Overview

## Motivation

The Project is intended to be the main hands-on part of the training provided for the Design Verification course; it will allow the Fellows to properly implement the knowledge and skills they acquired up to that point and demonstrate creativity and professionalism. It was also a chance to learn how to manage time and risk in a Verification project and finally, gain valuable experience that will serve the Fellows upon interviewing for companies and kicking-off their careers.

In their final project, the Fellows will utilize their capabilities in Verilog, SystemVerilog and scripting in order to plan, implement and verify a new block with basic image processing capabilities.

The main goal of the project is to complete a full-cycle process - from architecture specification to 100% pass rate and 100% coverage hit verification sign-off.

## Goals

The main goal is to complete a full-cycle project from Arch specification to complete (100% pass rate and 100% coverage hit) verification sign-off. The fellows will do that by:

* Implementing various sub blocks using Verilog
* Planning a verification environment including coverage goals and a detailed verification plan
* Implementing the verification environment and debugging to 100% using SystemVerilog and Cadence platforms
* Creating scripts for automation processes using Perl and/or Python

## Project description:

In their final project the Fellows will be implementing a block which is dedicated for image processing purposes. The block will support two operations:

1. Threshold – Convert an image from RGB to binary by turning all pixels below some threshold value to zero and all pixels above that threshold to one.
2. Brighteness – Add a constant value to each pixel of the image.

The block receives requests from two different external sources, uses a fixed arbitration (slv0 is the highest) if needed and gives service to the granted source according to the request mode, relevant operation value and the data sent from the granted source.

The data is sent from the sources through the data bus (configurable width of 32/64 bits) to the block where every byte represents either a header value or an actual pixel R/G/B value. The block supports processing BMP with pixel format of 24-bit pixel (24bpp).

An operation starts by receiving all the image relevant data from the block, performing the necessary calculations and then outputting the results outside.

If the block is during processing operation then no other requests can be received and thus no grant/rdy should be generated during processing time.

## Infrastructure

The fellows will mostly use the Cadence Cloud Orchestrator platform and utilize tools such as Xcelium, vManager and more.

## Project format

The fellows will be divided to groups (2-3 persons in each group) and each group will be responsible to divide the different responsibilities to the different group members. The golden rule for this project is that you either the designer of a specific sub-block or you are in charge of the verification for this block. By this we will ensure a full understanding of the entire block and each fellow will understand better the relationship between a chip design engineer and a design verification engineer.

Evaluation and Mentorship

Through weekly status meetings with the program’s Tech Lead the Fellows will be measured according to the following categories:

* Full understanding of the module and its complexity
* Presenting a working verification environment (reference module, coverage definitions and implementation, etc.) according to a defined verification plan
* Reaching 100% pass rate (no Design bugs) and 100% coverage (all wanted inputs and scenarios were checked)
* Demonstrating good communication and team-work skills

# Professional checkpoints

Checkpoint #1: Submitting a project planning document which will include the responsibilities for each sub-block implementation, Verification plan and a list of required scripts. **Deadline:** TBD

