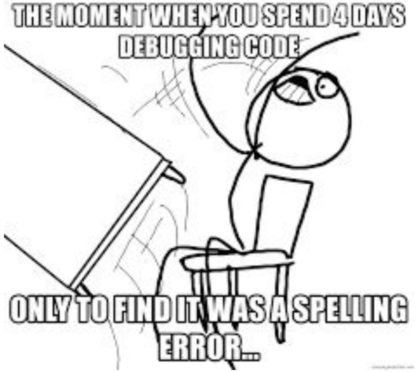
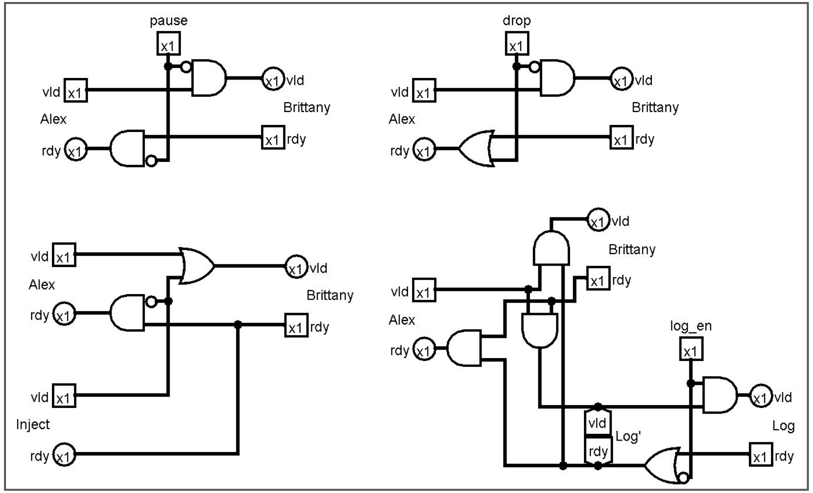
Debug Governor

with AXI-Lite

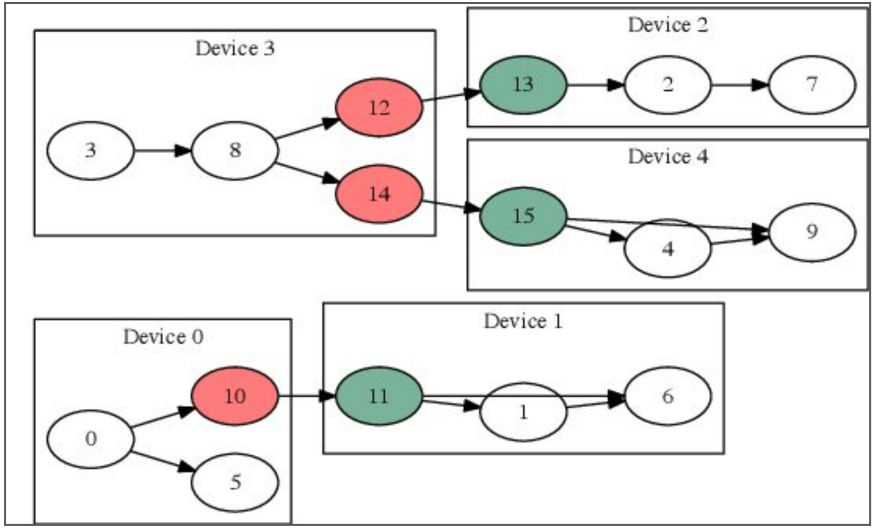
Introduction


THE MOMENT WHEN YOU SPEND 4 DAYS DEBUGGING CODE
ONLY TO FIND IT WAS A SPELLING ERROR...

