Inherits from base_test and allows configuration of the reset signal (rst) duration. This test performs the same behavior as base_test with randomly inserted sequence pairs, while also providing control over the reset duration.
Clk_rst_test	Inherits from base_test and combines the configuration of both the clock (clk) and reset (rst). It behaves like base_test in terms of data and control signals.
start_test	Inherits from base_test and allows configuration of the start signal's duration and behavior. This test behaves the same as base_test with respect to the clock (clk), reset (rst), and data, but the start signal is controlled by this test.
Letters_test	Inherits from base_test and includes a configuration for controlling the similarity of sequences using a mask. This test has full control over the inserted data and the sequence similarity.

Generate Sequences

Generate Random 4 letters for the Query

EX: A T G T

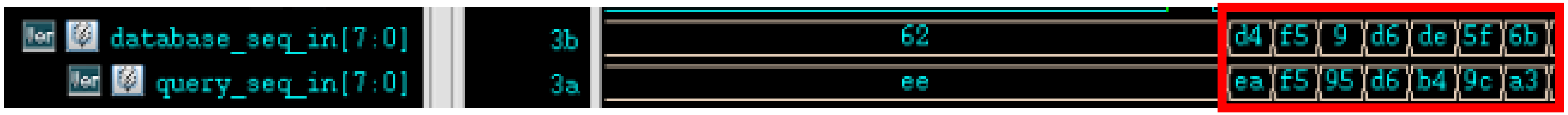
Generate a Mask using probability variable

1	0	1	1

Make the Database using the generated Query and the Mask

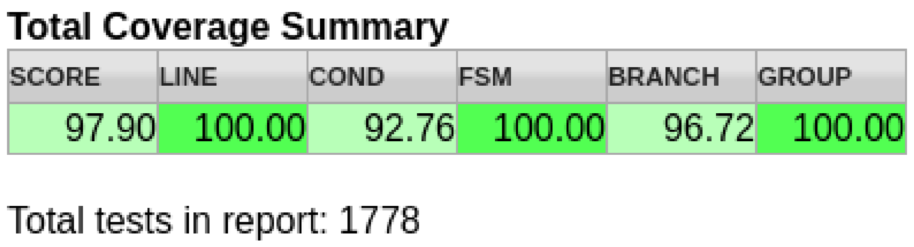
EX: A A G T

Letters_test



Generated sequences based on mask

Coverage Results

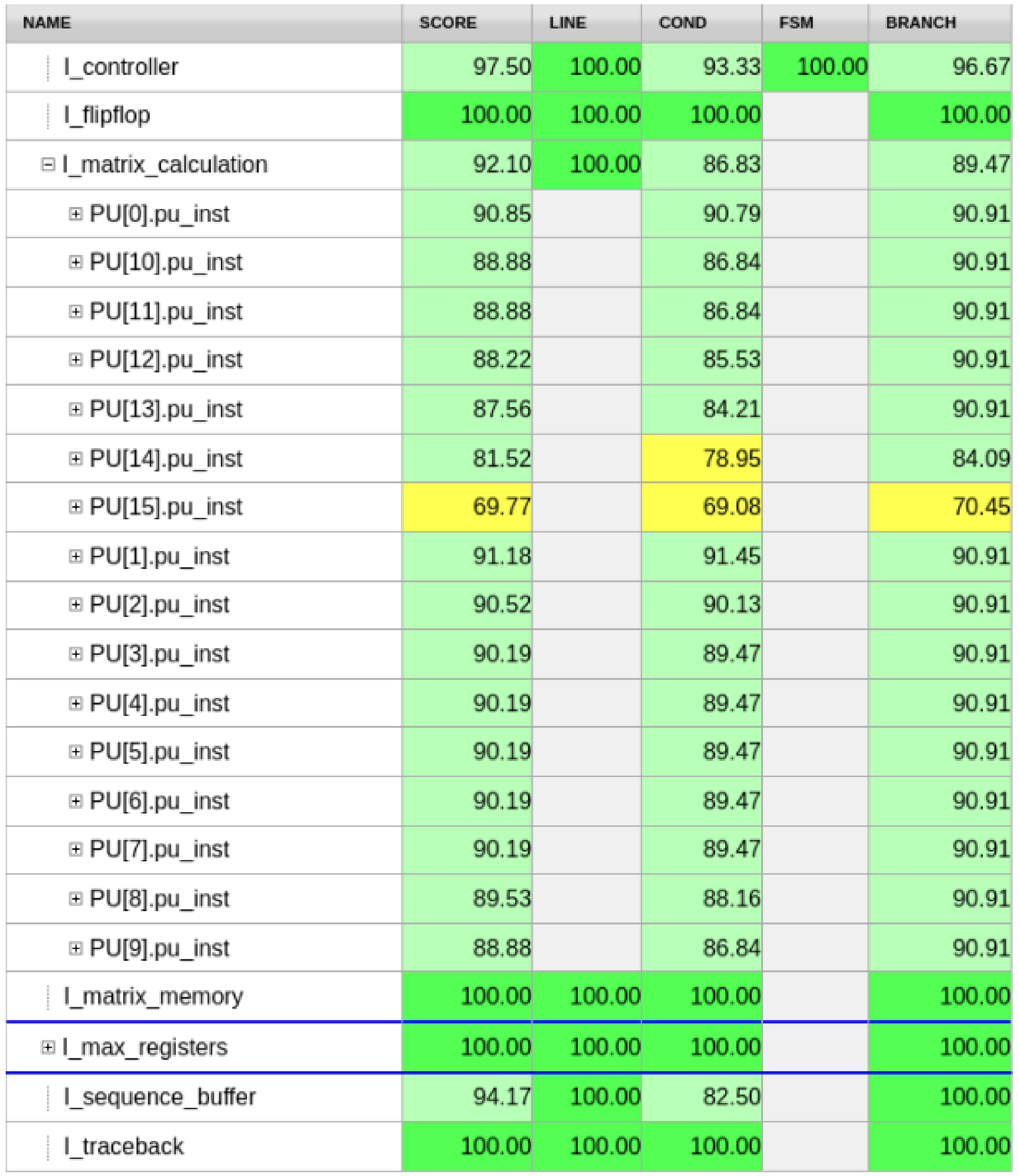


Total Coverage Summary

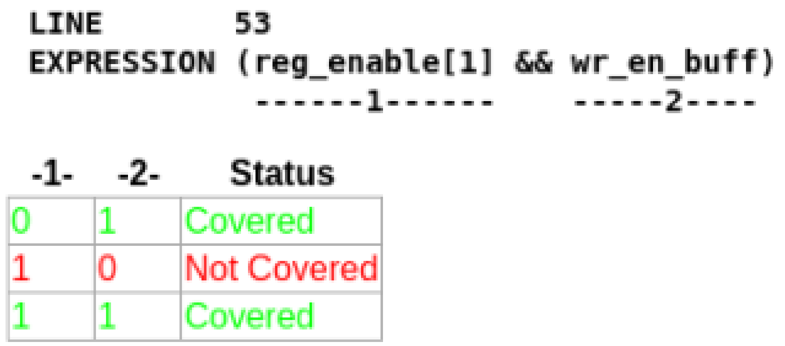
SCORE	LINE	COND	FSM	BRANCH	GROUP
97.90	100.00	92.76	100.00	96.72	100.00

Total tests in report: 1778

Code Coverage



NAME	SCORE	LINE	COND	FSM	BRANCH
I_controller	97.50	100.00	93.33	100.00	96.67
I_flipflop	100.00	100.00	100.00		100.00
⊟ I_matrix_calculation	92.10	100.00	86.83		89.47
⊞ PU[0].pu_inst	90.85		90.79		90.91
⊞ PU[10].pu_inst	88.88		86.84		90.91
⊞ PU[11].pu_inst	88.88		86.84		90.91
⊞ PU[12].pu_inst	88.22		85.53		90.91
⊞ PU[13].pu_inst	87.56		84.21		90.91
⊞ PU[14].pu_inst	81.52		78.95		84.09
⊞ PU[15].pu_inst	69.77		69.08		70.45
⊞ PU[1].pu_inst	91.18		91.45		90.91
⊞ PU[2].pu_inst	90.52		90.13		90.91
⊞ PU[3].pu_inst	90.19		89.47		90.91
⊞ PU[4].pu_inst	90.19		89.47		90.91
⊞ PU[5].pu_inst	90.19		89.47		90.91
⊞ PU[6].pu_inst	90.19		89.47		90.91
⊞ PU[7].pu_inst	90.19		89.47		90.91
⊞ PU[8].pu_inst	89.53		88.16		90.91
⊞ PU[9].pu_inst	88.88		86.84		90.91
I_matrix_memory	100.00	100.00	100.00		100.00
⊞ I_max_registers	100.00	100.00	100.00		100.00
I_sequence_buffer	94.17	100.00	82.50		100.00
I_traceback	100.00	100.00	100.00		100.00



