# Digital Verification using SV and UVM

# Assignment-3

# Name: Fares Khalaf Sultan

# Q1) ALU:

# Package Code:

# 

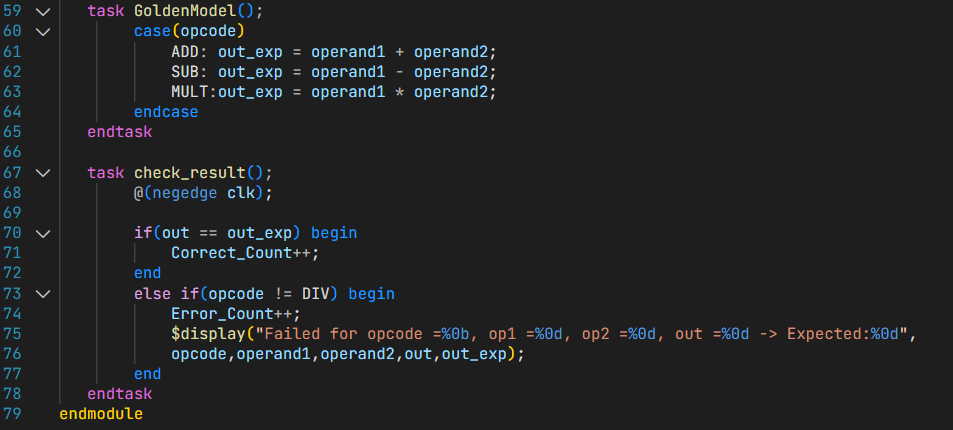
# 

# 

# testbench:

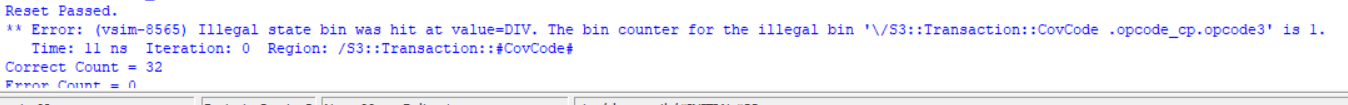
# 

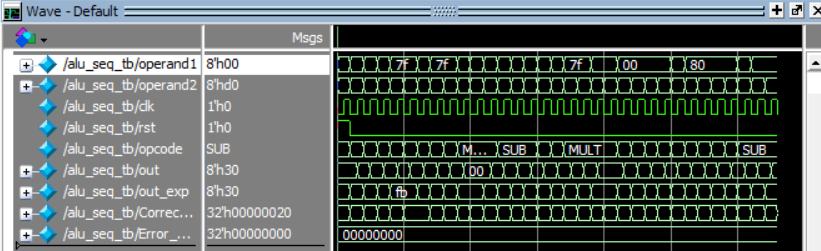
# 



# Do file:

# Result:



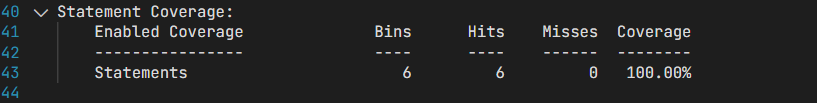
Waveform:

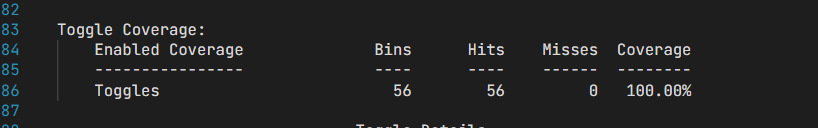
# Functional coverage:

# 

# Code coverage:

**Excluded the default case for the opcode, because it is unreachable since all possible opcode values are handled in the case statement.**

****

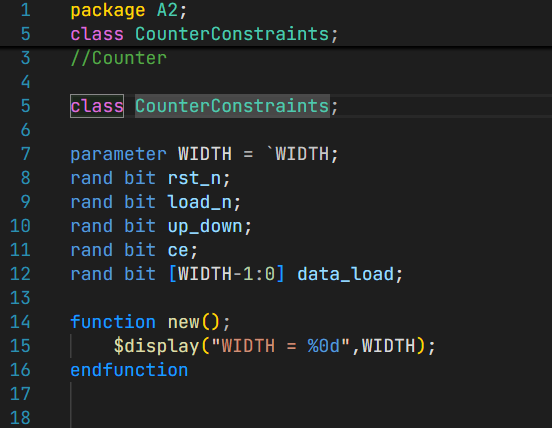


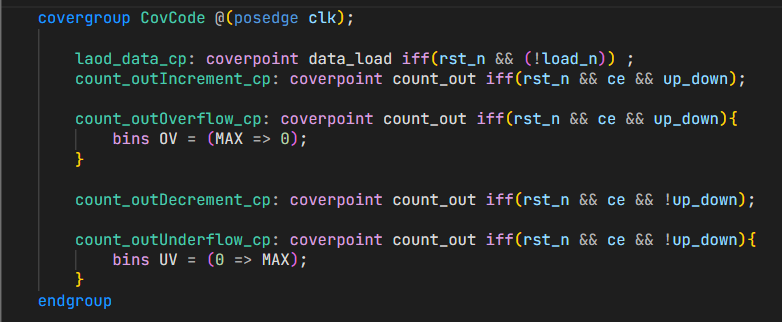
# Q2) Counter:

