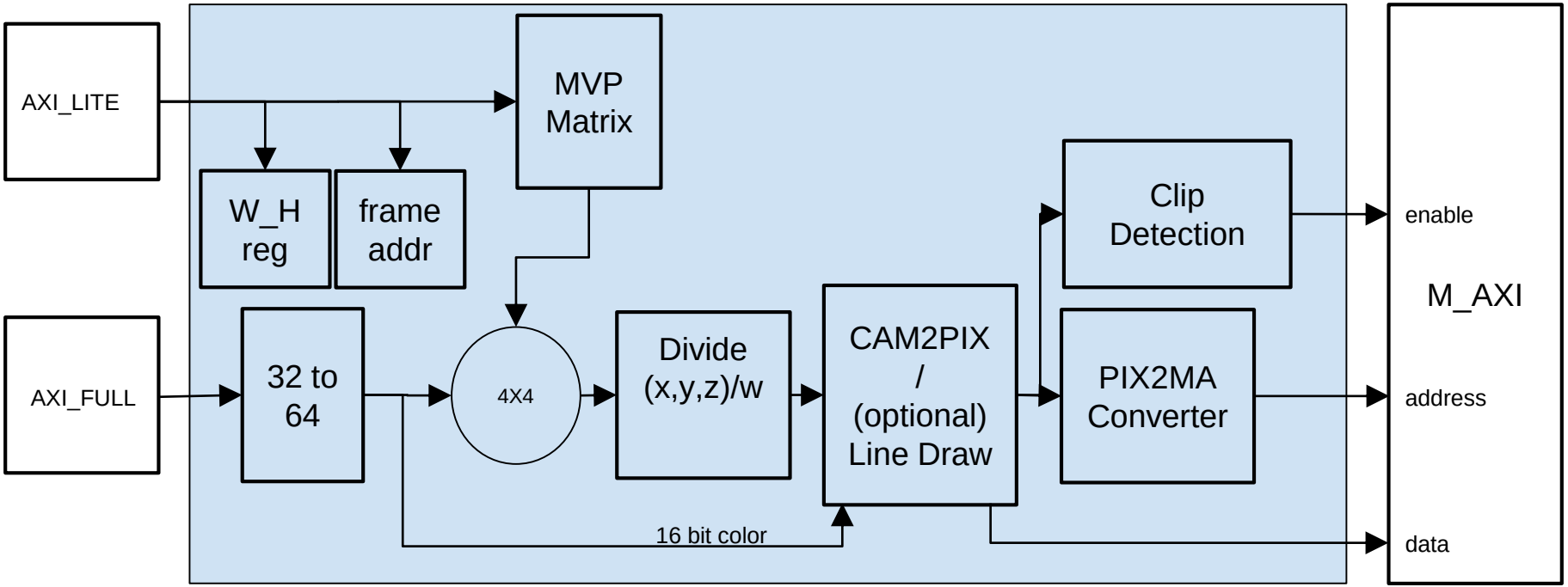
Simple GPU

Anthony Bogedin
Michael Lohrer

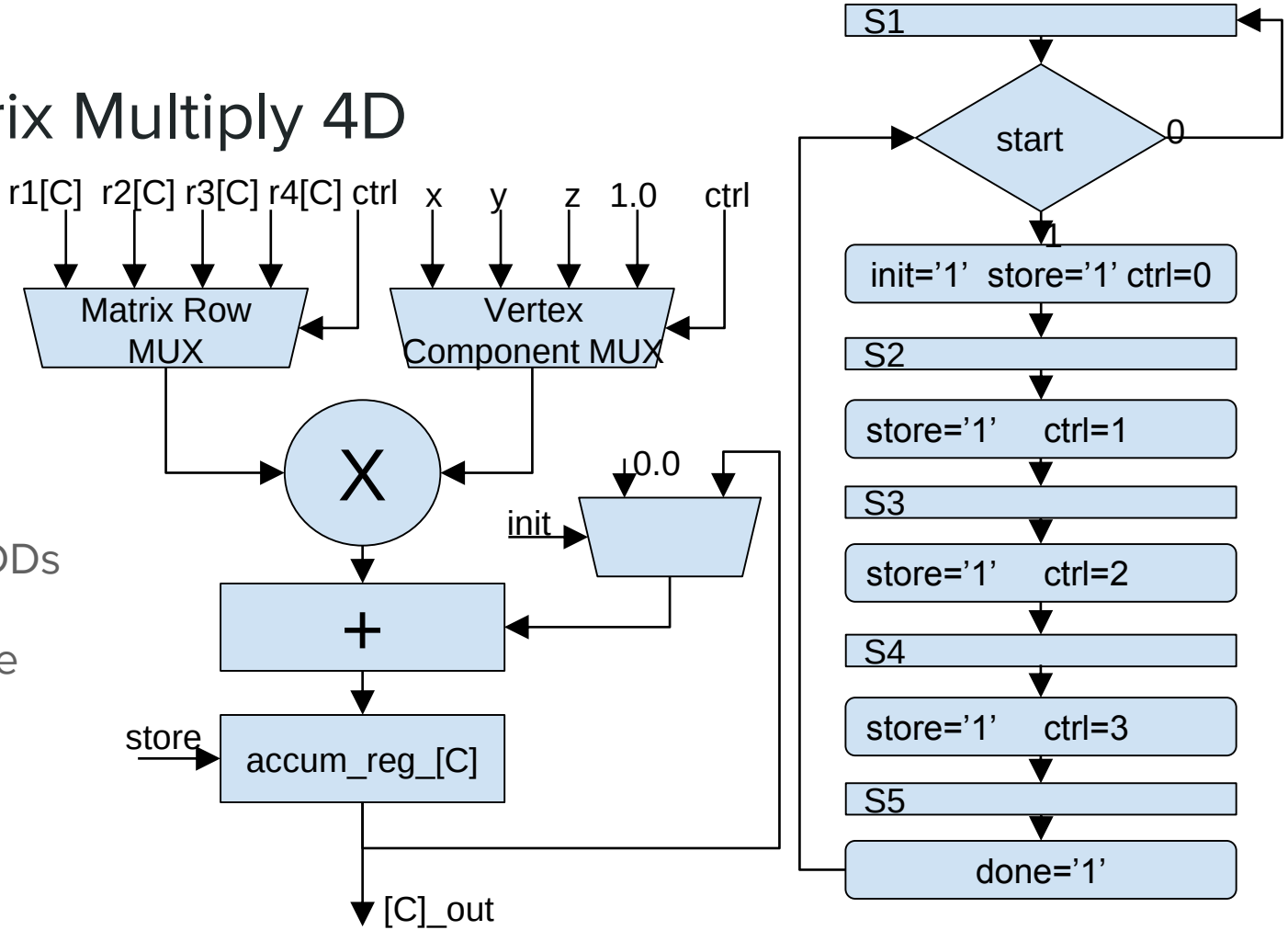
GPU Architecture



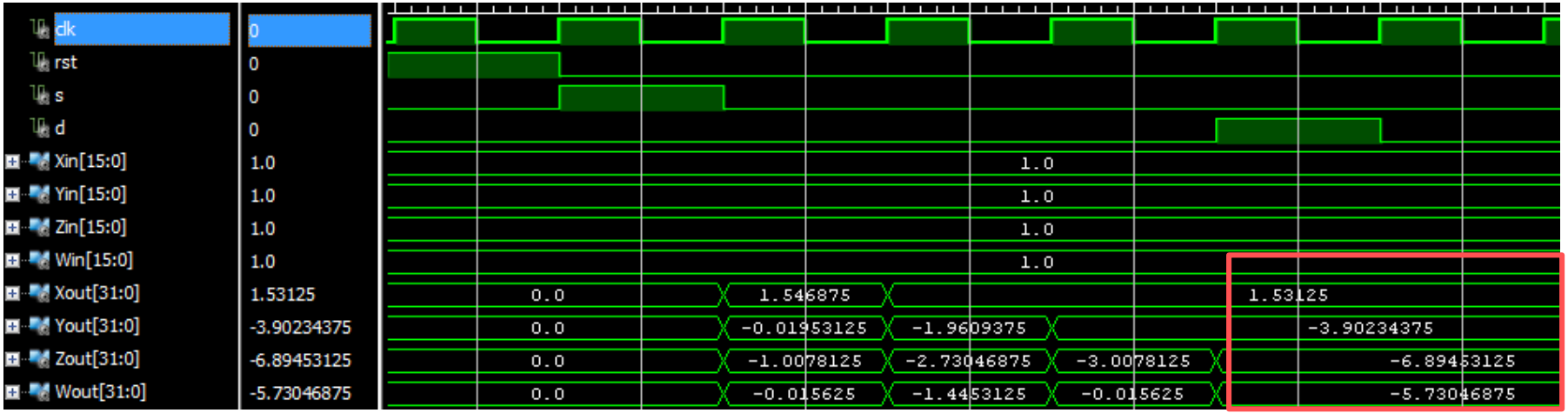
Vector Matrix Multiply 4D

Contains 4 MADDs

C represents the column



Matrix Multiply Simulation



Matrix Under Test:

1.5469	-0.0195	-1.0078	-0.0156
-0.0156	-1.9414	-1.7227	-1.4297
0.0	-1.9414	-0.2773	1.4297
0.0	0.0	-3.8867	-5.7148

Matlab Output Comparison:

	X	Y	Z	W
Vertex:	1.5313	-3.9023	-6.8945	-5.7305

Divider