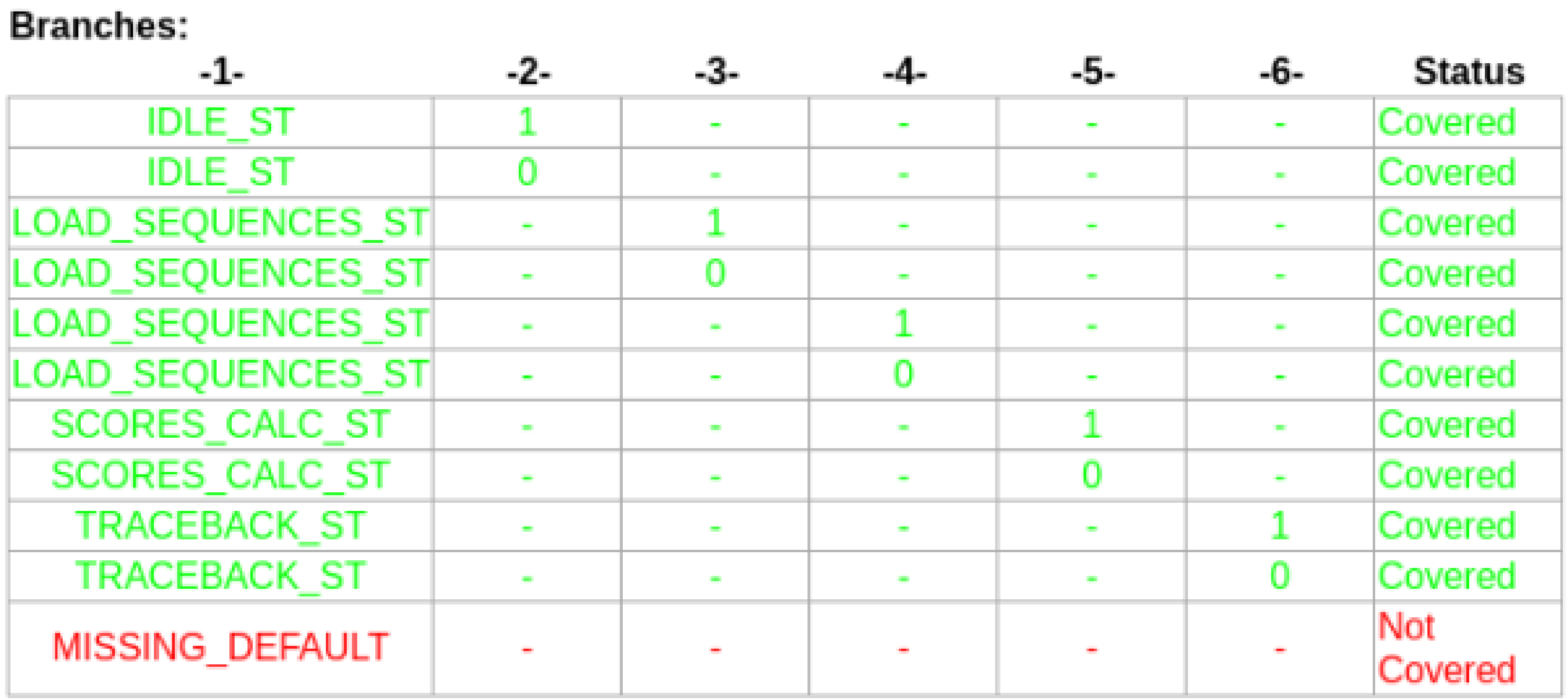
```
LINE       53
EXPRESSION (reg_enable[1] && wr_en_buff)
            ------1------    -----2----
```

-1-	-2-	Status
0	1	Covered
1	0	Not Covered
1	1	Covered

Sequence Buffer

- The **'we_en_buff'** is responsible for setting the **'reg_enable[i]'** to 1 for the matching bit. Thefore, neither of the bits in **'reg_enable'** will set to 1 when **'we_en_buff'** is not active.

Controller



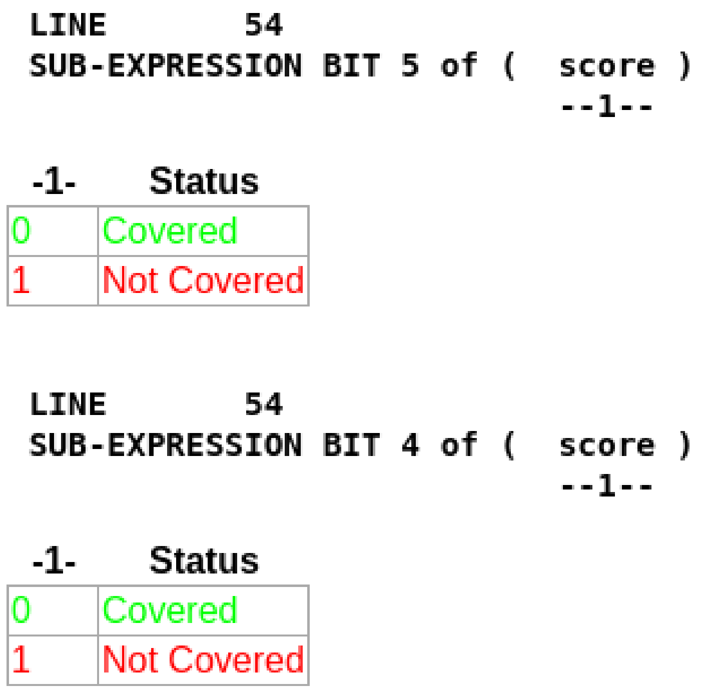
Branches:

-1-	-2-	-3-	-4-	-5-	-6-	Status
IDLE_ST	1	-	-	-	-	Covered
IDLE_ST	0	-	-	-	-	Covered
LOAD_SEQUENCES_ST	-	1	-	-	-	Covered
LOAD_SEQUENCES_ST	-	0	-	-	-	Covered
LOAD_SEQUENCES_ST	-	-	1	-	-	Covered
LOAD_SEQUENCES_ST	-	-	0	-	-	Covered
SCORES_CALC_ST	-	-	-	1	-	Covered
SCORES_CALC_ST	-	-	-	0	-	Covered
TRACEBACK_ST	-	-	-	-	1	Covered
TRACEBACK_ST	-	-	-	-	0	Covered
MISSING_DEFAULT	-	-	-	-	-	Not Covered

- The controller is missing the default value explicitly in the FSM case. However, it is set to be the previous state, therefore it is not considered a problem.

Matrix Calculation

- In all PUs, the parameters and algorithm limit the score to a certain maximum value.



```
LINE        54
SUB-EXPRESSION BIT 5 of (   score )
                           --1--
```

-1-	Status
0	Covered
1	Not Covered

```
LINE        54
SUB-EXPRESSION BIT 4 of (   score )
                           --1--
```

-1-	Status
0	Covered
1	Not Covered

PU number i when $1 \leq i \leq 15$

- Let's for example check PU number 1. This PU is responsible for the values across the first column. Those cannot provide values in the range of 16 to 25.

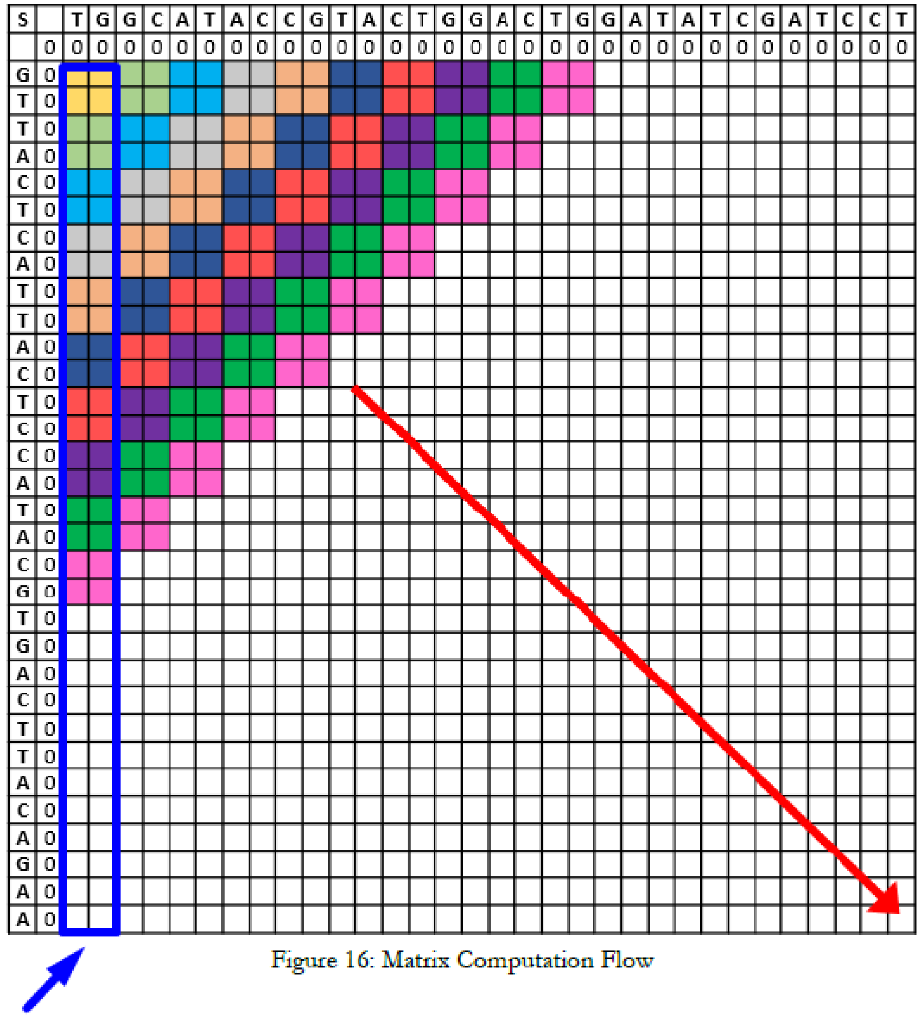
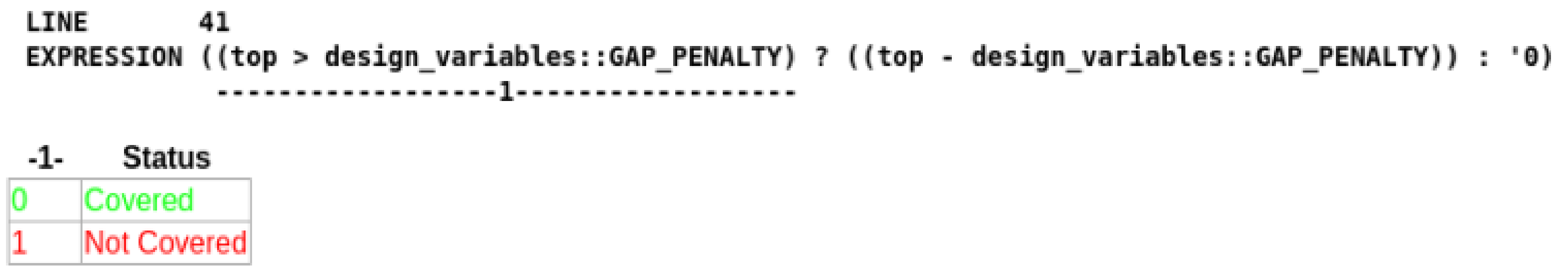