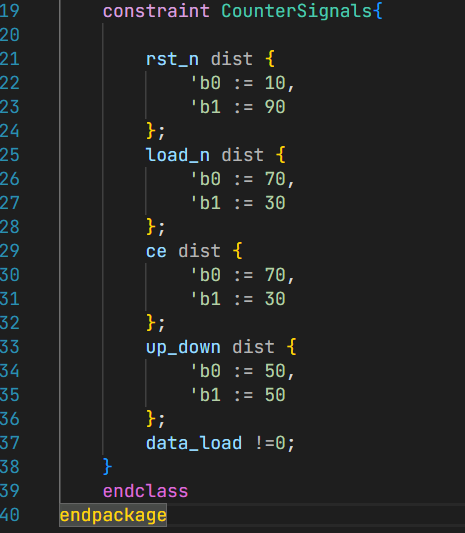
# Verification Plan:

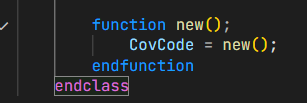
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Design Requirement Description** | **Stimulus Generation** | **Functional Coverage** | **Functionality Check** |
| Counter1 | when reset is asserted, Output should be low, and **zero** should be high | Directed at the start of the simulation, then randomized with constraint to be inactive 90% of time during the simulation | - | A checker in the testbench to make sure the output is correct |
| Counter2 | When **load\_n** is low, **count\_out** should take the value of **load\_data** input | Randomization with constraint on **load\_n** to be high 70% of simulation time | Cover all values of load\_data | Output Checked against golden model |
| Counter3 | Counter should only increment or decrement if **rst\_n** is inactive and, **ce** signal is high else keep the current **count\_out** value. | Randomization with constraint on **ce** to be high 70% of simulation time | Cover all values of count\_out& transition bin from max to zero | Output Checked against golden model |
| Counter4 | If rst\_n is disabled and ce is enabeled, if: **up\_down = o → decrement**  **Up\_down = 1 → increment** | Randomization with constraint on **up\_down** to be high 50% of simulation time | Bin for decrement & another for increment | Output Checked against golden model |
| Counter5 | Check that when the bus **count\_out** value equals the max possible value, **max\_count** should be high | Randomized with no constraints | Bin for max\_count | Output Checked against golden model |