Used the pipelined divider provided by Llamoca, but placed a wrapper around it to handle negative numbers and overflow

To get a fractional output, W was truncated by 12 bits ([32 16]/[20 4] = [32 12])

The X, Y, & Z components of the vertex need to be normalized before being mapped to screen space

Dividing all three by W ensures that all values are between -1 and 1 if they are on screen

Since the matrix multiply takes 4 clock cycles, X, Y, & Z are feed into the divider one at a time

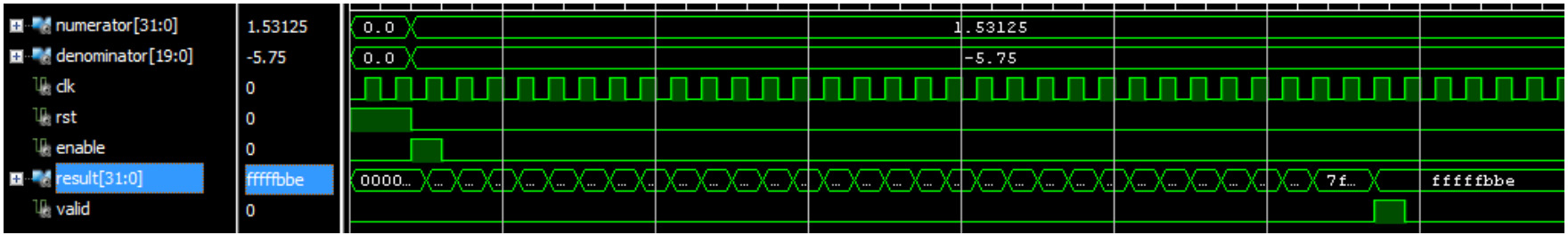
After division, they are stored in 3 registers to await processing in the next stage