Past Work: Debug Guvs with AXI-Stream

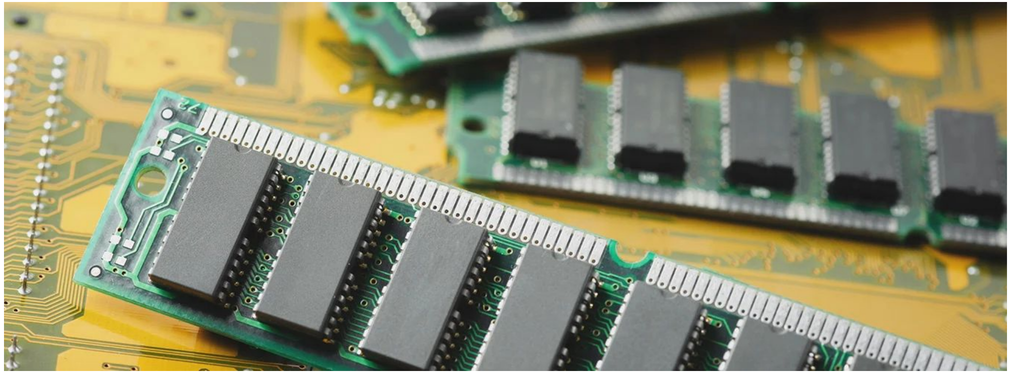


- **Pause**: Put a transaction on hold
- **Log**: Log data from transaction
- **Drop**: Cancel a transaction and move on
- **Inject**: Forcibly add new data on a stream



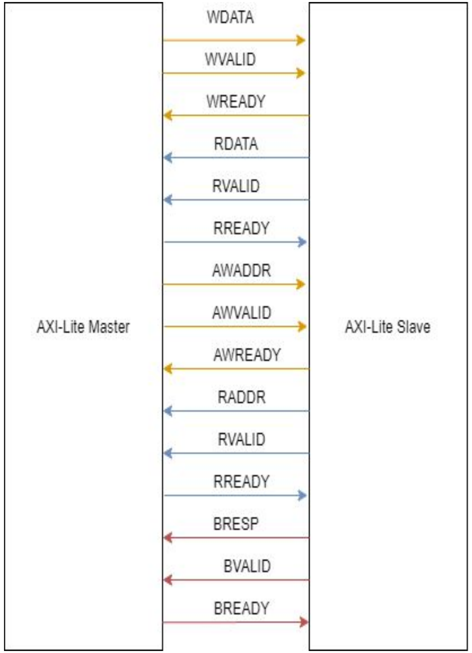
Motivation: Debug Guvs with AXI-Lite

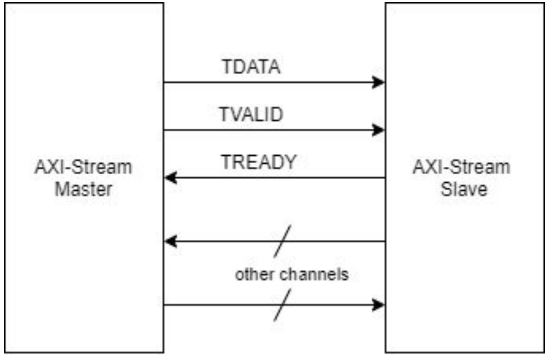
- AXI-Stream does not support addressing



- Adding 5 AXI-Stream Debug Guvs on AXI-Lite does not work!