# Constraints class:

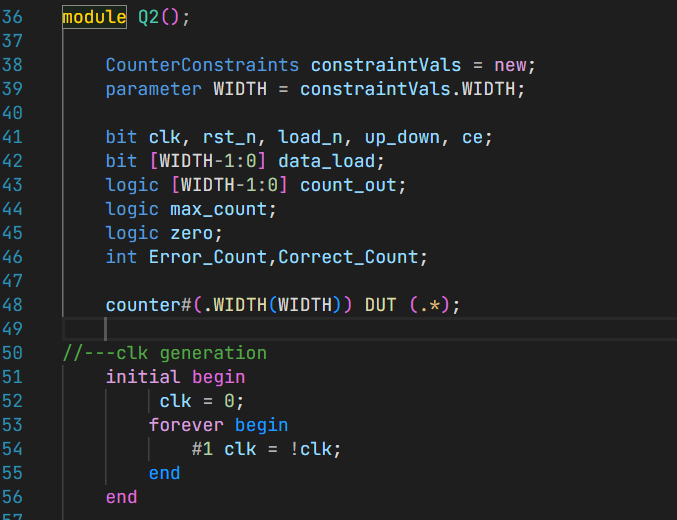


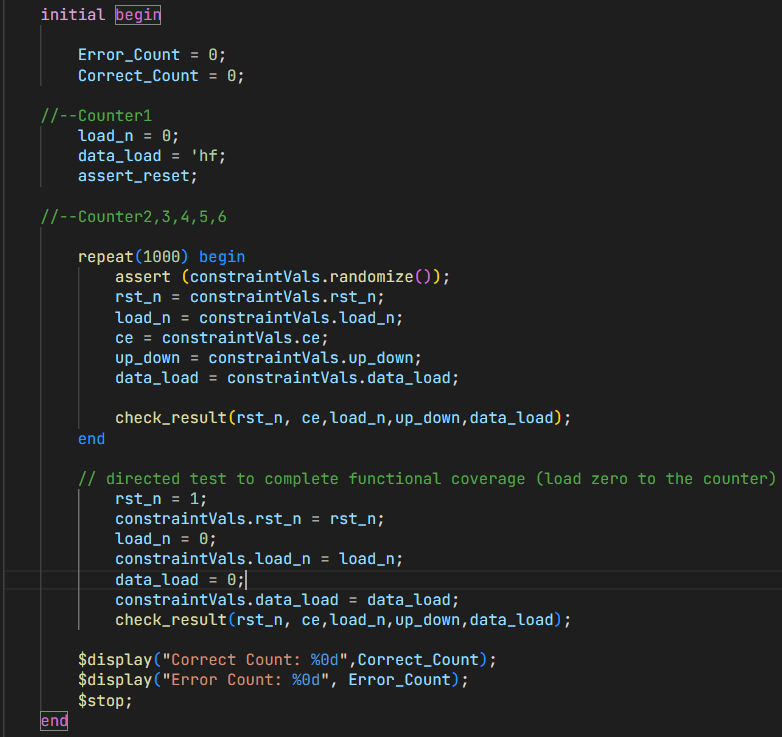


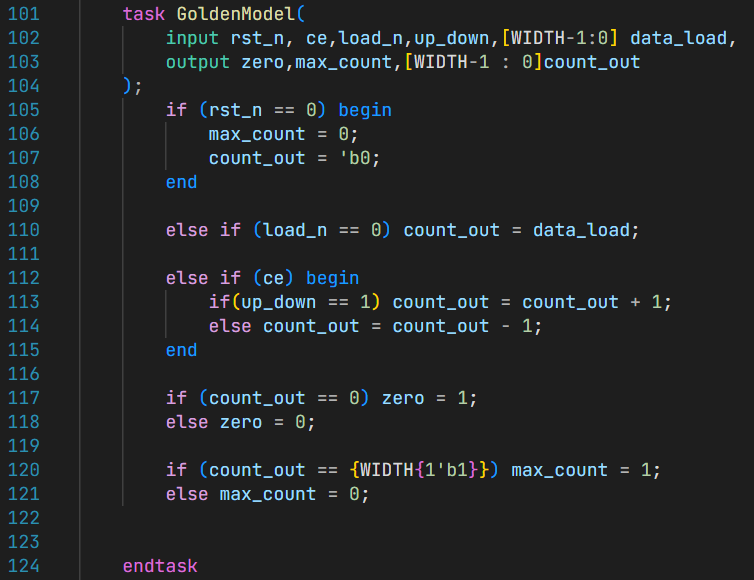
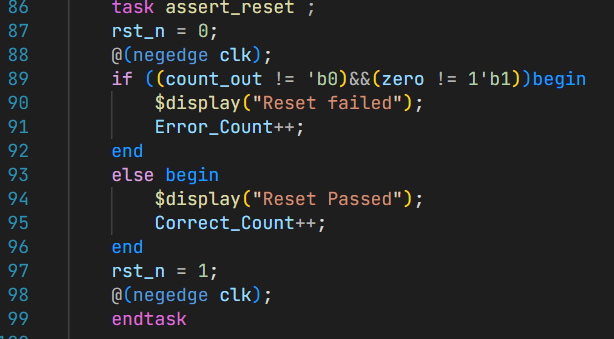