Divider Tests

Example value with : 1.53125 [32 16] / -5.6875 [20 4] = -0.26904296875 [32 12]

= FFFFFBB2 [32 12]

Result from divider: -0.26611328125 [32 12] = FFFFFBBE [32 12]



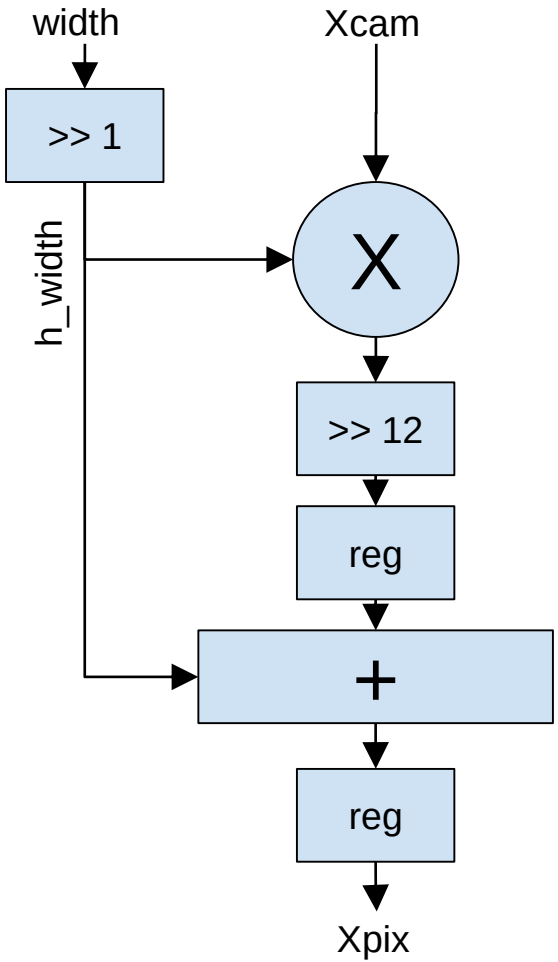
Camera to Screen Space

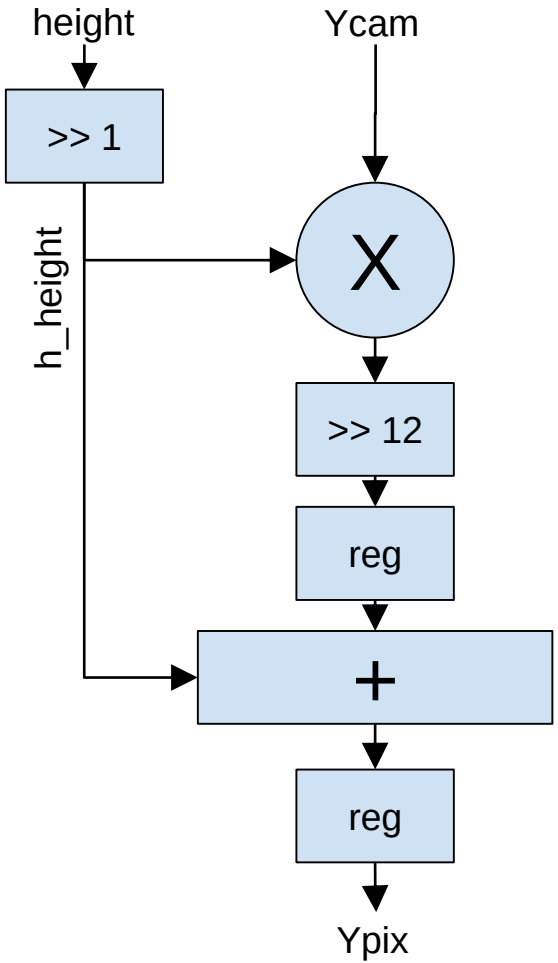
Since Xcam and Ycam are from -1 to 1

