Unit A: 1-D array image pixel processing.

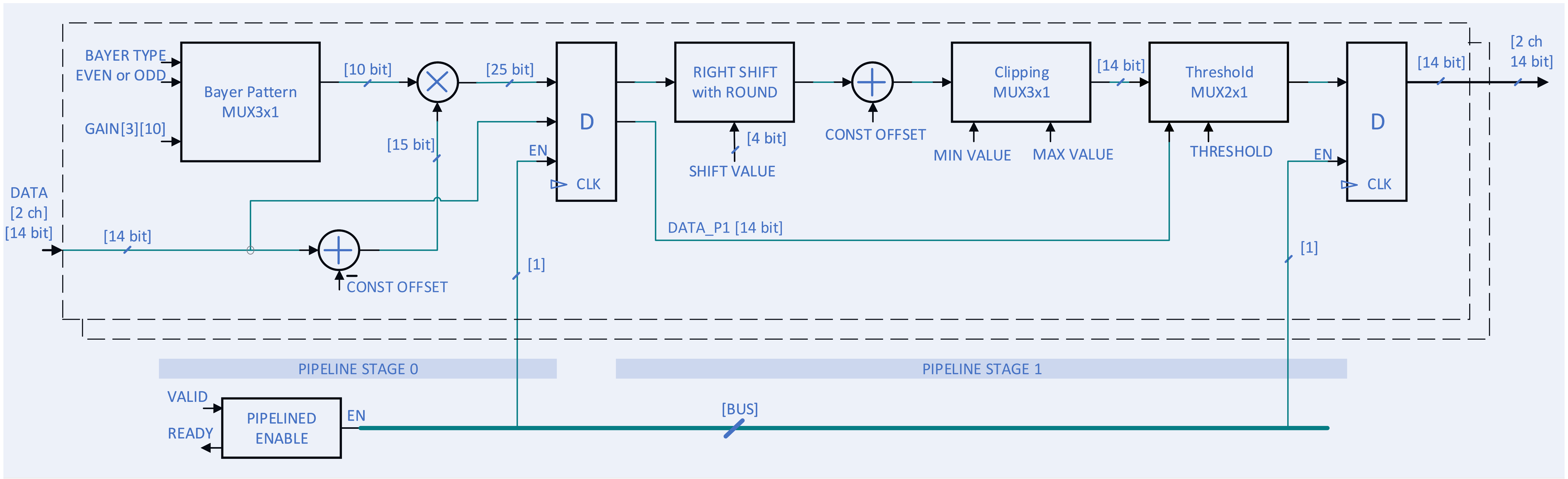
Implementation: RTL scheme (page 2) and Verilog code (link to github)

General testbench based on UVM framework with this module as DUT (link to github)

Unit B: Simple ALU to perform trivial math operations: ADD/SUB/MULT. 5 pipelines stage.

Implementation: Block diagram (page 3) and Verilog code (link to github)

SystemVerilog testbench for verification (link to github)



Unit function: Correction of image pixels.

Input data: 1-D array of pixels(14 bit unsigned). Two pixels are processed in parallel.
Output data: Corrected 1-D array of pixels(14 bit unsigned)

Correction blocks:
1. Add / Subtract const offset.
2. Multiply by input gain(10 bit unsigned).
3. Right shift by 4 bit value with rounding. Barrel shifter. Implemented as chain of four 2x1 multiplexers (more optimal that 16x1 multiplexer from area point of view).
4. Clip result. Saturate to min(0) / max(2^14 - 1) values.
5. Threshold 2x1 Multiplexer. Provide correction bypass when input is less than const threshold value.
6. Bayer pattern Multiplexer 3x1. Choose appropriate gain from three input ones (RGB) based on Bayer pattern type and line index (odd or even).