S
T G G C A T A C C G T A C T G G A C T G G A T A T C G A T C C T
G T T A C T C A T T A C T C C A T A C G T G A C T T A C A G A A
0

Figure 16: Matrix Computation Flow

Matrix calculation - PU number 15



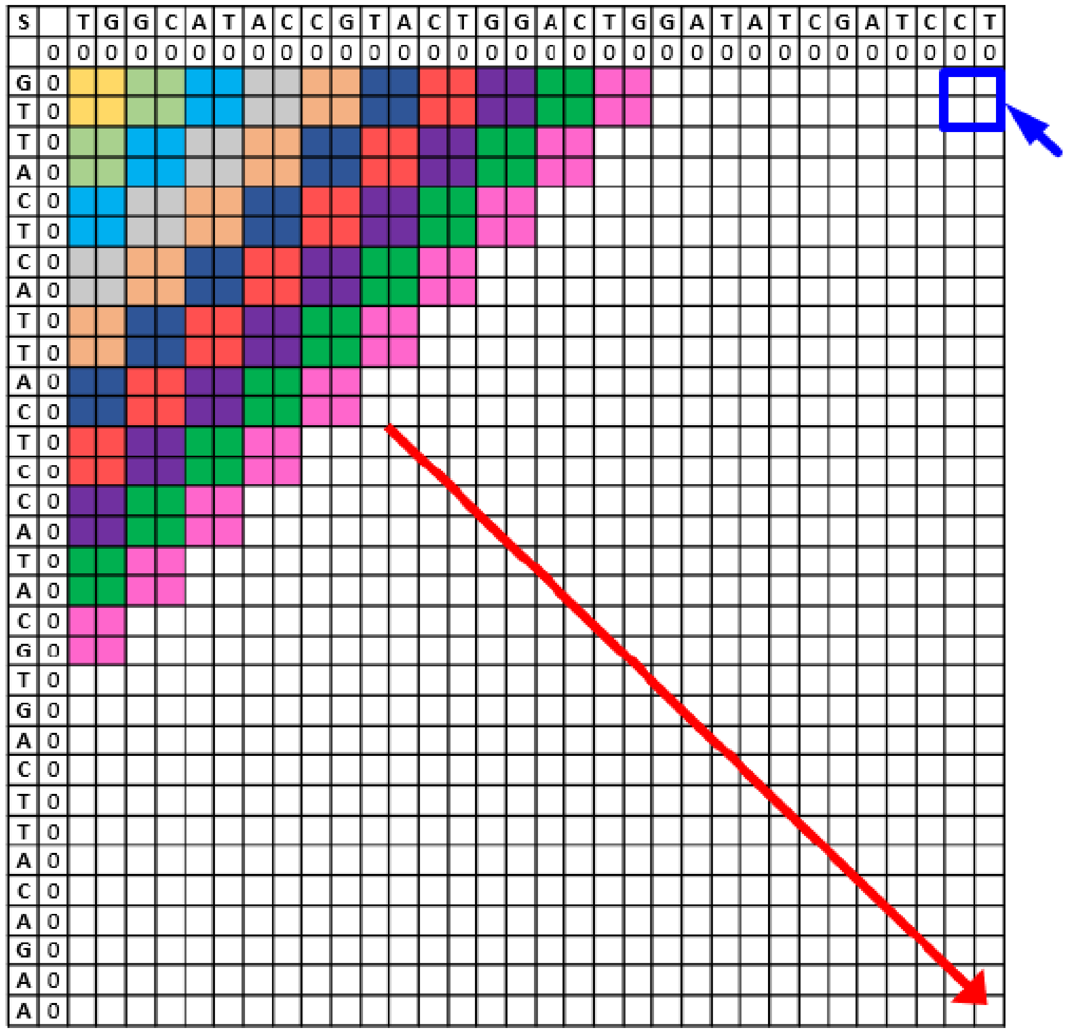
```
LINE       41
EXPRESSION ((top > design_variables::GAP_PENALTY) ? ((top - design_variables::GAP_PENALTY)) : '0)
            ------------------1-------------------
```

-1-	Status
0	Covered
1	Not Covered

- PU number 15 , never fulfil this or identical conditions in the code

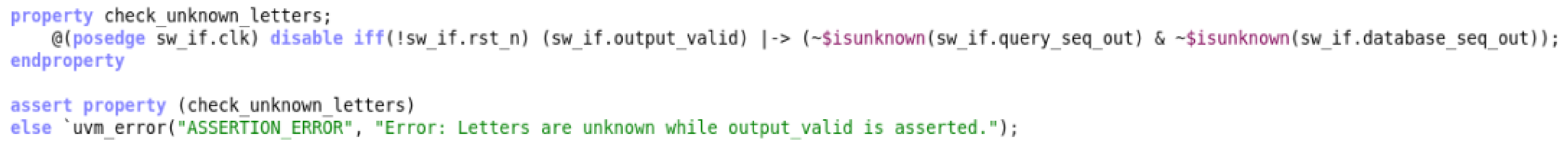
PU number 15

- by looking at PU number 15, and his area of responsibility, we can tell that the result coming from the top and diagonal cannot satisfy the condition based on the algorithm that updates the matrix where there is a mismatch.


S
T G G C A T A C C G T A C T G G A C T G G A T A T C G A T C C T
G T T A C T C A T T A C T C C A T A C G T G A C T T A C A G A A
0
Figure 16: Matrix Computation Flow

SVA – SystemVerilog Assertions

- SystemVerilog assertions are essential for chip design verification, helping engineers define expected behaviors and detect issues early.
- They monitor signal consistency and state transitions, ensuring the design functions as intended. By embedding assertions in designs or testbenches, engineers can automatically capture violations, improving design quality and reducing verification time.



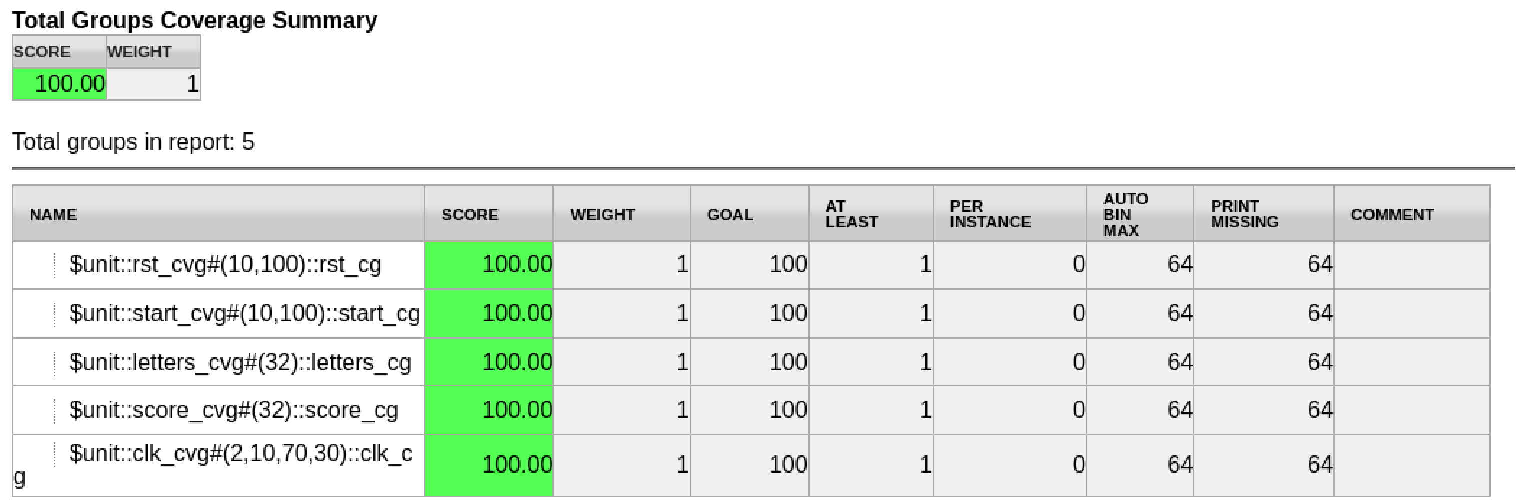
```
property check_unknown_letters;
    @(posedge sw_if.clk) disable iff(!sw_if.rst_n) (sw_if.output_valid) |-> (~$isunknown(sw_if.query_seq_out) & ~$isunknown(sw_if.database_seq_out));
endproperty

assert property (check_unknown_letters)
else `uvm_error("ASSERTION_ERROR", "Error: Letters are unknown while output_valid is asserted.");
```

Coverage plan

clk coverage	Rst coverage	Score coverage	Start coverage	Letters coverage
• Duty cycle : 30% – 70% • Clock period : 2ns – 10ns • Clk on/off – 0 or 1	• Rst duration: 10ns – 100 ns	• Score values: 0 – 32	• Start duration: 10 ns – 100ns	• Identical number of letters: 0 – 32. where 0 means different sequences. 32 means identical sequence.