AXI-Lite vs AXI-Stream





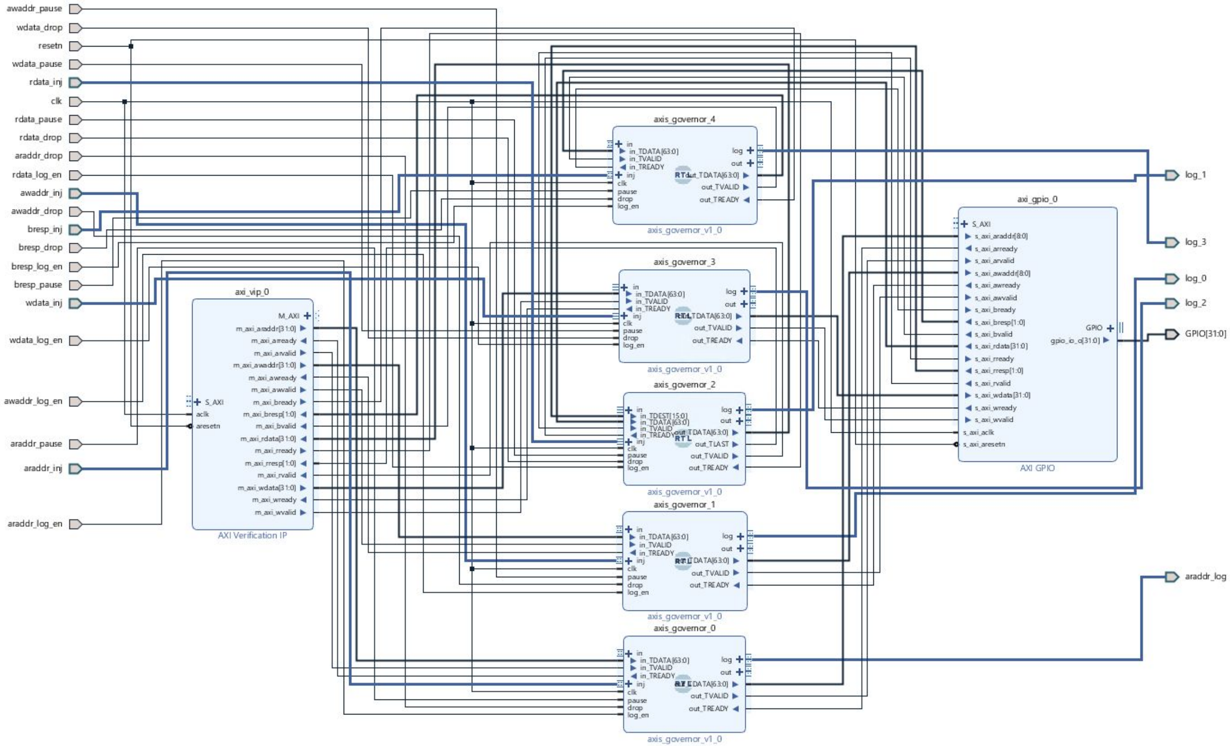
In order to transition from AXI-Stream to AXI-Lite, the debug governor must consider corner cases in implementing injecting to WDATA/RDATA:

1. For each WDATA and RDATA being, there must be AWADDR and ARADDR

2. For injecting into WDATA, BRESP must immediately be dropped with the drop command, so it never reaches the master. Otherwise this will violate the protocol.

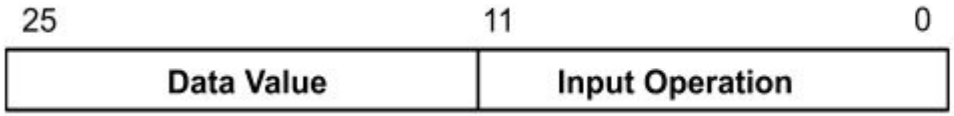
3. For injecting into RDATA, one must consider supplying ARADDR, and dropping ARADDR before it reaches the slave.

Implementation



In theory it is possible to use 5 debug guvs, but it is practically impossible to work with!