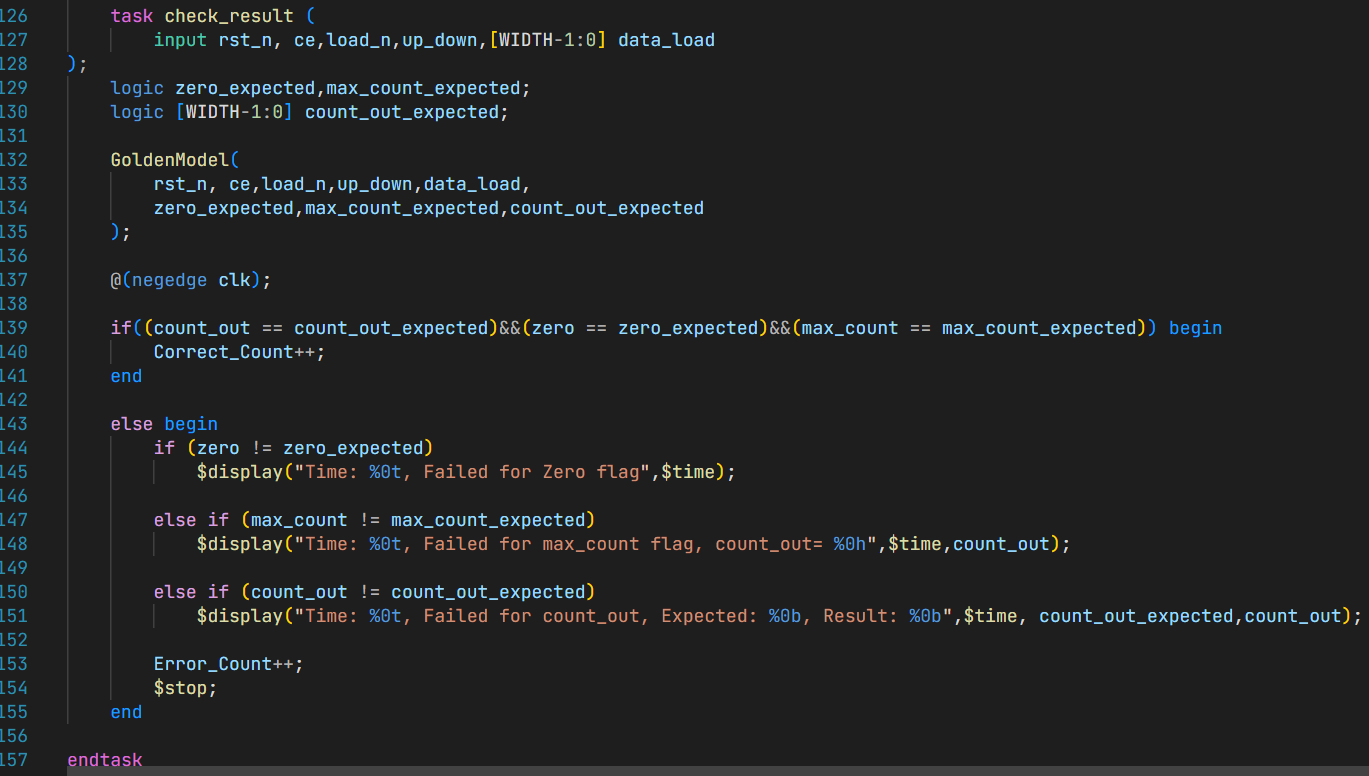


# Test bench:







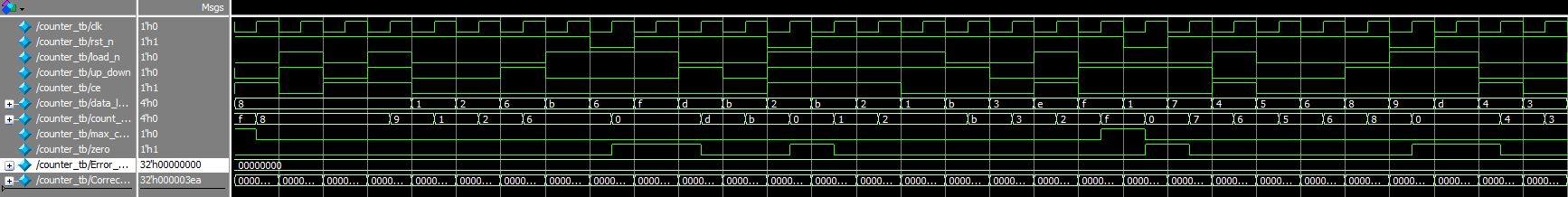


# Do file:

# 

# Result:

# 

Waveform:

# Functional Coverage:

# 

# Code Coverage :

# 

# 

# Q3) ALSU

# Verification Plan:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Design Requirement Description** | **Stimulus Generation** | **Functional Coverage** | **Functionality Check** |
| ALSU1 | when reset is asserted, Outputs should be Low for at least one clk cycle. | Directed at the start of the simulation | - | A checker in the testbench to make sure the output is correct |
| ALSU2 | In case opcode = ADD/MULT, & no invalid case, **LEDs** should = 0 &**out** should = A + B or A\*B | Randomized under constraint of having A and B equals to( Max., Min., Zero) most of the time, | **Bins\_arith[],** Ensures the happeness of this opcode values. | Comparing results to a refrence golden model |
| ALSU3 | If **opcode = OR/XOR** and no invalid cases, if both red\_op\_A/B are low, **out = A OR/XOR B**, else if red\_op\_A is high, **out = redOR/XOR A**, else if red\_op\_A is low and red\_op\_b is high,  **out = redOR/XOR B** | Randomized under constraint of having A/B having a high bit is reduction operation | **Bins\_bitwise[],** Ensures the happeness of this opcode values. | Comparing results to a refrence golden model |
| ALSU4 | verify Shift/ Rotate operations | Random during the simulation | **Bins\_shift[],** Ensures the happeness of this opcode values. | Comparing results to a refrence golden model |
| ALSU5 | Verifying correct behaviour if invalid operation happens, **LEDs** should blink and **out** should be low,  if bypass → out = the bypassed signal with higher priority for A | Randomized under constraints of having less invalid operations, and low probabity of having a bypass signal high | Bin\_invalid[] | Comparing results to a refrence golden model |