Just multiply by half of width and height

Truncate fractional bits

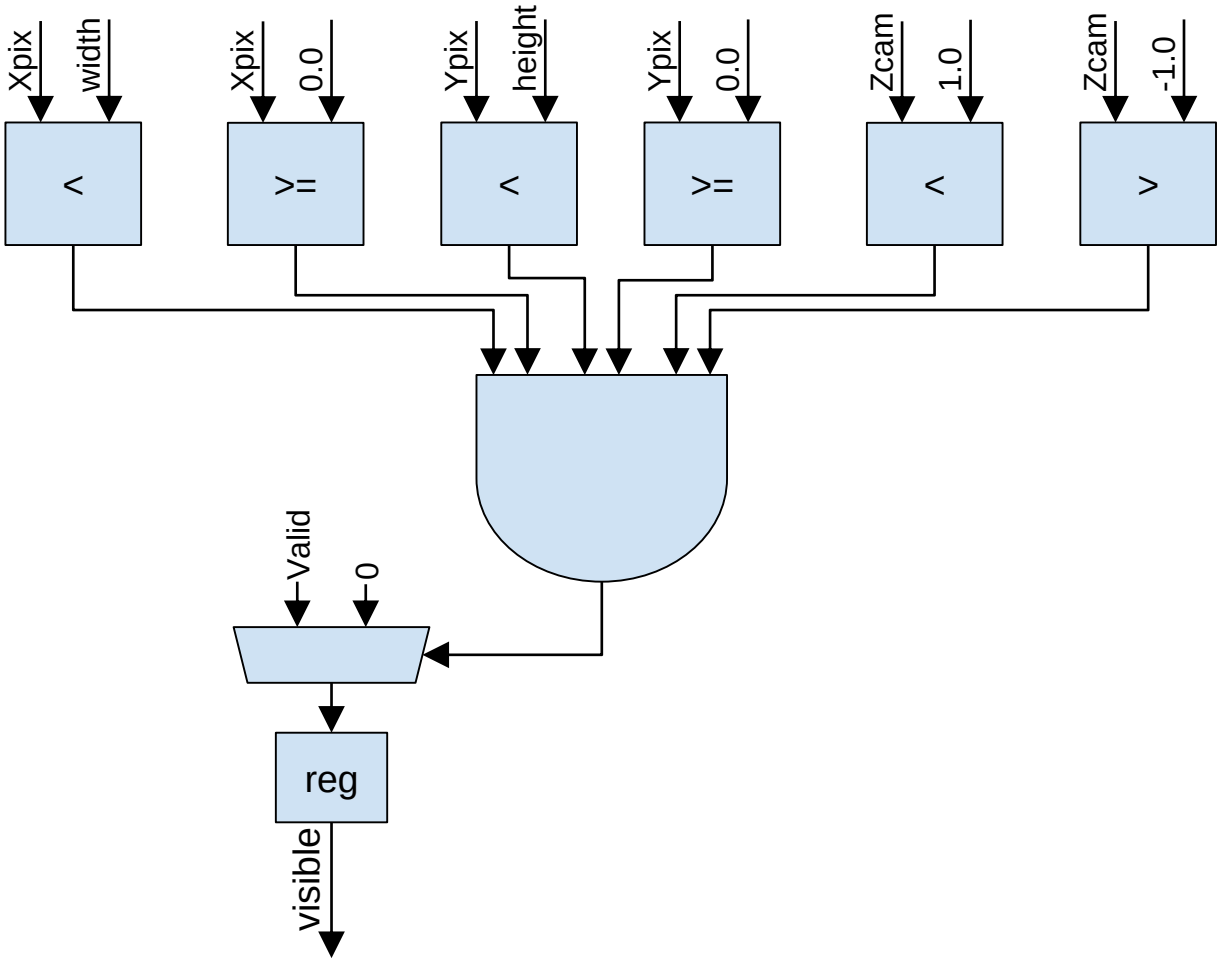
Add half of width and height





Clip Detection

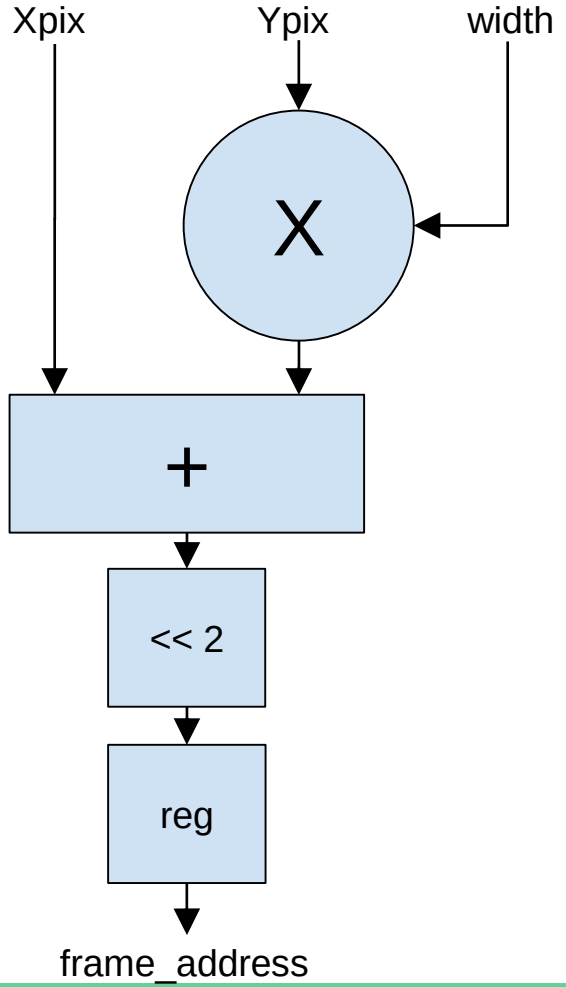
If the input X and Y coordinates are within the screen dimensions and there is a valid vertex, set visible high