Functional Coverage



Total Groups Coverage Summary

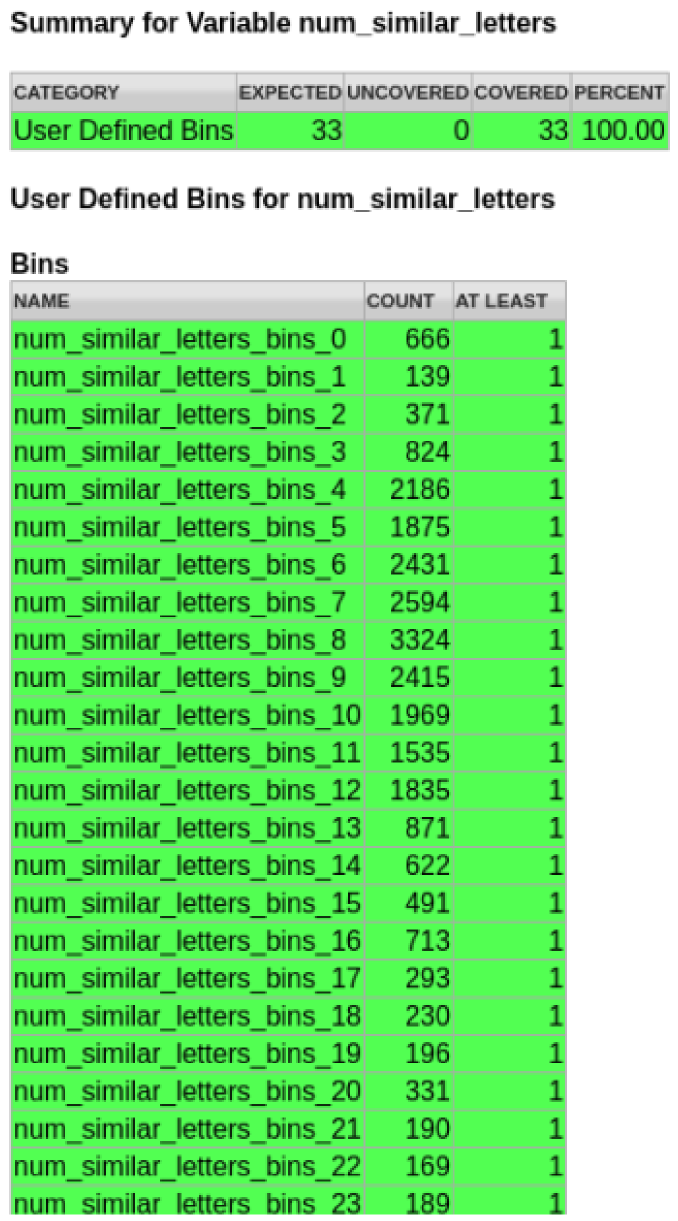
SCORE	WEIGHT
100.00	1

Total groups in report: 5

NAME	SCORE	WEIGHT	GOAL	AT LEAST	PER INSTANCE	AUTO BIN MAX	PRINT MISSING	COMMENT
$unit::rst_cvg#(10,100)::rst_cg	100.00	1	100	1	0	64	64	
$unit::start_cvg#(10,100)::start_cg	100.00	1	100	1	0	64	64	
$unit::letters_cvg#(32)::letters_cg	100.00	1	100	1	0	64	64	
$unit::score_cvg#(32)::score_cg	100.00	1	100	1	0	64	64	
$unit::clk_cvg#(2,10,70,30)::clk_cg	100.00	1	100	1	0	64	64	

- This result confirms that all groups in our coverage plan were addressed.
- We gave equal weight to each group by assigning the same weight – all of them are as important as others.

Letters Coverage



Summary for Variable num_similar_letters

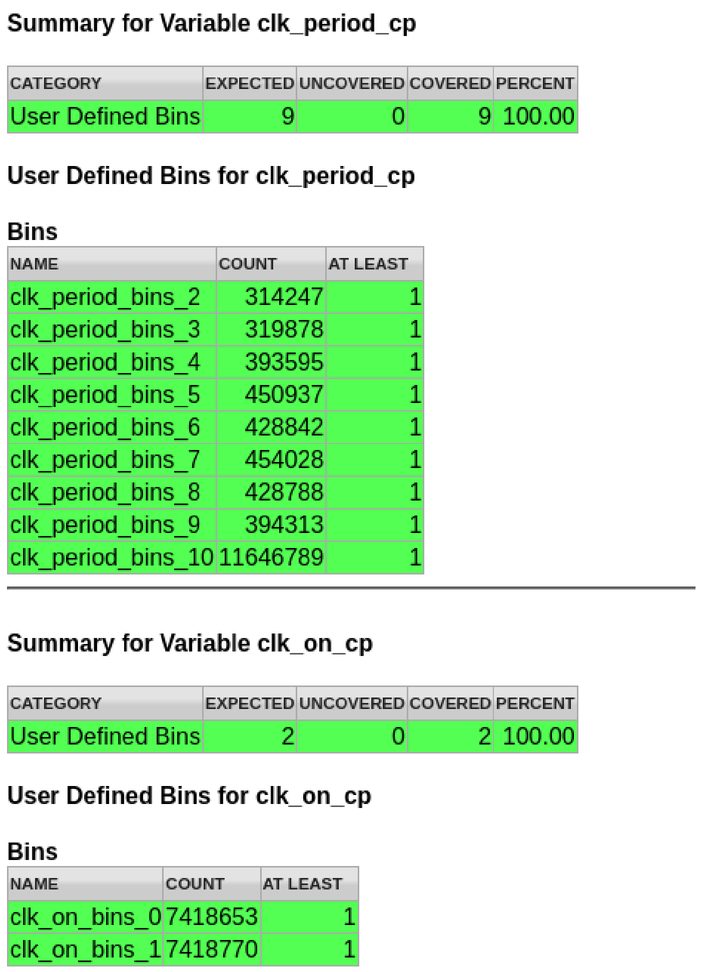
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	33	0	33	100.00

User Defined Bins for num_similar_letters

Bins

NAME	COUNT	AT LEAST
num_similar_letters_bins_0	666	1
num_similar_letters_bins_1	139	1
num_similar_letters_bins_2	371	1
num_similar_letters_bins_3	824	1
num_similar_letters_bins_4	2186	1
num_similar_letters_bins_5	1875	1
num_similar_letters_bins_6	2431	1
num_similar_letters_bins_7	2594	1
num_similar_letters_bins_8	3324	1
num_similar_letters_bins_9	2415	1
num_similar_letters_bins_10	1969	1
num_similar_letters_bins_11	1535	1
num_similar_letters_bins_12	1835	1
num_similar_letters_bins_13	871	1
num_similar_letters_bins_14	622	1
num_similar_letters_bins_15	491	1
num_similar_letters_bins_16	713	1
num_similar_letters_bins_17	293	1
num_similar_letters_bins_18	230	1
num_similar_letters_bins_19	196	1
num_similar_letters_bins_20	331	1
num_similar_letters_bins_21	190	1
num_similar_letters_bins_22	169	1
num_similar_letters_bins_23	189	1

Clock Coverage



Summary for Variable clk_period_cp

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	9	0	9	100.00

User Defined Bins for clk_period_cp

Bins

NAME	COUNT	AT LEAST
clk_period_bins_2	314247	1
clk_period_bins_3	319878	1
clk_period_bins_4	393595	1
clk_period_bins_5	450937	1
clk_period_bins_6	428842	1
clk_period_bins_7	454028	1
clk_period_bins_8	428788	1
clk_period_bins_9	394313	1
clk_period_bins_10	11646789	1