New interface



25

11

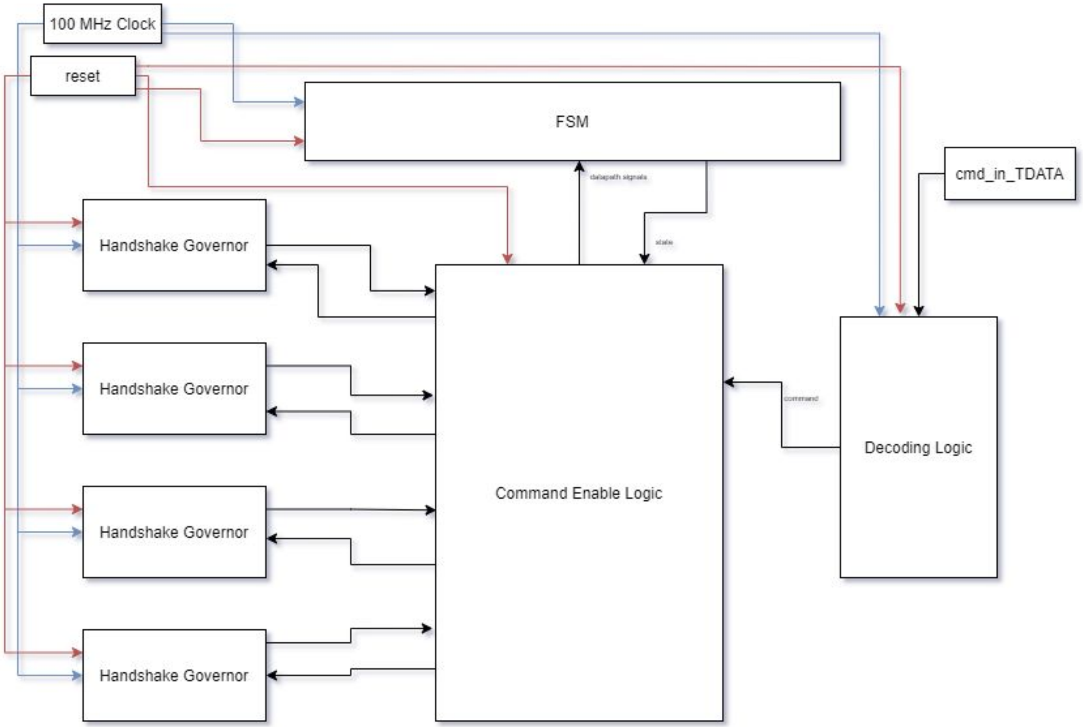
0

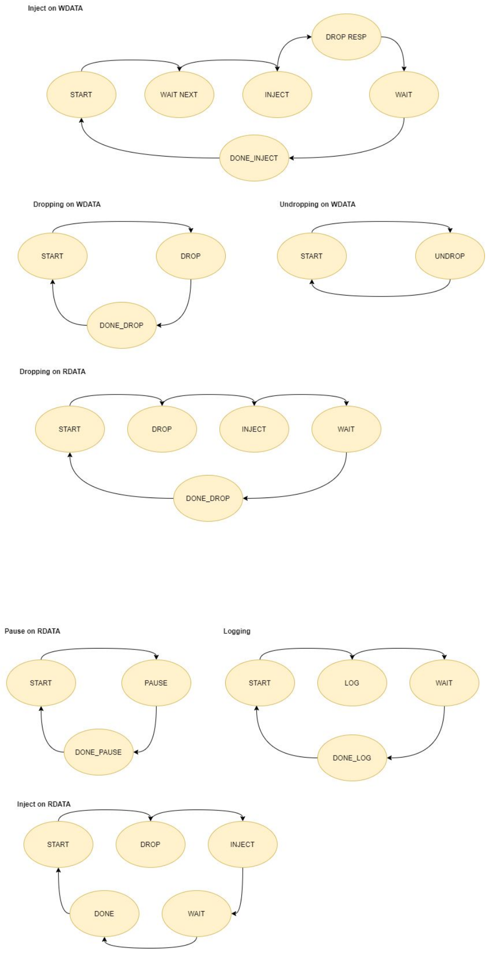
Data Value

Input Operation

- The top level module takes in a command input that is structured as an operation code (OP code).
- Part of the OP code consists of an encoded command on a certain stream, and the remaining bits correspond to data for write data.