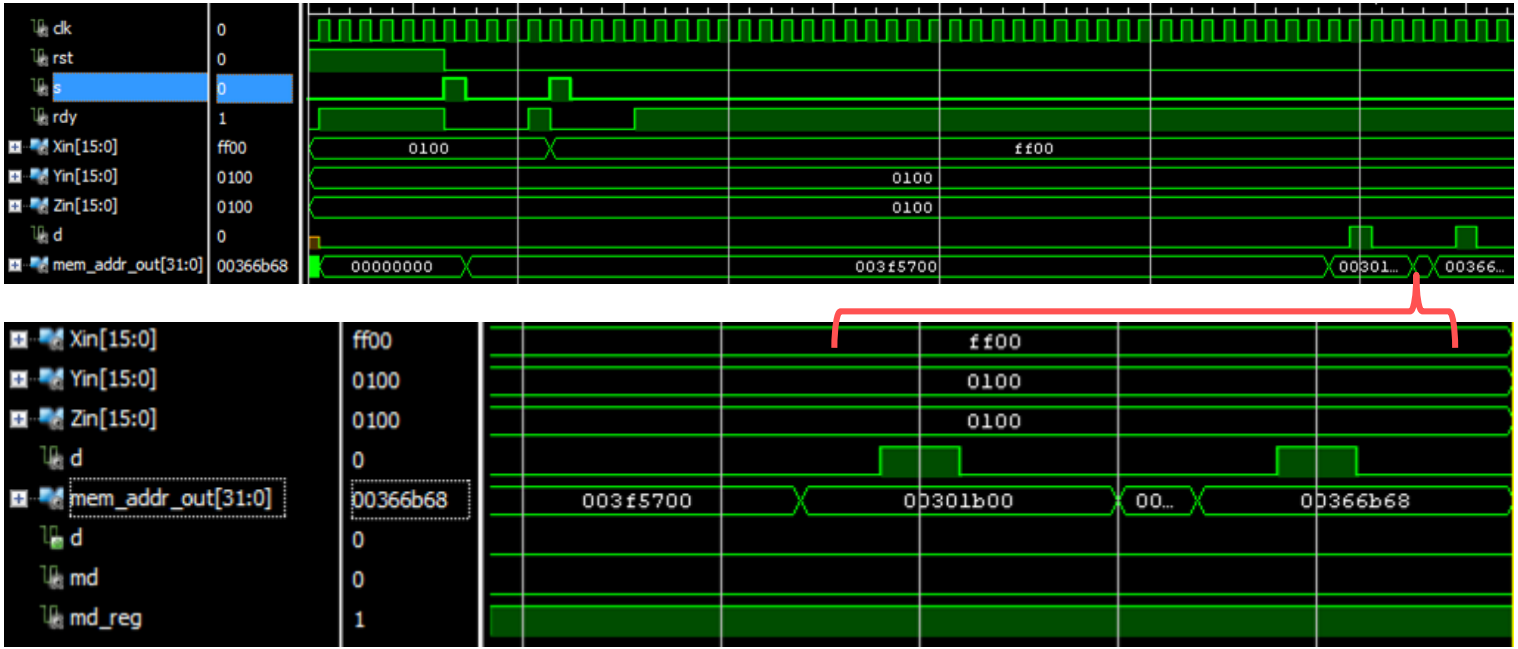
Screen to Array Index

frame_address = Xpix + width * Ypix

- Multiply Ypix by width to jump to the correct row
- Add Xpix to jump to the correct column
- Add 2 zeros for AXI word alignment



SGPU Simulation



Integer Line Drawing

Bresenham's line algorithm:

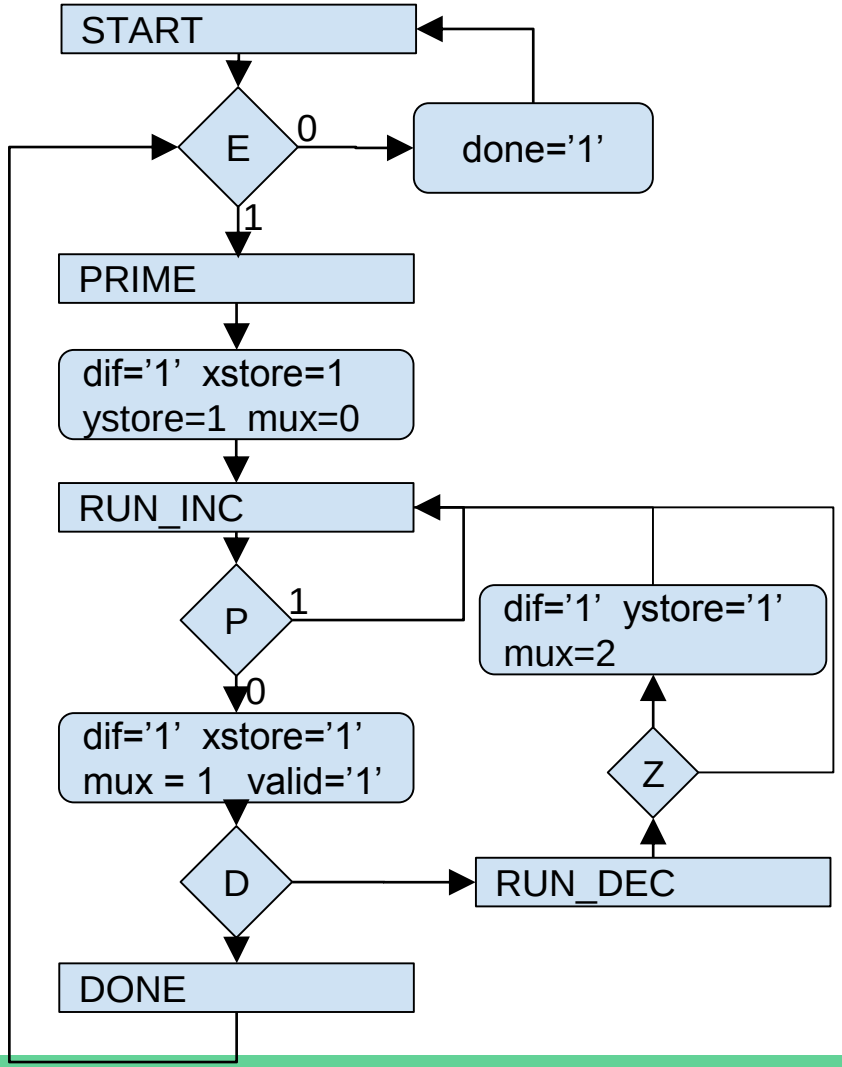
```
plotLine(x0,y0, x1,y1)
  dx=x1-x0
  dy=y1-y0

  DIF = 4*dy - dx
  y=y0

  for x from x0 to x1
    plot(x,y)
    DIF = DIF + (2*dy)
    if DIF > 0
      y = y+1
      DIF = DIF - (2*dx)
```