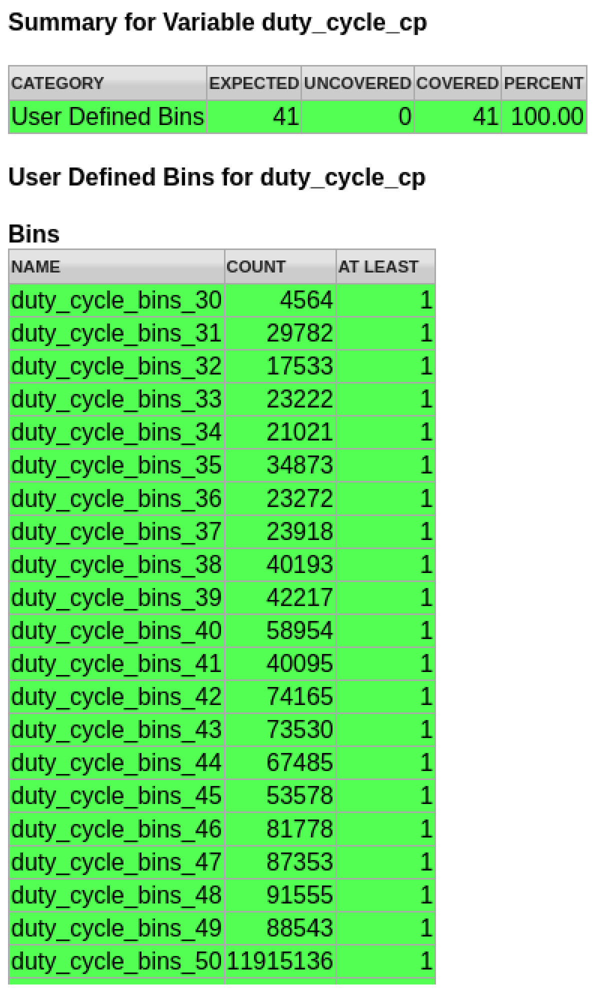
Summary for Variable clk_on_cp

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	2	0	2	100.00

User Defined Bins for clk_on_cp

Bins

NAME	COUNT	AT LEAST
clk_on_bins_0	7418653	1
clk_on_bins_1	7418770	1



Summary for Variable duty_cycle_cp

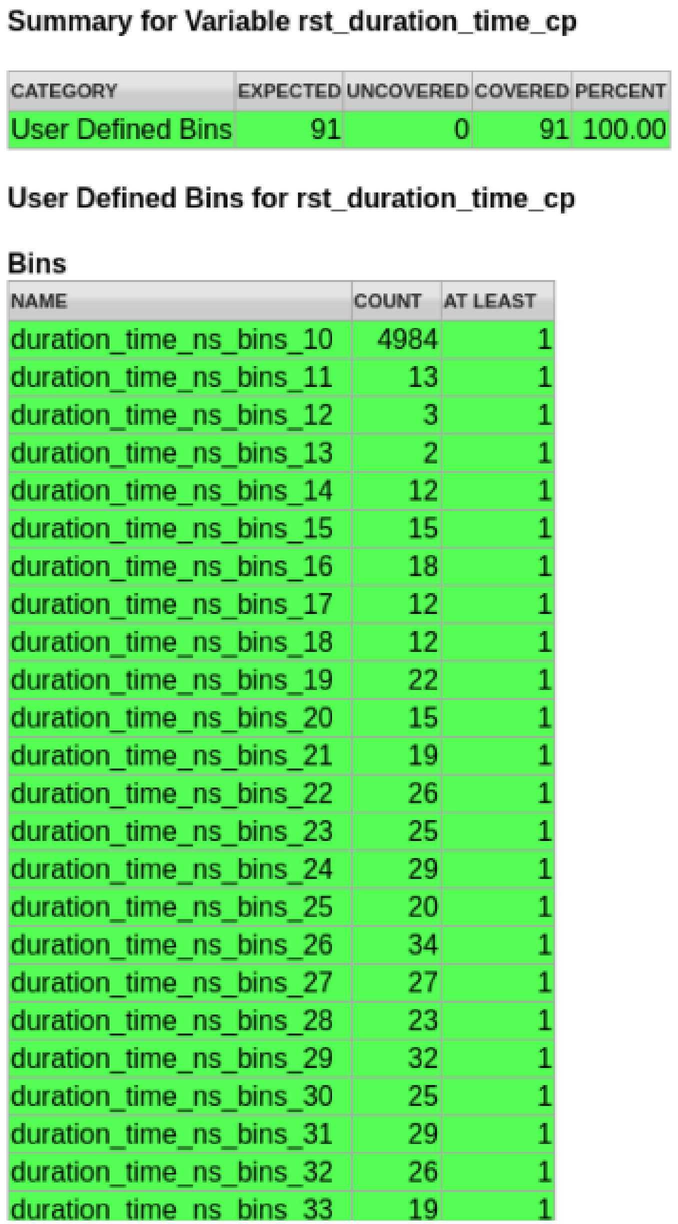
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	41	0	41	100.00

User Defined Bins for duty_cycle_cp

Bins

NAME	COUNT	AT LEAST
duty_cycle_bins_30	4564	1
duty_cycle_bins_31	29782	1
duty_cycle_bins_32	17533	1
duty_cycle_bins_33	23222	1
duty_cycle_bins_34	21021	1
duty_cycle_bins_35	34873	1
duty_cycle_bins_36	23272	1
duty_cycle_bins_37	23918	1
duty_cycle_bins_38	40193	1
duty_cycle_bins_39	42217	1
duty_cycle_bins_40	58954	1
duty_cycle_bins_41	40095	1
duty_cycle_bins_42	74165	1
duty_cycle_bins_43	73530	1
duty_cycle_bins_44	67485	1
duty_cycle_bins_45	53578	1
duty_cycle_bins_46	81778	1
duty_cycle_bins_47	87353	1
duty_cycle_bins_48	91555	1
duty_cycle_bins_49	88543	1
duty_cycle_bins_50	11915136	1

Reset Coverage



Summary for Variable rst_duration_time_cp

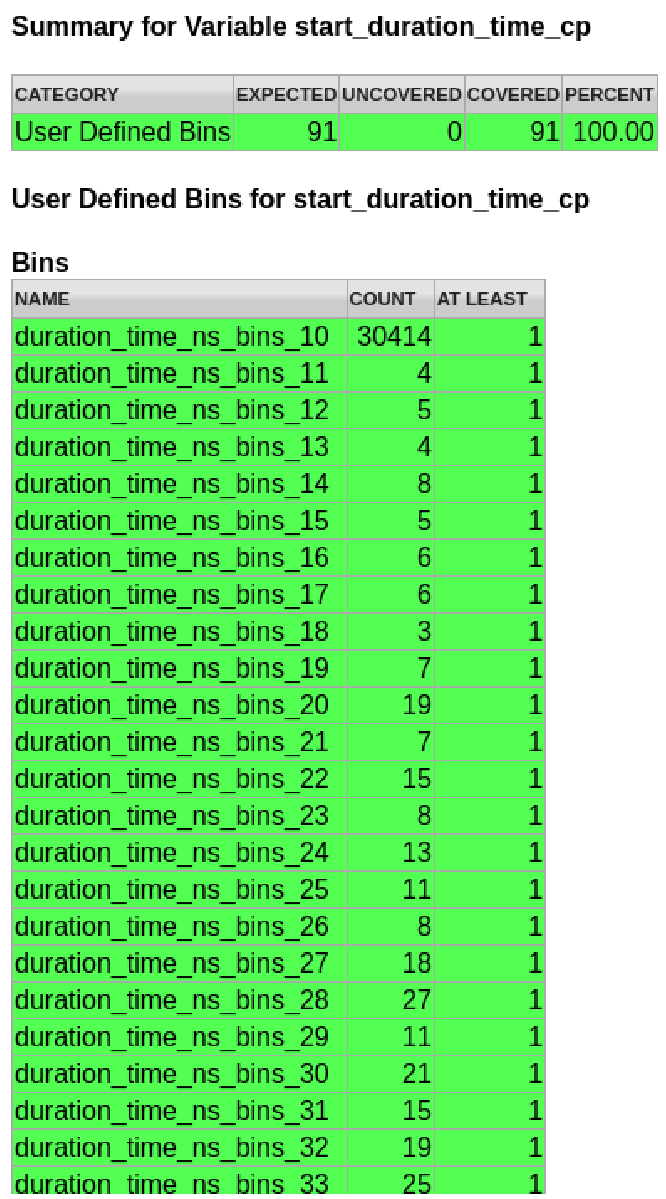
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	91	0	91	100.00

User Defined Bins for rst_duration_time_cp

Bins

NAME	COUNT	AT LEAST
duration_time_ns_bins_10	4984	1
duration_time_ns_bins_11	13	1
duration_time_ns_bins_12	3	1
duration_time_ns_bins_13	2	1
duration_time_ns_bins_14	12	1
duration_time_ns_bins_15	15	1
duration_time_ns_bins_16	18	1
duration_time_ns_bins_17	12	1
duration_time_ns_bins_18	12	1
duration_time_ns_bins_19	22	1
duration_time_ns_bins_20	15	1
duration_time_ns_bins_21	19	1
duration_time_ns_bins_22	26	1
duration_time_ns_bins_23	25	1
duration_time_ns_bins_24	29	1
duration_time_ns_bins_25	20	1
duration_time_ns_bins_26	34	1
duration_time_ns_bins_27	27	1
duration_time_ns_bins_28	23	1
duration_time_ns_bins_29	32	1
duration_time_ns_bins_30	25	1
duration_time_ns_bins_31	29	1
duration_time_ns_bins_32	26	1
duration_time_ns_bins_33	19	1