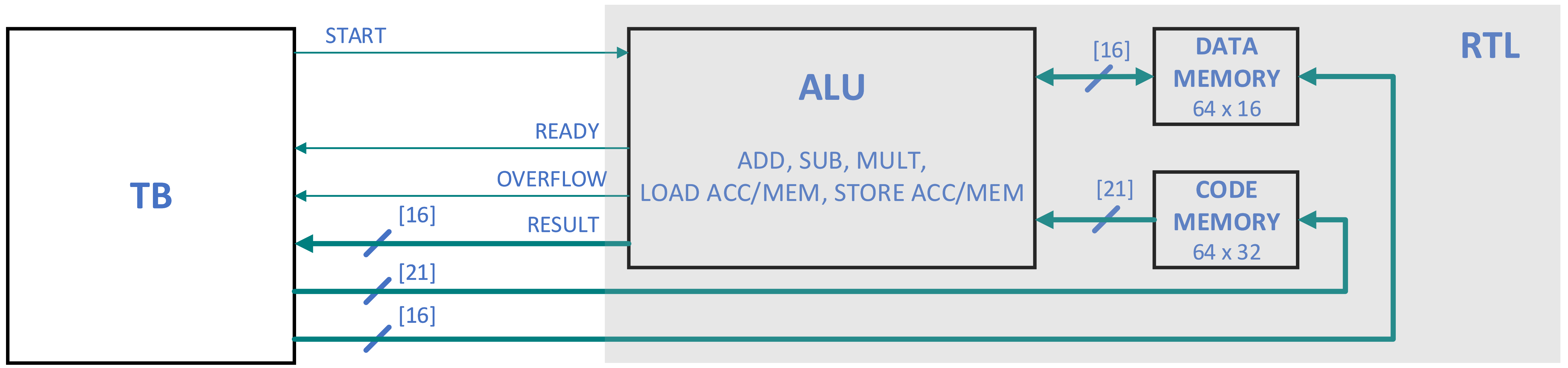
Implementation: 2-staged pipeline processing.
7. Pipelined Enable. Taking reduced AMBA bus handshake signals (VALID, READY) generate shifted Enable signal to support pipeline.

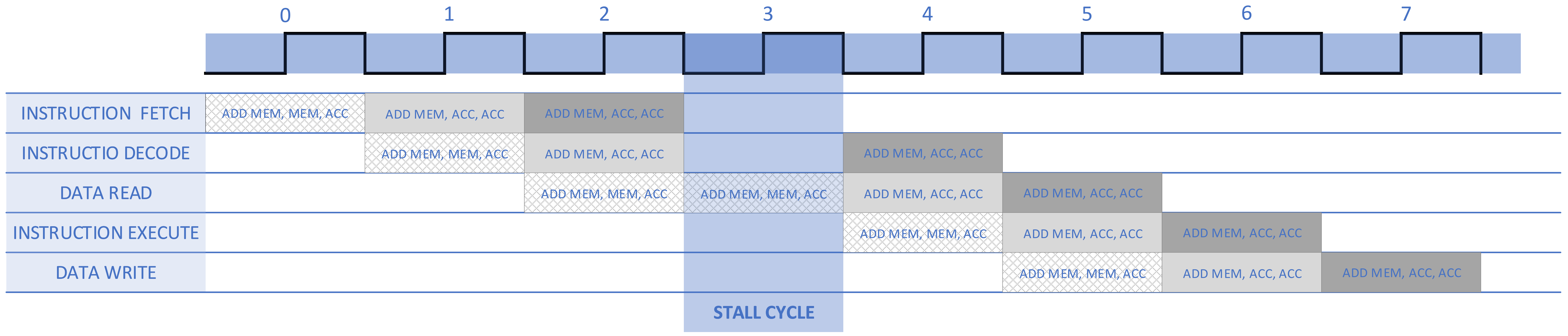
Sources:

1. Functional C-model (link to github)

2. Verilog implementation (link to github)

3. General SystemVerilog testbench based on UVM framework (link to github)





ALU Unit: Performs operations: Add, Subtract, Multiply on two 16 bit operands (accumulator or loaded from memory), stores 16 bit result to accumulator or memory.

Input data: Program. Set of asm instructions stored in Code memory (up to 64 instructions).
Output data: Result of program execution: value stored in accumulator and overflow flag (set when overflow takes place during program execution).

Implementation (link to github):
Code and Data memory – single port memory implemented on Flip flops. Support only one write or read during one cycle.
ALU has 5 pipeline stages: Fetch, Decode, Read data from memory, Execution, Store result to memory.
Three instruction tpes are supported: ADD, SUB, MULT. Every instruction takes 3 operands: 2 inputs and one output to store result. Every operand can be either accumulator or memory address.
In case of two memory access during single cycle, unit generates stall cycle providing ability to perform second memory access (read or write).
Thus ALU supports uninterruptible pipeline flow when “multiple memory access scenario” occurs.

TB (link to github):
SystemVerilog testbench. Reads program and data from input txt file, loads set of instructions to Code memory, initializes Data memory with preloaded data. Then initiates ALU to execute the program, checks execution result and overflow flag and prints results of testing to stdout.