Top Level Design




Inject on WDATA

Dropping on WDATA

Undropping on WDATA

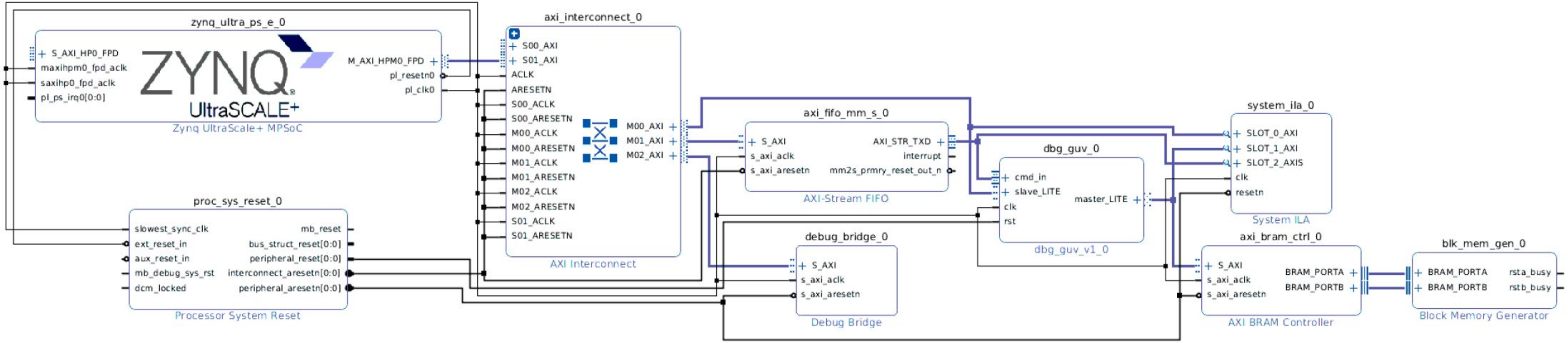
Dropping on RDATA

Pause on RDATA

Logging

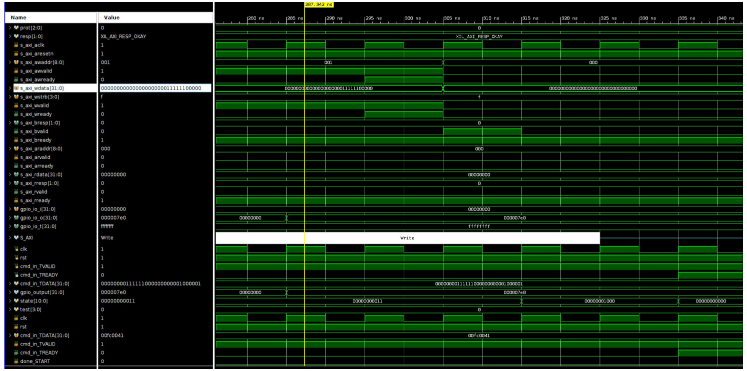
Inject on RDATA

Block Design



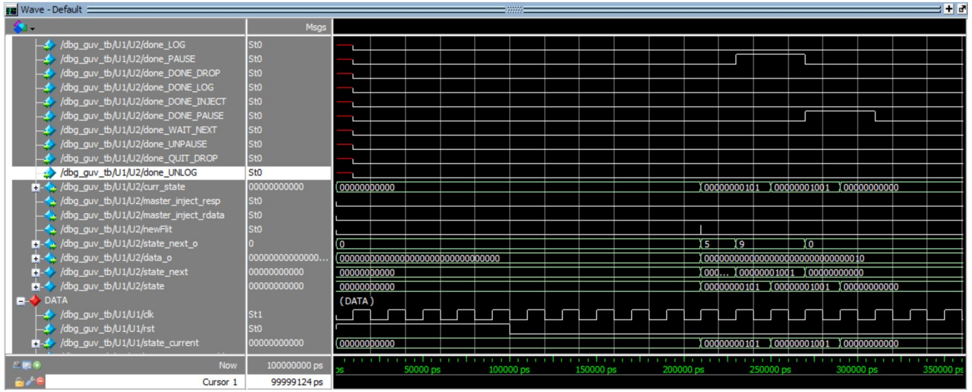
Report of success

Testbenched on Vivado using AXI-VIP and AXI-GPIO



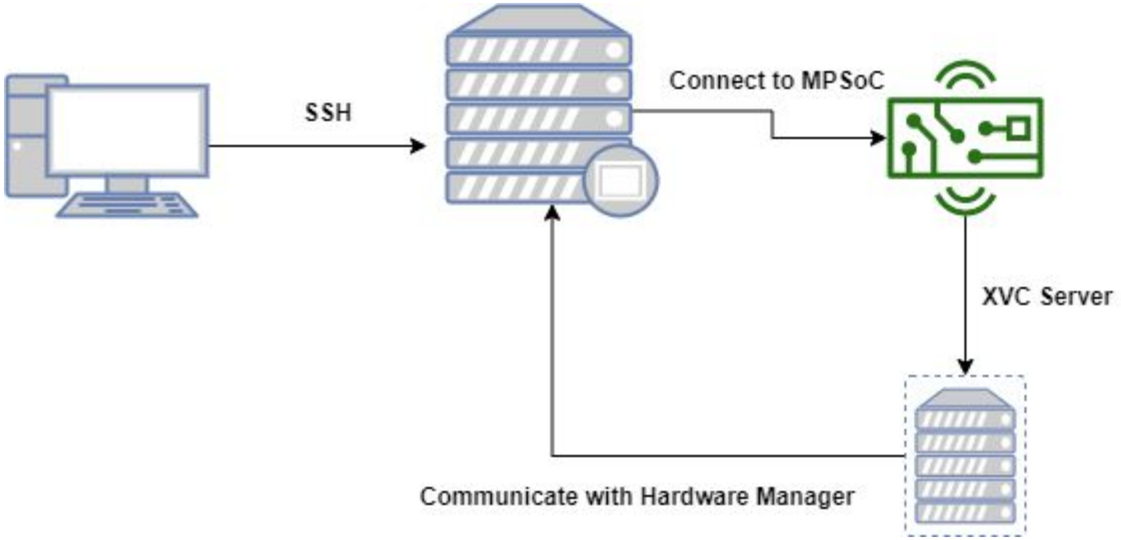
Name	Value
prot[2:0]	0
resp[1:0]	XIL_AXI_RESP_OKAY
s_axi_aclk	1
s_axi_aresetn	1
s_axi_awaddr[8:0]	001
s_axi_awvalid	1
s_axi_awready	0
s_axi_wdata[31:0]	00000000000000000000011111100000
s_axi_wstrb[3:0]	f
s_axi_wvalid	1
s_axi_wready	0
s_axi_bresp[1:0]	0
s_axi_bvalid	0
s_axi_bready	1
s_axi_araddr[8:0]	000
s_axi_arvalid	0
s_axi_arready	0
s_axi_rdata[31:0]	00000000
s_axi_rresp[1:0]	0
s_axi_rvalid	0
s_axi_rready	1
gpio_io_i[31:0]	00000000
gpio_io_o[31:0]	000007e0
gpio_io_t[31:0]	ffffffff
S_AXI	Write
clk	1
rst	1
cmd_in_TVALID	1
cmd_in_TREADY	0
cmd_in_TDATA[31:0]	00000000111111000000000001000001
gpio_output[31:0]	000007e0