Start Coverage



Summary for Variable start_duration_time_cp

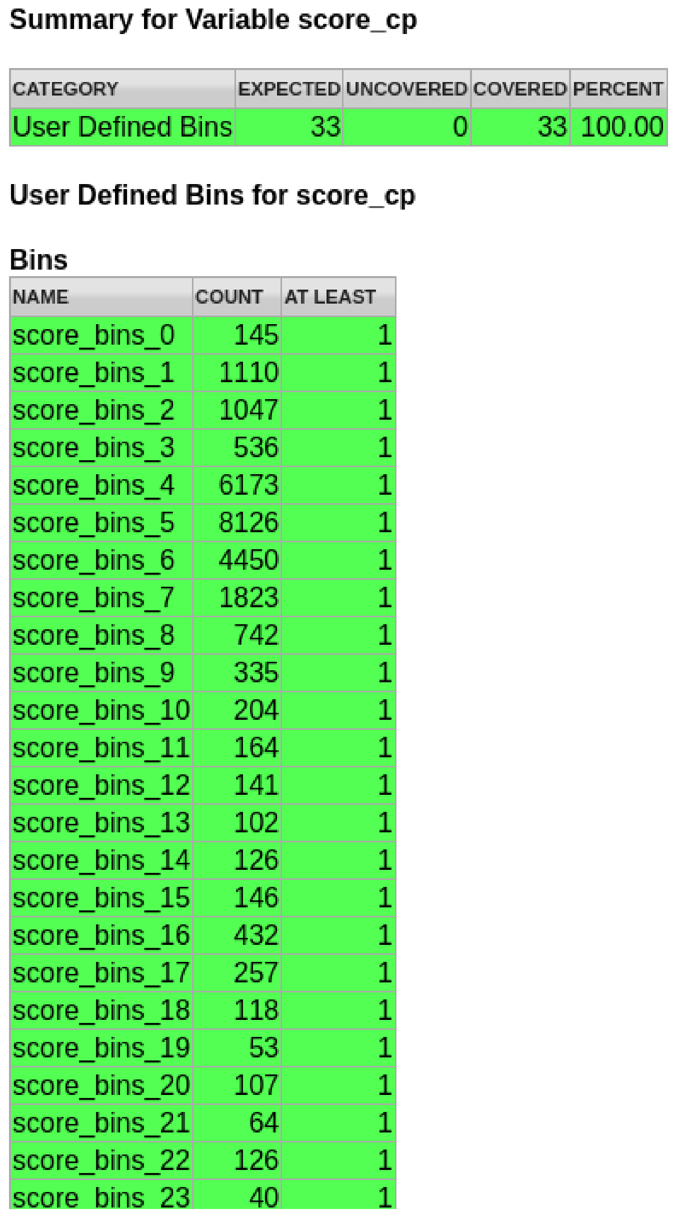
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	91	0	91	100.00

User Defined Bins for start_duration_time_cp

Bins

NAME	COUNT	AT LEAST
duration_time_ns_bins_10	30414	1
duration_time_ns_bins_11	4	1
duration_time_ns_bins_12	5	1
duration_time_ns_bins_13	4	1
duration_time_ns_bins_14	8	1
duration_time_ns_bins_15	5	1
duration_time_ns_bins_16	6	1
duration_time_ns_bins_17	6	1
duration_time_ns_bins_18	3	1
duration_time_ns_bins_19	7	1
duration_time_ns_bins_20	19	1
duration_time_ns_bins_21	7	1
duration_time_ns_bins_22	15	1
duration_time_ns_bins_23	8	1
duration_time_ns_bins_24	13	1
duration_time_ns_bins_25	11	1
duration_time_ns_bins_26	8	1
duration_time_ns_bins_27	18	1
duration_time_ns_bins_28	27	1
duration_time_ns_bins_29	11	1
duration_time_ns_bins_30	21	1
duration_time_ns_bins_31	15	1
duration_time_ns_bins_32	19	1
duration_time_ns_bins_33	25	1

Score Coverage



Summary for Variable score_cp

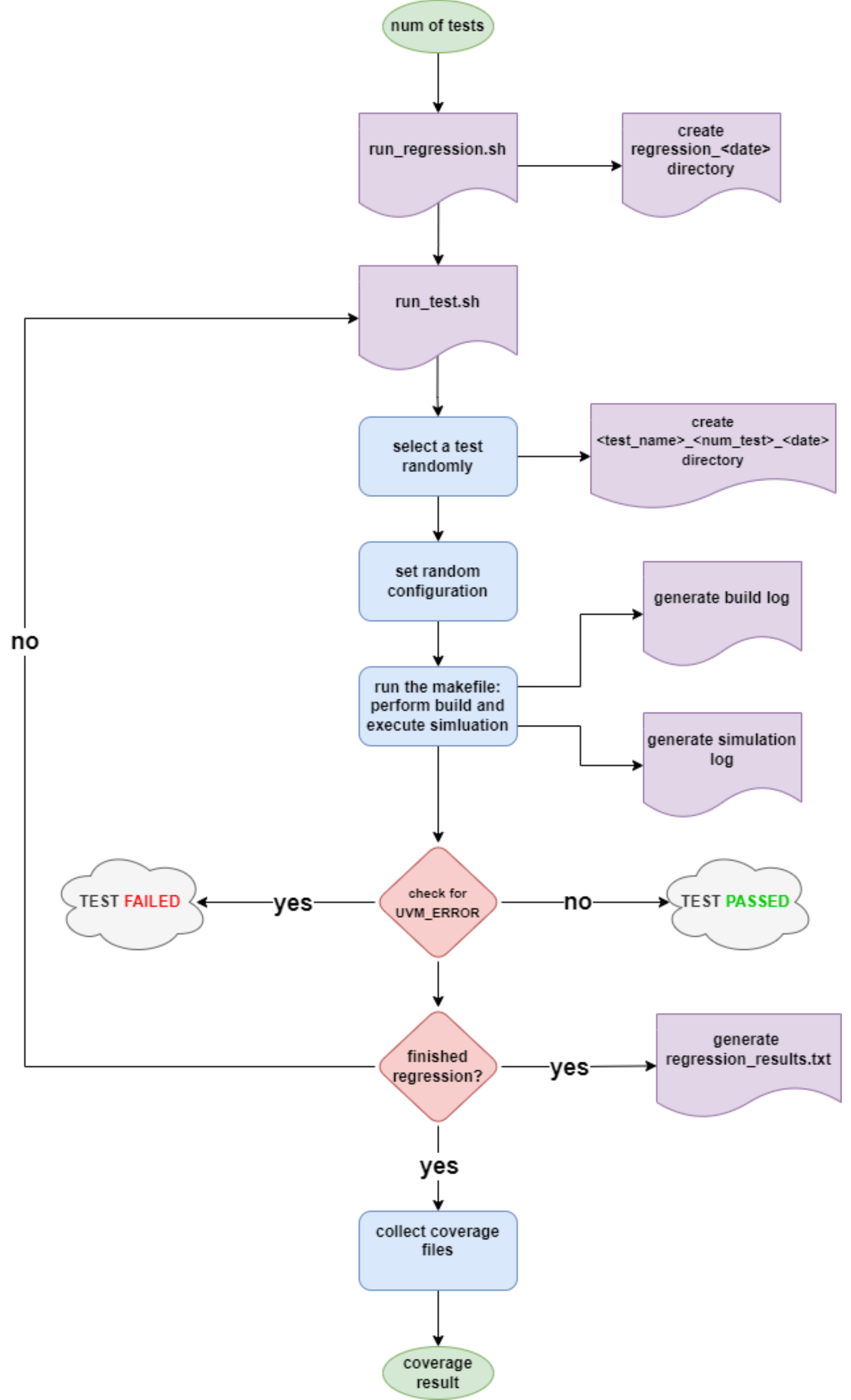
CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
User Defined Bins	33	0	33	100.00

User Defined Bins for score_cp

Bins

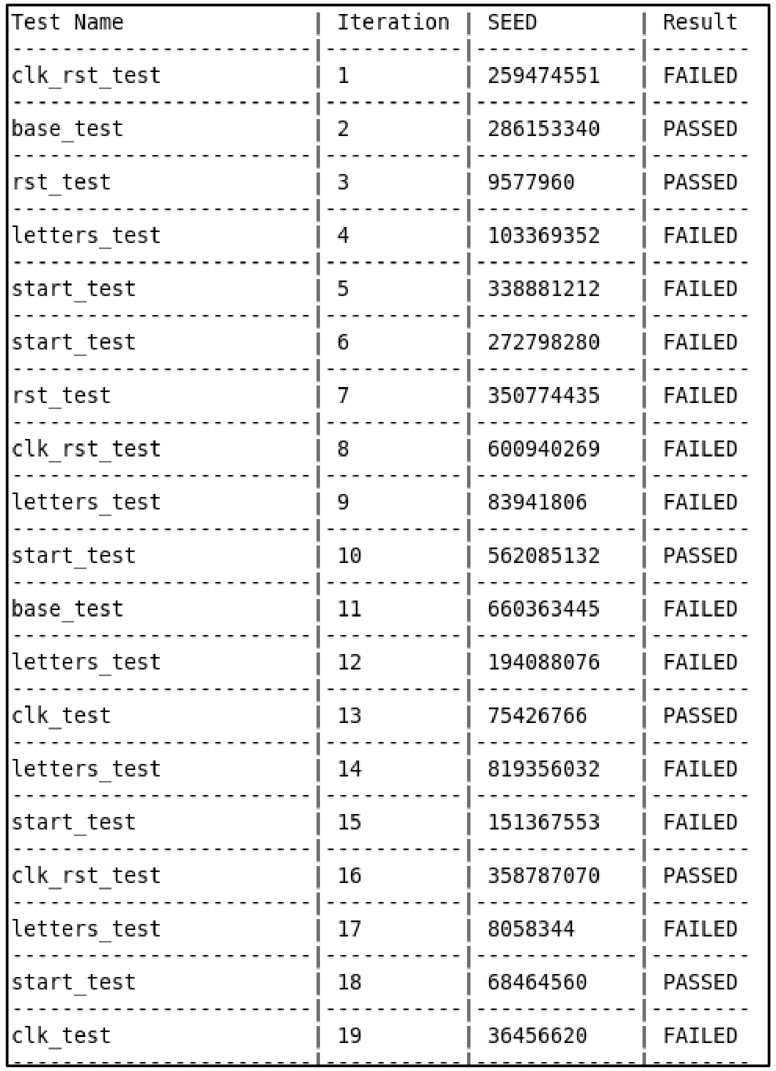
NAME	COUNT	AT LEAST
score_bins_0	145	1
score_bins_1	1110	1
score_bins_2	1047	1
score_bins_3	536	1
score_bins_4	6173	1
score_bins_5	8126	1
score_bins_6	4450	1
score_bins_7	1823	1
score_bins_8	742	1
score_bins_9	335	1
score_bins_10	204	1
score_bins_11	164	1
score_bins_12	141	1
score_bins_13	102	1
score_bins_14	126	1
score_bins_15	146	1
score_bins_16	432	1
score_bins_17	257	1
score_bins_18	118	1
score_bins_19	53	1
score_bins_20	107	1
score_bins_21	64	1
score_bins_22	126	1
score_bins_23	40	1

