# TEST PLAN

## Introduction

The Design Under Test (DUT) is an accumulator that is capable of taking in N 8-bit data. It adds them and outputs the result. It makes use of the RDY-EN protocol and features a RESET input that resets the DUT to zero. The number of bytes the accumulator can accumulate depends on the length programmed into the DUT through a port at the interface or configuration register.

## Objectives

The objective of this test plan is to verify whether the DUT correctly adds each input byte and produces the expected output based on the set length.

## Design specification

It uses the RDY EN Protocol on all interfaces

The DUT takes N 8 bit input values. Adds them and outputs the result.

The Value of N can be provided either

* At the Interface via the `length` port
* Or via the length register accessible via the configuration register.

The decision on which length is used (port or register) is controlled via a bit in the configuration space.

Once the DUT starts accumulating the bytes it ignores changes to the length field or register until it has generated the output.

## Interfaces

| Port | Direction | Width | Description |
|---|---|---|---|
| din_rdy | out | 1 | RDY for Input data |
| din_en | in | 1 | EN for Input data |
| din_value | in | 8 | Data |
| dout_rdy | out | 1 | RDY for Output data |
| dout_en | in | 1 | EN for output data |
| dout_value | out | 8 | Accumulated data output |
| len_rdy | out | 1 | RDY for Length |

| | | | |
|---|---|---|---|
| len_en | in | 1 | EN for length |
| len_value | in | 8 | Number of bytes that should be accumulated |
| cfg_rdy | out | 1 | RDY for Configuration Interface |
| cfg_en | in | 1 | EN for Configuration Interface |
| cfg_address | in | 8 | Address of the register  in cfg space |
| cfg_op | in | 1 | Operation type, 0=Read, 1=Write |
| cfg_data_in | in | 32 | The data that needs to be written, ignored for read |
| cfg_data_out | out | 32 | The data returned by read operation |

## Configuration Space Register Map

| Address | Access | Bit Map | Reset Value | Field | Description |
|---|---|---|---|---|---|
| | | | | | |
| 0 | R | 7:0 | 0 | current_count | The count of bytes processed |
| 0 | R | 15:8 | 0 | programmed_length | The length programmed for this session |
| 0 | R | 16 | 0 | busy | An operation has started and is ongoing |
| 4 | R/W | 0 | 0 | s/w override | 0 => use len from port.<br>1=>use len |
| 4 | R/W | 1 | 0 | pause | 0 => normal mode.<br>1=>Input Rdy will be deasserted after end of current data set. |
| 8 | R/W | 7:0 | 0 | len | len register |
| | | | | | |

## Verification plan

Logistics:

- Machine: 1 PC, Linux
- Repository: https://github.com/Dyumnin-Interns/interfaces-amrith-v#interfaces
- Regression: Github Actions

- Software : Python >3.10.6,  iverilog, cocotb, cocotb-bus, cocotb-coverage
- Licenses: Simulator License.
- BFM: None

## Environment

Unit Level DUT

Verification components: Drivers, Monitors, Scoreboard, Assertions, Generator

## Testcases

*TC1*

Feature: Accumulator test.

Description: Give known inputs to din pins, configuration cfg pins and len ports of the DUT and check whether the expected calculated value matches the accumulated value at the output of the DUT based on the given programmed length.

Scenario: Directed test

Given: Unit Test Environment

When: Input *din* is (1,2,9,5,3,10) and the *programmed length* is (5)
Then: Output is (20)

*TC2*

Feature: Accumulator test.

Description: Give known inputs to din pins, configuration cfg pins and len ports of the DUT and check whether the expected calculated value matches the accumulated value at the output of the DUT based on the given programmed length.

Scenario: Directed test – max unisgned values

Given: Unit Test Environment

When: Input *din* is (0,255) and the *programmed length* is (2)
Then: Output is (255)

*TC3*

Feature: Accumulator randomized test.