state[10:0]	00000000011
test[3:0]	0
clk	1
rst	1
cmd_in_TDATA[31:0]	00fc0041
cmd_in_TVALID	1
cmd_in_TREADY	0
done_START	0

Testbenched on ModelSim for faster tests on functional FSM



	Msgs
/dbg_guv_tb/U1/U2/done_LOG	St0
/dbg_guv_tb/U1/U2/done_PAUSE	St0
/dbg_guv_tb/U1/U2/done_DONE_DROP	St0
/dbg_guv_tb/U1/U2/done_DONE_LOG	St0
/dbg_guv_tb/U1/U2/done_DONE_INJECT	St0
/dbg_guv_tb/U1/U2/done_DONE_PAUSE	St0
/dbg_guv_tb/U1/U2/done_WAIT_NEXT	St0
/dbg_guv_tb/U1/U2/done_UNPAUSE	St0
/dbg_guv_tb/U1/U2/done_QUIT_DROP	St0
/dbg_guv_tb/U1/U2/done_UNLOG	St0
/dbg_guv_tb/U1/U2/curr_state	00000000000
/dbg_guv_tb/U1/U2/master_inject_resp	St0
/dbg_guv_tb/U1/U2/master_inject_rdata	St0
/dbg_guv_tb/U1/U2/newFlit	St0
/dbg_guv_tb/U1/U2/state_next_o	0
/dbg_guv_tb/U1/U2/data_o	00000000000000...
/dbg_guv_tb/U1/U2/state_next	00000000000
/dbg_guv_tb/U1/U2/state	00000000000
DATA	
/dbg_guv_tb/U1/U1/clk	St1
/dbg_guv_tb/U1/U1/rst	St0
/dbg_guv_tb/U1/U1/state_current	00000000000
Now	100000000 ps
Cursor 1	99999124 ps