Line Drawing State Machine

Using two coordinates in Screen Space, trace line between them

Takes two clock cycles

This can saturate the AXI bus

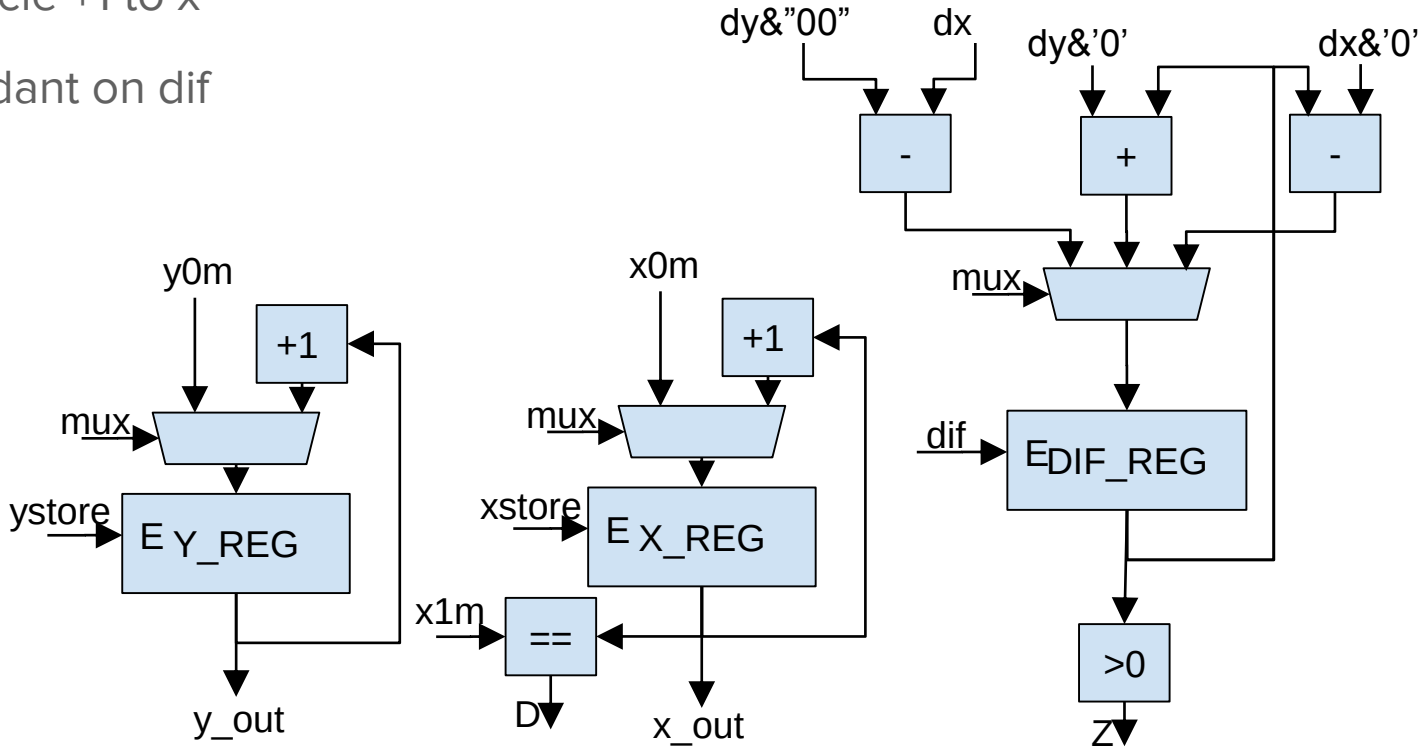


Line Drawing Data Path

Every pixel cycle +1 to x

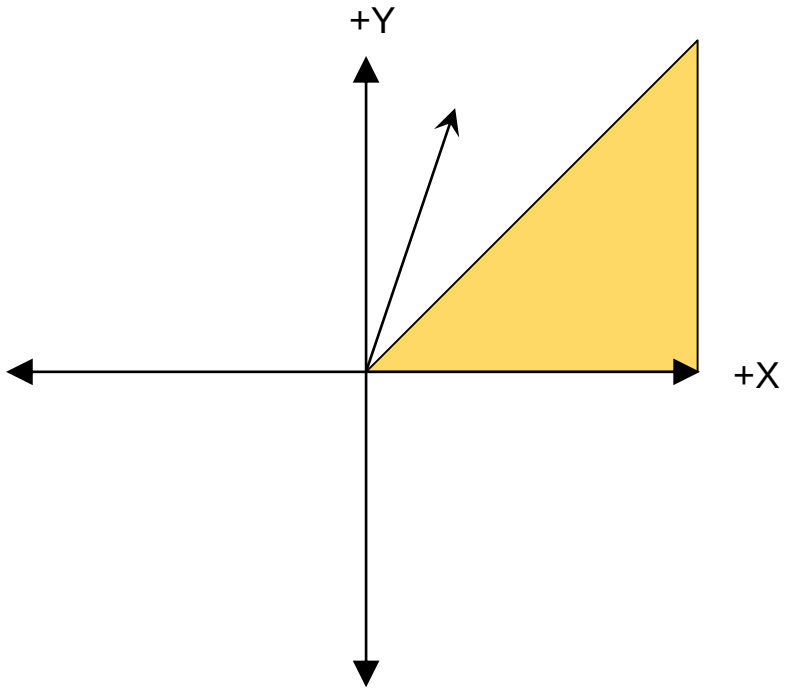
+1 to y dependant on dif

```
plotLine(x0,y0, x1,y1)
  dx=x1-x0
  dy=y1-y0

  DIF = 4*dy - dx
  y=y0

  for x from x0 to x1
    plot(x,y)
    DIF = DIF + (2*dy)
    if DIF > 0
      y = y+1
      DIF = DIF - (2*dx)
```