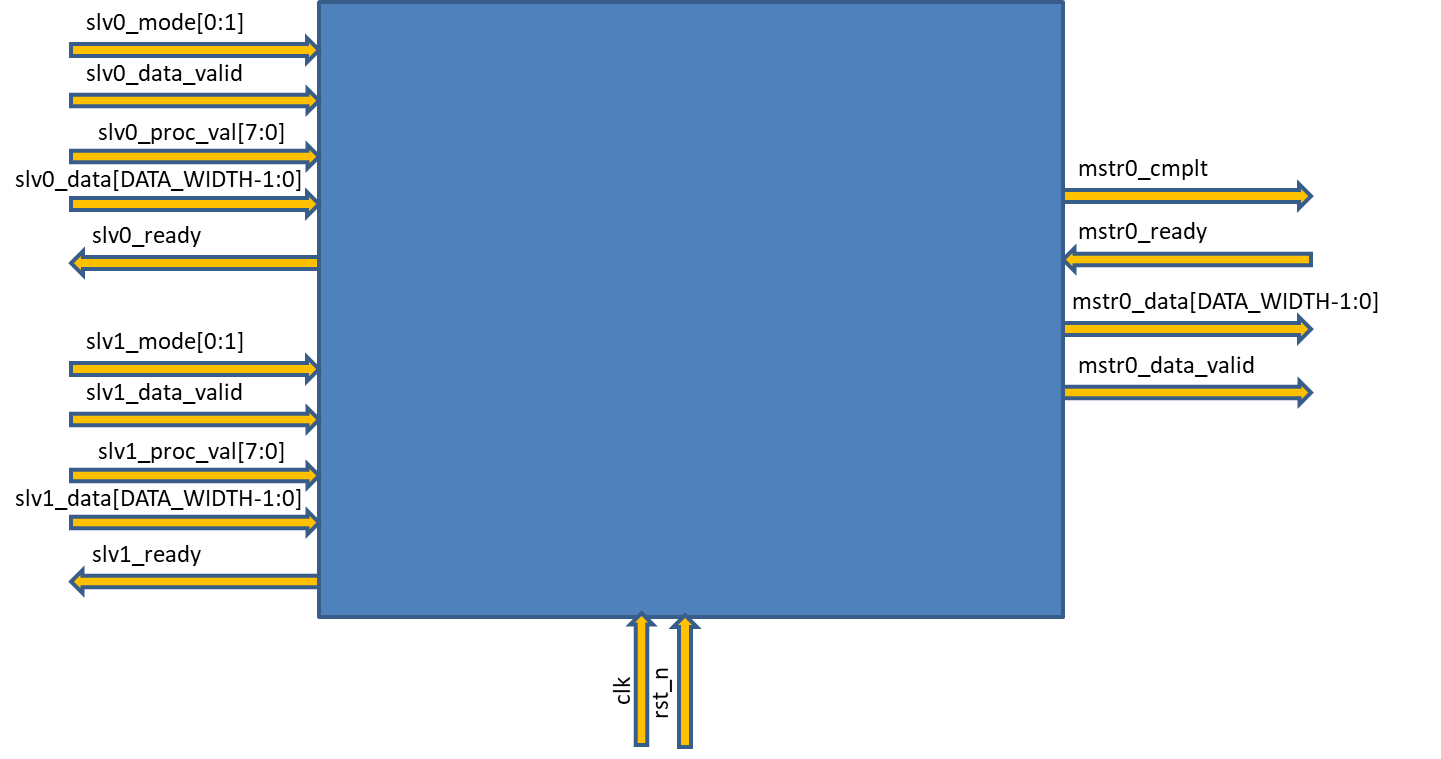
Checkpoint #2: Final implementation of all the sub-blocks using Verilog. **Deadline:** TBD

Checkpoint #3: Verification environment is ready for simulation. **Deadline:** TBD

Checkpoint #4: Achieving 100% pass rate and coverage. **Deadline:** TBD

Checkpoint #5: Submitting a project report (summary). **Deadline:** TBD

Block Diagram



Signal definitions

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Width** | **Direction** | **Description** |
| clk |  | Input | Clock signal |
| rst\_n |  | Input | reset signal (active low) |
|  |  |  |  |
| slvX\_mode | 2 bits | Input | 0 = no req, 1 = threshold operation, 2 = brightness operation |
| slvX\_data\_valid | 1 bit | Input | Indicates the data on the data bus is valid |
| slvX\_proc\_val | 8 bits | Input | The header/image data (header comes first and then row by row starting from the top left pixel) |
| slvX\_data | 32/64 bits | Input | Only relevant for brightness and threshold operations |
| slvX\_ready | bit | Output | Indicates req is granted or not |
|  |  |  |  |
| mstr0\_cmplt | 1 bit | Output | Indicates processing is completed |
| mstr0\_ready | 1 bit | Input | Source X is ready to accept the processed data |
| mstr0\_data | 32/64 bits | Output | The processed data |
| mstr0\_data\_valid | 1 bit | Output | Indicates the data on the data bus is valid |

Getting started: Instructions

* Load an array of images with multiple images before simulation. The values should be in HEX so each image should be converted from BMP to HEX (using Perl or Python or an online coverter). These images can be used by dedicated drivers to inject the data into the block.

Note that it’s using real images data is not mandatory. You can drive random values for simulation purposes.

* Every time there’s a valid processed data on the bus, the data should be captured and written into an array of images so the images can be automatically or manually checked once the simulation is complete.
* Use your knowledge of the images parameters (address in memory, height, width, etc.) to exercise you driver.
* Reduce the problem by planning and implementing sub-blocks. For example, it’s recommended to implement a module that its only purpose is interacting with the sources requests. Think and plan the modules and how they connect with each other and with the outside world.