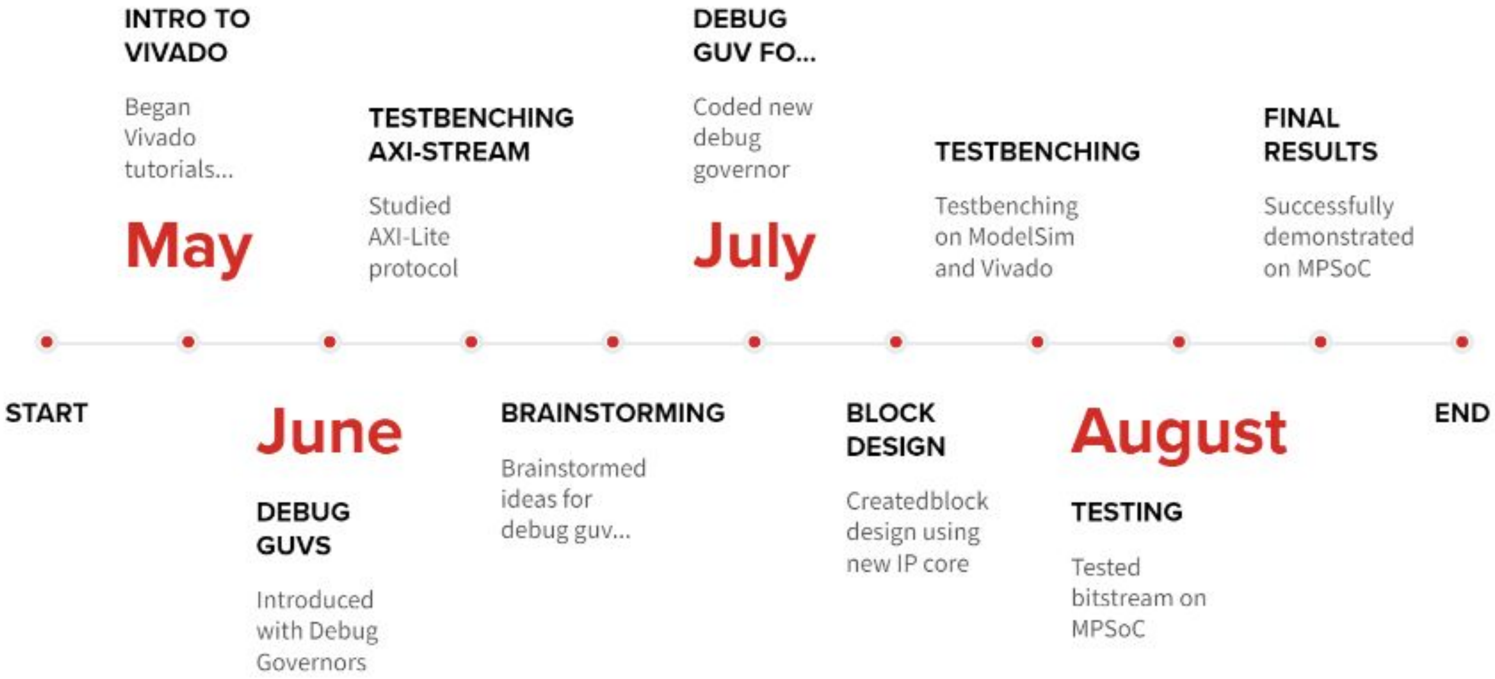
Test setup



- On capstone container, generate bitstream and .bin file
- On MPSoC17, program the FPGA and set up XVC server to connect with hardware manager
- Use Poke and DPoke to send flits to AXI-FIFO

Demo

Progress Timeline



START

May

INTRO TO VIVADO

Began Vivado tutorials...

June

DEBUG GUVS

Introduced with Debug Governors

TESTBENCHING AXI-STREAM

Studied AXI-Lite protocol

BRAINSTORMING

Brainstormed ideas for debug guv...

July

DEBUG GUV FO...

Coded new debug governor

BLOCK DESIGN

Createdblock design using new IP core

TESTBENCHING

Testbenching on ModelSim and Vivado

August

TESTING

Tested bitstream on MPSoC

FINAL RESULTS

Successfully demonstrated on MPSoC

END

Conclusion

- Debug Governor presents a novel backend framework for AXI-Lite systems i.e. allowing for memory-mapped transactions

- The design may be optimized via pipelining techniques for more efficient performance, and further state optimizations may be made to reduce code complexity

- A developer may use this code to develop more practical functions that resemble software debugging tools

- Design needs improvement on ease of use. Since users with no background on Debug Governors should be able to use the tool, an intuitive user interface should be designed to interact between the FPGA and Debug Governor.