# Package class:

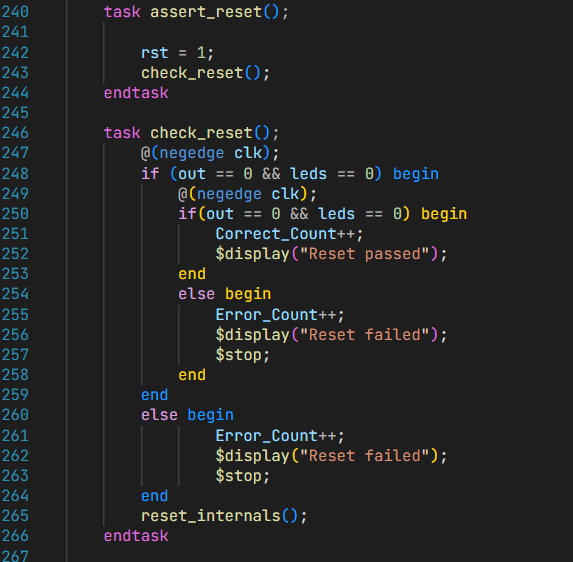
# 

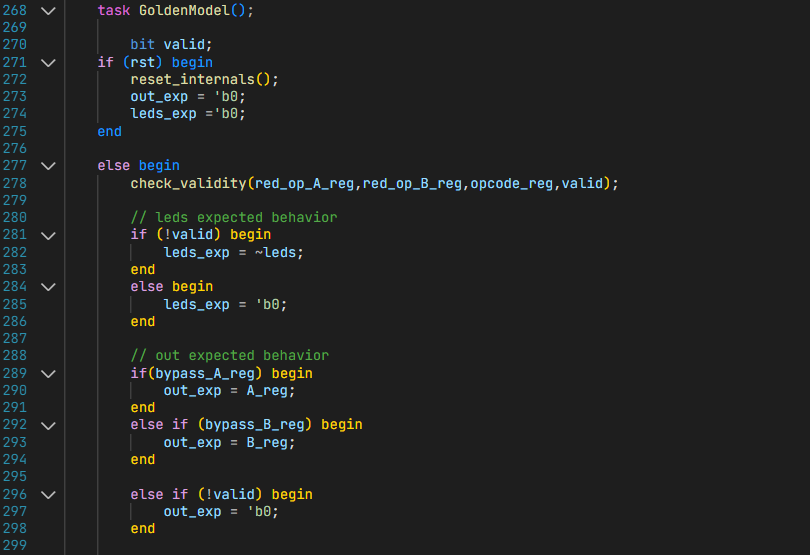
# 

# TestBench:

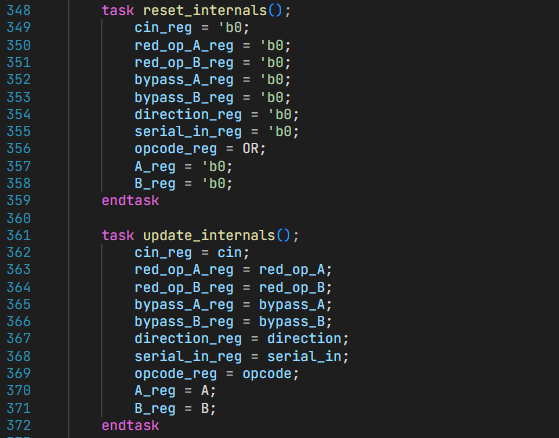
# 

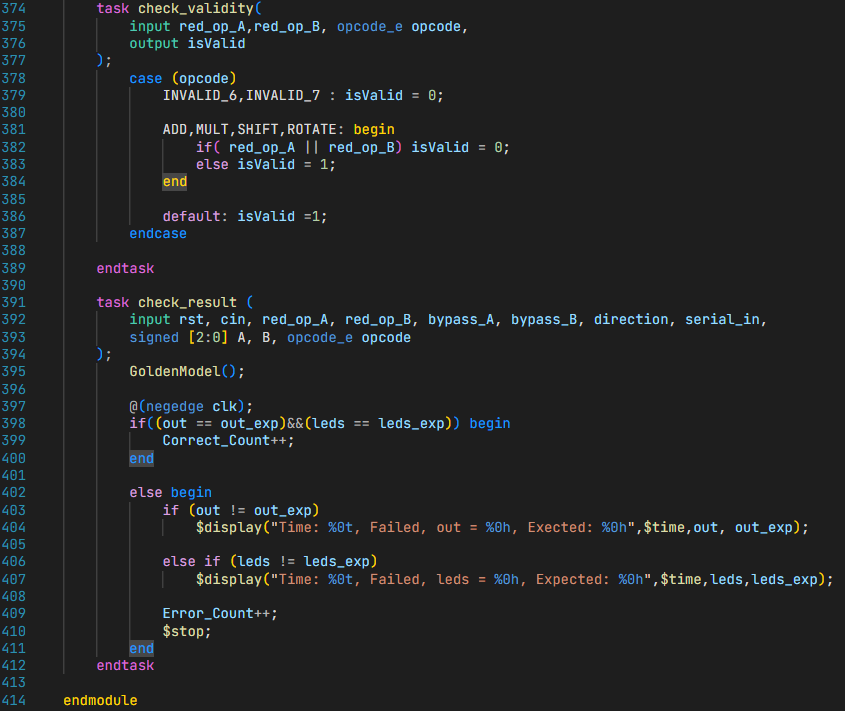
# 











# Do file:

# Bugs found:

|  |  |  |
| --- | --- | --- |
| **Bug Description** | **Wrong code** | **Correction** |
| Internal wire (cin\_reg) was defined to be signed, which causes sign extension in the addition processes. |  | Cin\_reg should be unsigned, and no need to have a width more than one bit |
| The case statement should use the value of opcode\_reg |  | Opcode → opcode\_reg |