Description: Give random inputs to the din pins, configuration cfg pins and len ports of the DUT and check if the expected value matches the accumulated value at the output of the DUT based on the randomized programmed length.

Scenario: Random test

Given: Unit Test Environment

When: Input *din* is 8 samples randomized based on the randint function random.randint(0,31)

AND: Input *cfg* is 1 sample of random.randint(1,7)

AND: Input *len* is 1 sample of random.randint(1,7)

Then: Output is *expected_value.append(data_in[i])* where i is dependant on the length of either *len_data or cfg_data_in* based on *cfg_op* (read or write) and *cfg_data[0]* at address *0x4 (override).* The resultant list is then summed to get the expected value.

Coverage: Partial functional coverage with coverpoints *din* and range of bin values to check how many times each of these inputs have hit the corresponding bin values.

*TC4*

Feature: Accumulator randomized test with packetGenerator.

Description: Give random inputs to the din pins, configuration cfg pins and len ports of the DUT and check if the expected value matches the accumulated value at the output of the DUT based on the randomized programmed length, generated by the packetGenerator class.

Scenario: Random test

Given: Unit Test Environment

When: Input *din* is 15 samples in length where each of the input ranges from 0 to 20

AND: Input *cfg* is 1 sample of random.randint(1,15)

AND: Input *len* is 1 sample of random.randint(1,15)

Then: Output is *expected_value.append(data_in[i])* where i is dependant on the length of either *len_data or cfg_data_in* based on *cfg_op* (read or write) and *cfg_data[0] (override bit)* at address *0x4 (override).* The resultant list is then summed to get the expected value.

Coverage: Functional coverage with coverpoints *din and* range of 8 bit bin values to check how many times each of these inputs have hit the corresponding bin values. Added coverpoints for *dout* and checked the coverage of *RDY-EN* protocol.

## Schedule:

| Resource | Testcase | Enviroment | Start Date | End Date | Status |
|---|---|---|---|---|---|
| | | | | | |
| Amrith | TC1 | Unit level | 03-02-2023 | 07-02-2023 | Done |
| Amrith | TC2 | Unit level | 07-02-2023 | 07-02-2023 | Done |
| Amrith | TC3 | Unit level | 07-02-2023 | 10-02-2023 | Done |
| Amrith | TC4 | Unit level | 11-02-2023 | 25-02-2023 | Done |
| | | | | | |

## Datapath:

Bins

d_in: [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]

d_out: entire range of (0,255)

## Protocol:

| Data | EN(Enable) | RDY(Ready) | Description |
|---|---|---|---|
| Dont care | 0 | 0 | IDLE |
| Valid | 0 | 1 | RDY is high. |
| Valid | 1 | 1 | Transaction.<br>EN is asserted. |
| Invalid state | 1 | 0 | EN cannot go high without RDY being high. |

From the IDLE state, the DUT can go to either RDY state in case of din_en delayed or it can go directly to TXN state in the case of din_en going high without any delays after din_rdy has gone high. Only one of these 2 transitions are possible in coverage.

The DUT  jumps to IDLE state from TXN state after the accumulation has ended with busy low and the pause signal already high.

## Functional Coverage

The coverage of samples d_in of various lengths ‘N’  in directed tests is 100%

The coverage of 20 samples input data d_in when randomized between 0-20 is around 60%.

For the output d_out, the coverage is  always 1 out of possible 255 values. This is due to the nature of the DUT where for any N input data samples, there can only be one 8 bit output.

The coverage of the RDY-EN protocol could range from 6  out of 9 (66.67%) to 7 out of 9 (77.77%) possible states overall. This is due to different delays in different drivers each run of the simulation. The larger the input set and more varied the delays, the higher the coverage.

## Protocol: Bins

Previous State = [Idle, Ready, Transaction]

Current State = [Idle, Ready, Transaction]

## Delay

Min,Max = [0,8]