Regression Flow



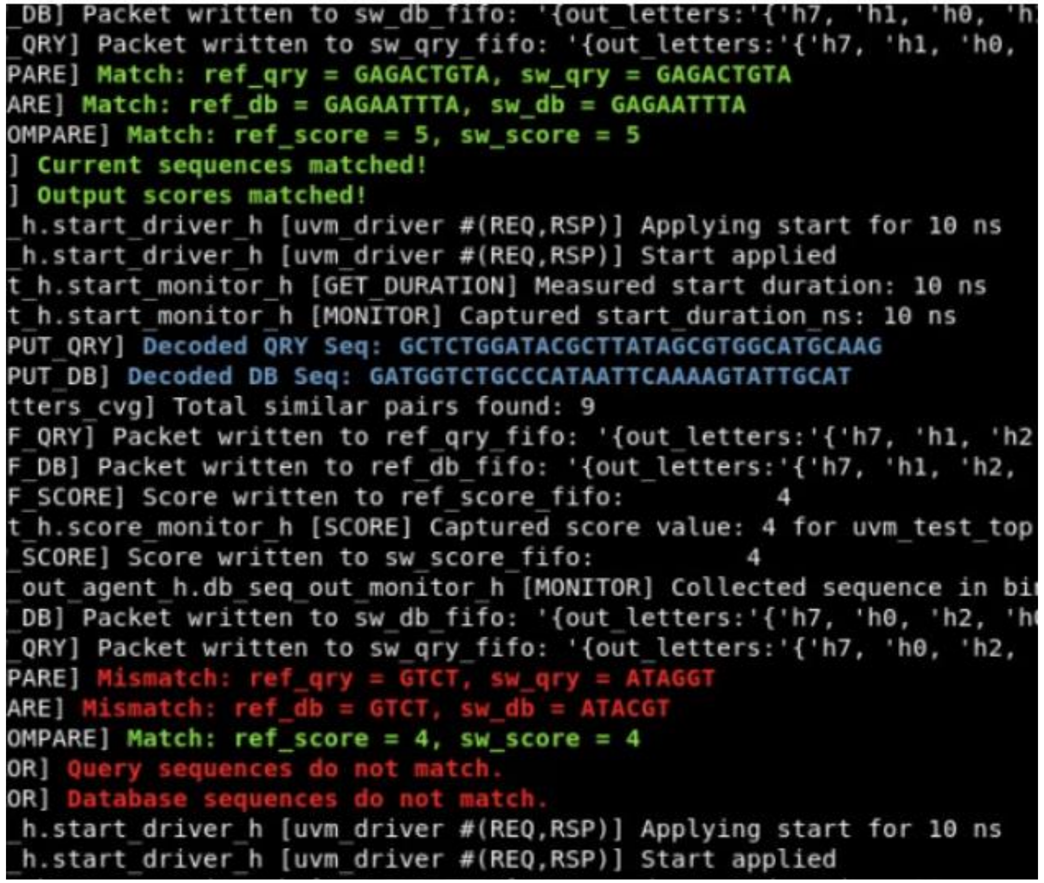
Regression Results

- The SW accelerator demonstrated a flawless design, combining an effective coding approach with accurate functionality. However:



```
Test Name                | Iteration | SEED          | Result
-------------------------|-----------|---------------|--------
clk_rst_test             | 1         | 259474551     | FAILED
-------------------------|-----------|---------------|--------
base_test                | 2         | 286153340     | PASSED
-------------------------|-----------|---------------|--------
rst_test                 | 3         | 9577960       | PASSED
-------------------------|-----------|---------------|--------
letters_test             | 4         | 103369352     | FAILED
-------------------------|-----------|---------------|--------
start_test               | 5         | 338881212     | FAILED
-------------------------|-----------|---------------|--------
start_test               | 6         | 272798280     | FAILED
-------------------------|-----------|---------------|--------
rst_test                 | 7         | 350774435     | FAILED
-------------------------|-----------|---------------|--------
clk_rst_test             | 8         | 600940269     | FAILED
-------------------------|-----------|---------------|--------
letters_test             | 9         | 83941806      | FAILED
-------------------------|-----------|---------------|--------
start_test               | 10        | 562085132     | PASSED
-------------------------|-----------|---------------|--------
base_test                | 11        | 660363445     | FAILED
-------------------------|-----------|---------------|--------
letters_test             | 12        | 194088076     | FAILED
-------------------------|-----------|---------------|--------
clk_test                 | 13        | 75426766      | PASSED
-------------------------|-----------|---------------|--------
letters_test             | 14        | 819356032     | FAILED
-------------------------|-----------|---------------|--------
start_test               | 15        | 151367553     | FAILED
-------------------------|-----------|---------------|--------
clk_rst_test             | 16        | 358787070     | PASSED
-------------------------|-----------|---------------|--------
letters_test             | 17        | 8058344       | FAILED
-------------------------|-----------|---------------|--------
start_test               | 18        | 68464560      | PASSED
-------------------------|-----------|---------------|--------
clk_test                 | 19        | 36456620      | FAILED
-------------------------|-----------|---------------|--------
```