| Missing FULL\_ADDER parameter check when opcode = add, hence, cin wasn’t taken into consideration |  |  |

# Results:

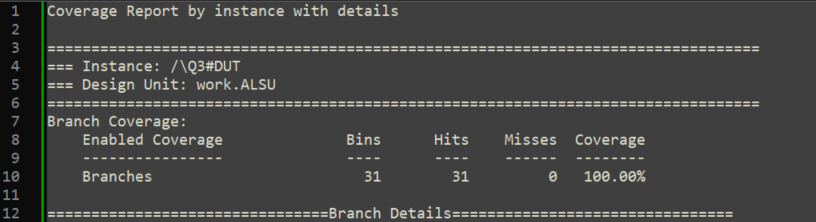
# 

# Waveform:

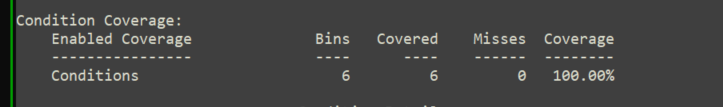
# Functional Coverage Report:

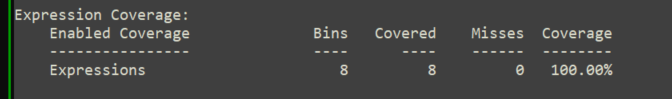
# 

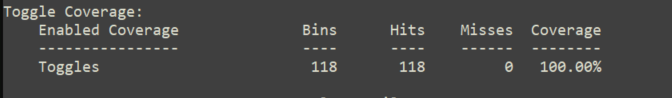
# Coverage Report:

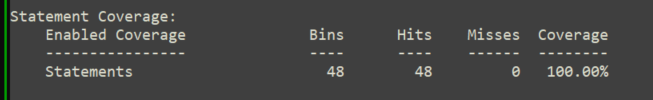


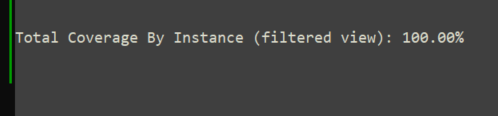
* Excluded **all False case** in the case statement, since invalid opcodes are handled by the flag (**invalid\_opcode**)