Limits of Simple Line Drawing

- Only increments
- Y increments at most once per X increment
- Only draws lines in the first 45° of the first quadrant

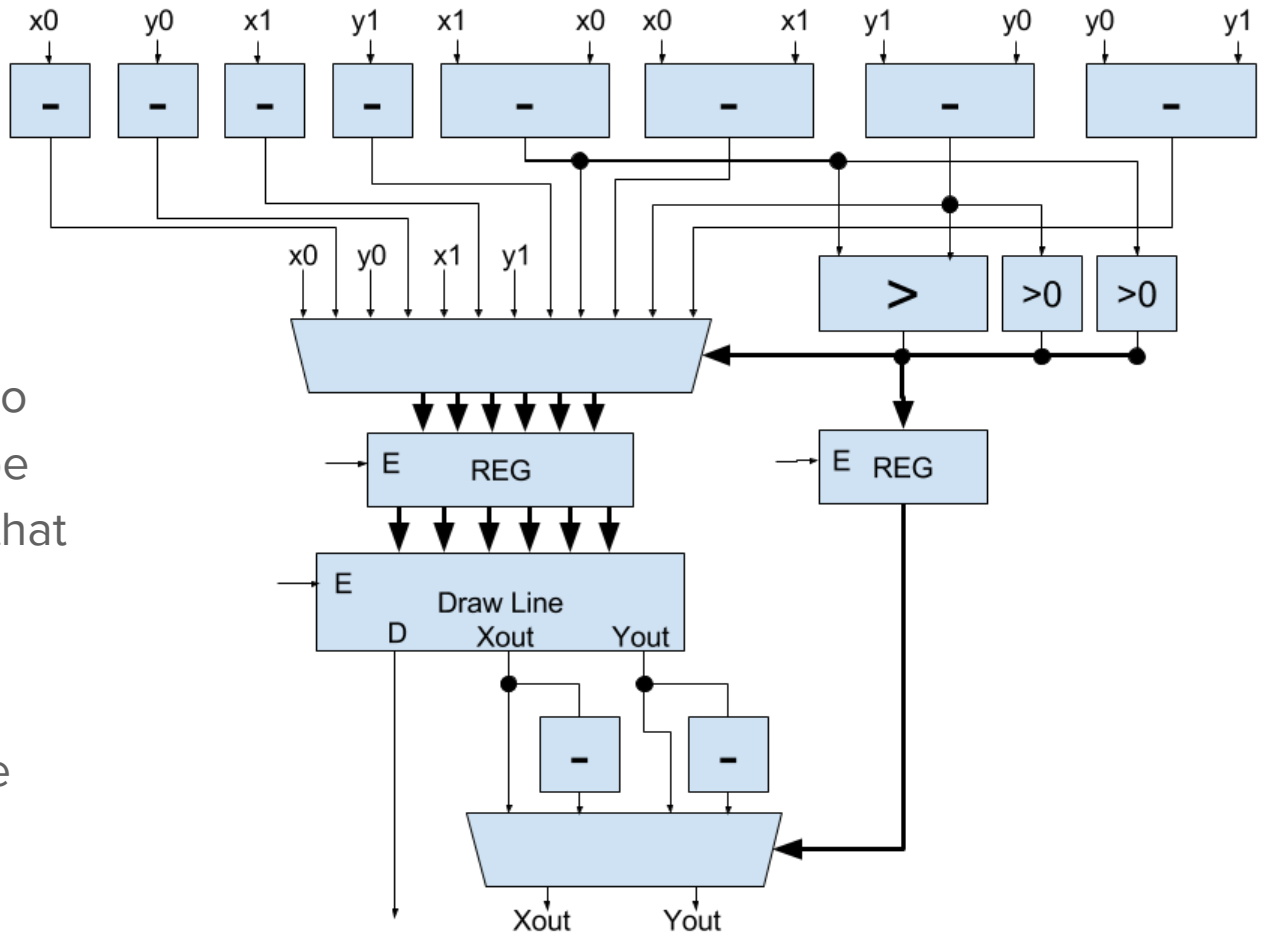


Folding Logic

Uses inputs and their negations

MUXs folds the inputs to draw line to always be positive and ensure that x is larger than y