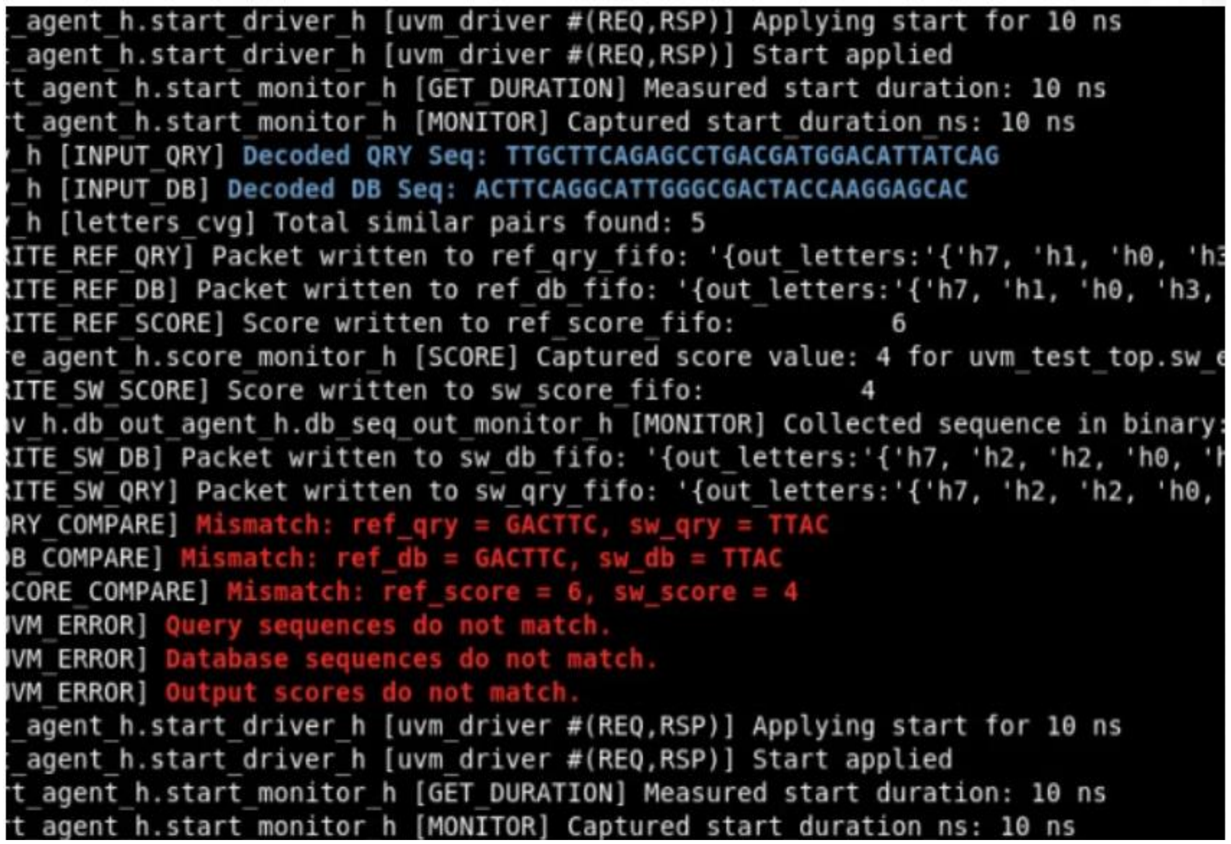
Regression Results



```
_DB] Packet written to sw_db_fifo: '{out_letters:'{'h7, 'h1, 'h0, 'h
_QRY] Packet written to sw_qry_fifo: '{out_letters:'{'h7, 'h1, 'h0,
PARE] Match: ref_qry = GAGACTGTA, sw_qry = GAGACTGTA
ARE] Match: ref_db = GAGAATTTA, sw_db = GAGAATTTA
OMPARE] Match: ref_score = 5, sw_score = 5
] Current sequences matched!
] Output scores matched!
_h.start_driver_h [uvm_driver #(REQ,RSP)] Applying start for 10 ns
_h.start_driver_h [uvm_driver #(REQ,RSP)] Start applied
t_h.start_monitor_h [GET_DURATION] Measured start duration: 10 ns
t_h.start_monitor_h [MONITOR] Captured start_duration_ns: 10 ns
PUT_QRY] Decoded QRY Seq: GCTCTGGATACGCTTATAGCGTGGCATGCAAG
PUT_DB] Decoded DB Seq: GATGGTCTGCCCATAATTCAAAAGTATTGCAT
tters_cvg] Total similar pairs found: 9
F_QRY] Packet written to ref_qry_fifo: '{out_letters:'{'h7, 'h1, 'h2
F_DB] Packet written to ref_db_fifo: '{out_letters:'{'h7, 'h1, 'h2,
F_SCORE] Score written to ref_score_fifo:          4
t_h.score_monitor_h [SCORE] Captured score value: 4 for uvm_test_top
_SCORE] Score written to sw_score_fifo:          4
_out_agent_h.db_seq_out_monitor_h [MONITOR] Collected sequence in bi
_DB] Packet written to sw_db_fifo: '{out_letters:'{'h7, 'h0, 'h2, 'h
_QRY] Packet written to sw_qry_fifo: '{out_letters:'{'h7, 'h0, 'h2,
PARE] Mismatch: ref_qry = GTCT, sw_qry = ATAGGT
ARE] Mismatch: ref_db = GTCT, sw_db = ATACGT
OMPARE] Match: ref_score = 4, sw_score = 4
OR] Query sequences do not match.
OR] Database sequences do not match.
_h.start_driver_h [uvm_driver #(REQ,RSP)] Applying start for 10 ns
_h.start_driver_h [uvm_driver #(REQ,RSP)] Start applied
```

- We observed that the traceback process is not correct in some cases. Although the maximum score remains the same, the alignment is incorrect.

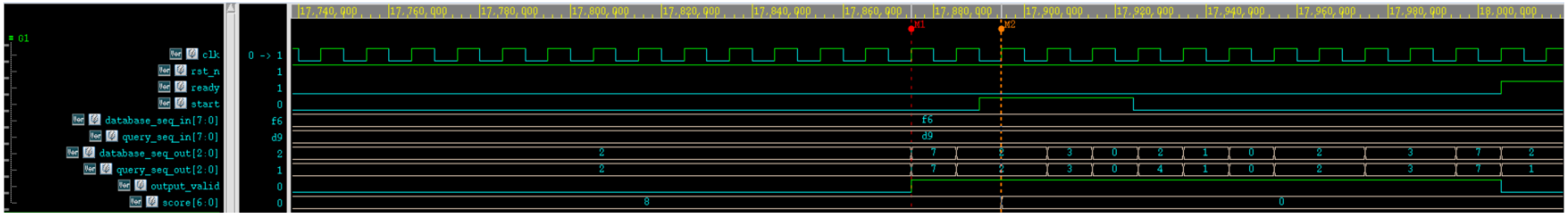
Regression Results



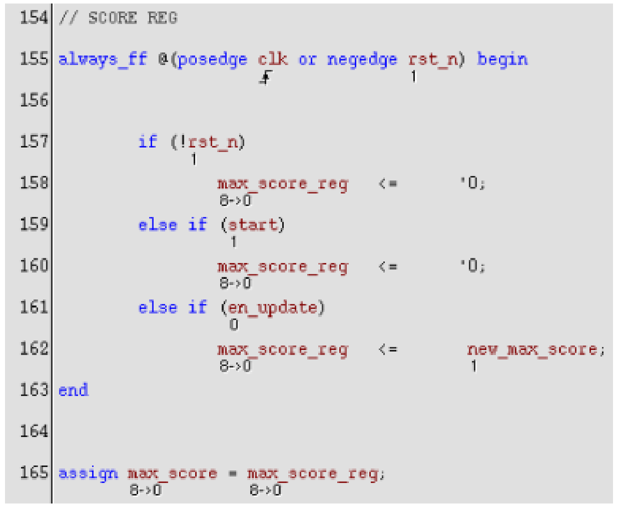
```
_agent_h.start_driver_h [uvm_driver #(REQ,RSP)] Applying start for 10 ns
_agent_h.start_driver_h [uvm_driver #(REQ,RSP)] Start applied
t_agent_h.start_monitor_h [GET_DURATION] Measured start duration: 10 ns
t_agent_h.start_monitor_h [MONITOR] Captured start_duration_ns: 10 ns
_h [INPUT_QRY] Decoded QRY Seq: TTGCTTCAGAGCCTGACGATGGACATTATCAG
_h [INPUT_DB] Decoded DB Seq: ACTTCAGGCATTGGGCGACTACCAAGGAGCAC
_h [letters_cvg] Total similar pairs found: 5
ITE_REF_QRY] Packet written to ref_qry_fifo: '{out_letters:'{'h7, 'h1, 'h0, 'h
ITE_REF_DB] Packet written to ref_db_fifo: '{out_letters:'{'h7, 'h1, 'h0, 'h3,
ITE_REF_SCORE] Score written to ref_score_fifo:          6
e_agent_h.score_monitor_h [SCORE] Captured score value: 4 for uvm_test_top.sw_e
ITE_SW_SCORE] Score written to sw_score_fifo:          4
v_h.db_out_agent_h.db_seq_out_monitor_h [MONITOR] Collected sequence in binary:
ITE_SW_DB] Packet written to sw_db_fifo: '{out_letters:'{'h7, 'h2, 'h2, 'h0, '
ITE_SW_QRY] Packet written to sw_qry_fifo: '{out_letters:'{'h7, 'h2, 'h2, 'h0,
RY_COMPARE] Mismatch: ref_qry = GACTTC, sw_qry = TTAC
B_COMPARE] Mismatch: ref_db = GACTTC, sw_db = TTAC
CORE_COMPARE] Mismatch: ref_score = 6, sw_score = 4
VM_ERROR] Query sequences do not match.
VM_ERROR] Database sequences do not match.
VM_ERROR] Output scores do not match.
_agent_h.start_driver_h [uvm_driver #(REQ,RSP)] Applying start for 10 ns
_agent_h.start_driver_h [uvm_driver #(REQ,RSP)] Start applied
t_agent_h.start_monitor_h [GET_DURATION] Measured start duration: 10 ns
t_agent_h.start_monitor_h [MONITOR] Captured start_duration_ns: 10 ns
```

- we found that the calculation process is wrong in some cases. We can observe this by looking at the max score we got from the reference model and the chip.

BUG Found



```
"./if_checker.sv", 33: tb_top.if_checker_i.unnamed$$_1: started at 17905000ps failed at 17905000ps
        Offending '(sw_if.score == $past(sw_if.score))'
UVM_ERROR ./if_checker.sv(34) @ 17905000: reporter [ASSERTION_ERROR] Error: Score changed while output_valid was asserted for consecutive cycles.
```



```
// SCORE REG
always_ff @(posedge clk or negedge rst_n) begin
                    ↿               1

        if (!rst_n)
            1
                max_score_reg   <=      '0;
                8->0
        else if (start)
                 1
                max_score_reg   <=      '0;
                8->0
        else if (en_update)
                 0
                max_score_reg   <=      new_max_score;
                8->0                    1
end

assign max_score = max_score_reg;
          8->0         8->0
```

Conclusions

- **Redundant Code**: certain code sections will never be satisfied, leading to redundant hardware, so they should be excluded from synthesis to optimize area and power.
- **Verification is important** : Verification found corner cases where the design fails, particularly in the calculation and traceback processes, despite generally functioning as expected.
- **Robust flow is a must**: we want it to be dynamic, easy to get feedback from and easy to maintain.
- **SVA help find bugs fast**: it is necessary to write assertions as part of the design to find bugs in an early stage.

Future work

- Our UVM environment is highly flexible, with generically designed components that can be easily reused across different projects with minimal changes
- Our coverage and constraint mechanisms are designed for broad applicability, supporting a variety of verification needs with only minor adjustments.

This project serves as the verification infrastructure for large-scale local sequence alignment.