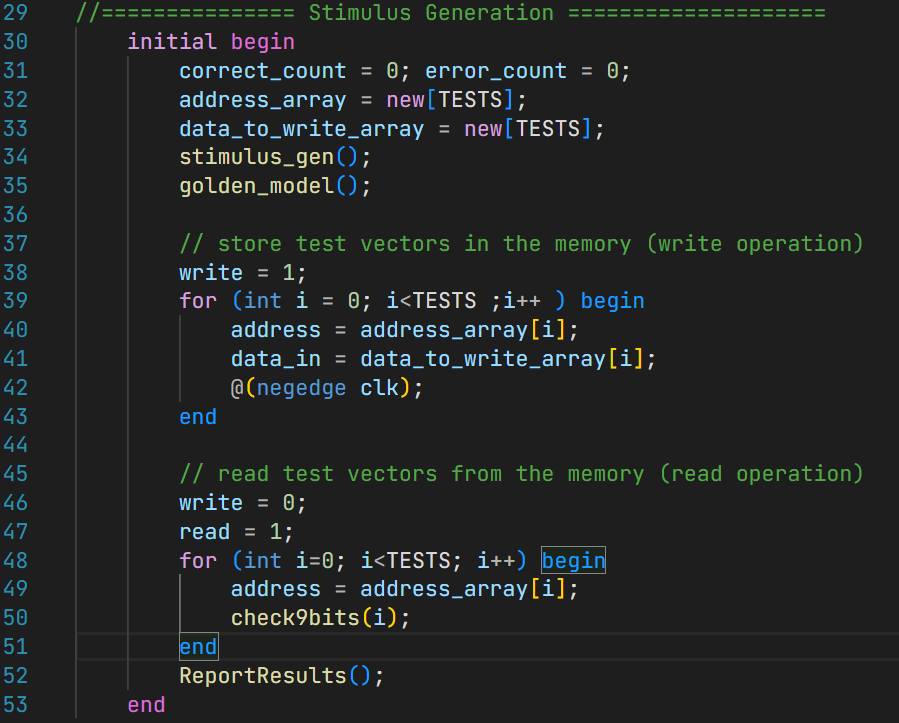
# Q4) RAM

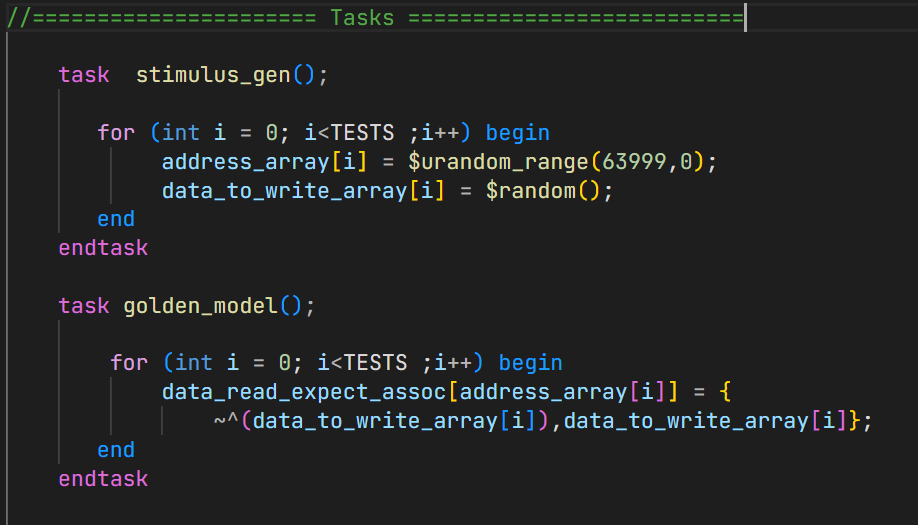
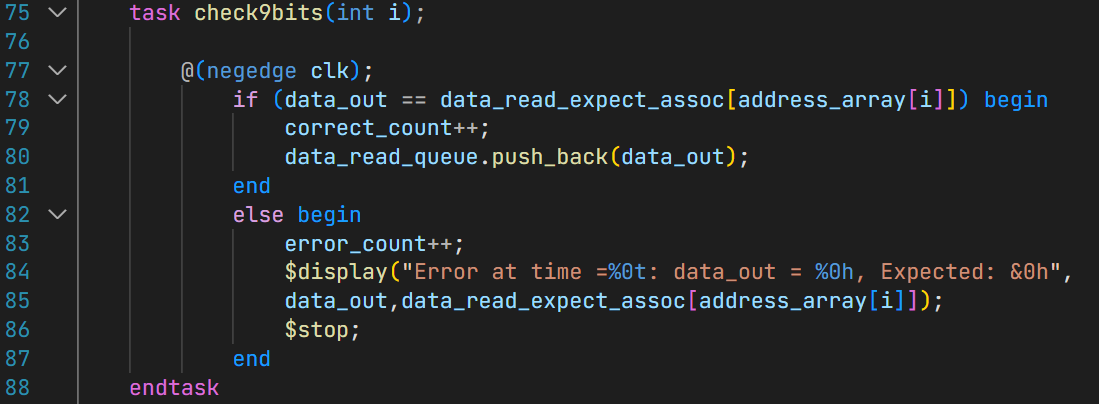
# Verification Plan:

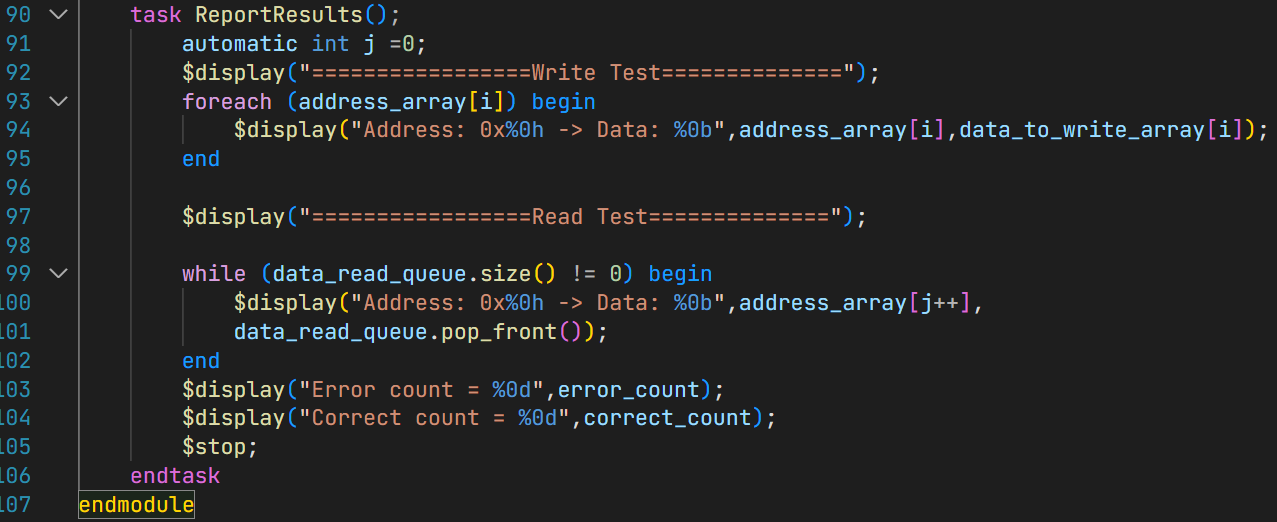
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Design Requirement Description** | **Stimulus Generation** | **Functional Coverage** | **Functionality Check** |
| RAM1 | Check write functionality by generating 100 random values to be written on 100 different memory locations. | Random during the simulation | - | Comparing results to a refrence golden model |
| RAM2 | Check read functionality by reading the random 100 test verctors written and compare them to the excpected values. | Random during the simulation | **-** | Comparing results to a refrence golden model |

# TestBench:

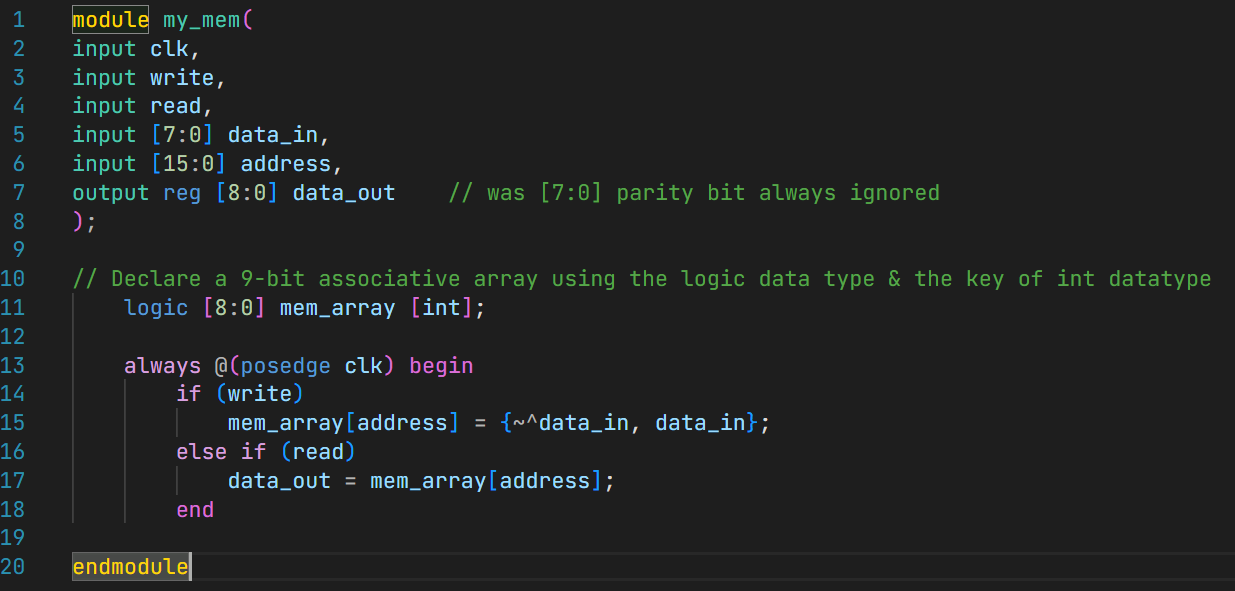
# 







# Fixed Design:



# Do file:

# Results:

# 

# 

# Waveform:

# 