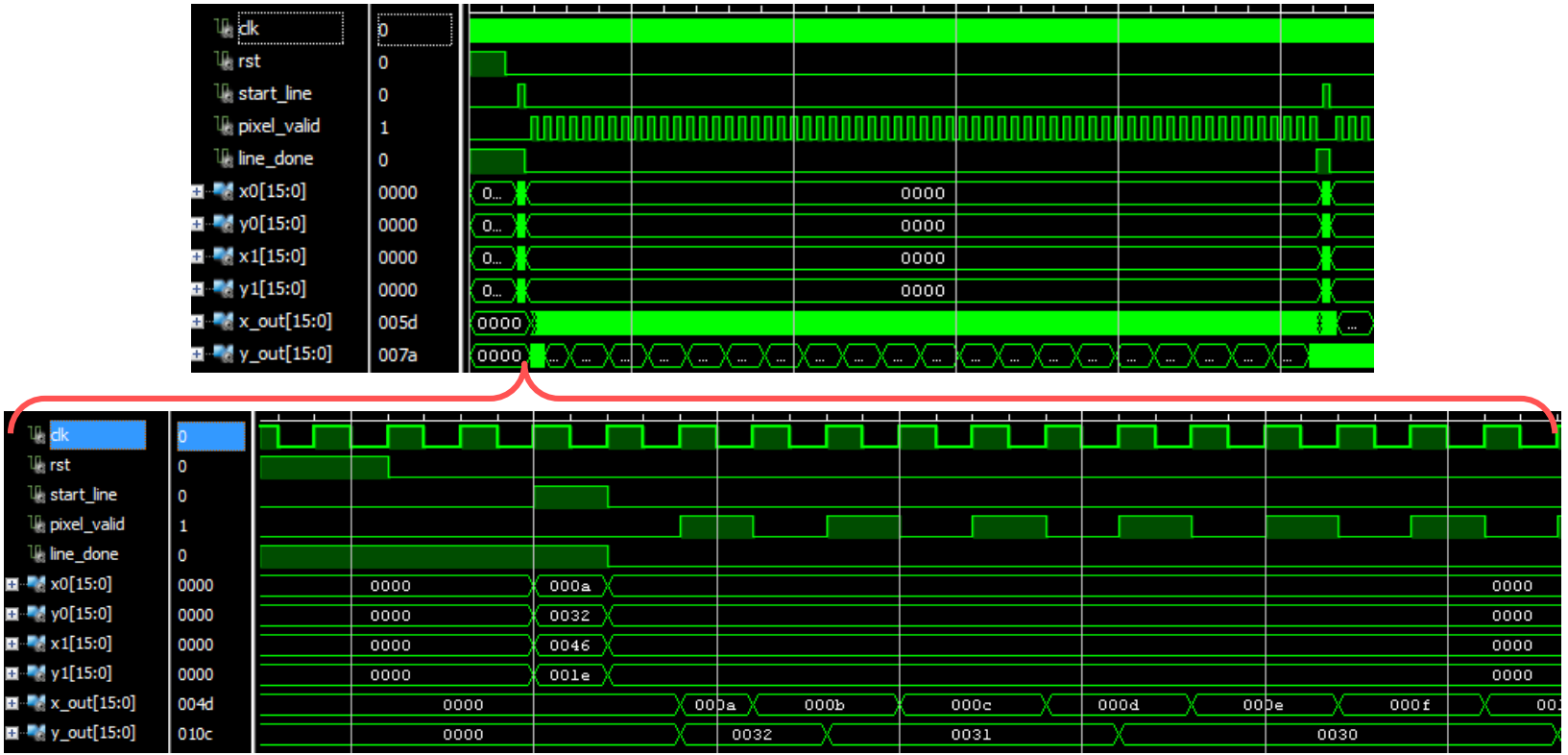
Final MUX folds values back into appropriate quadrant slice



Additional details

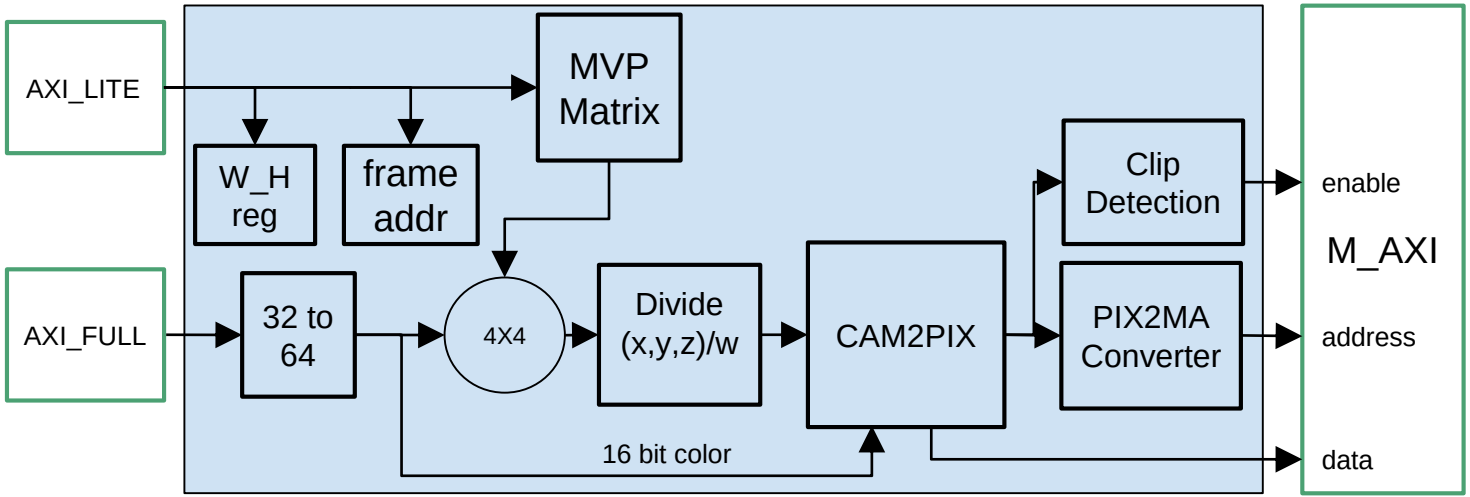
- When line drawing is active, two fifos are added into the data flow.
  - Drawing a line can take much longer than transforming a vertex into pixel space.
- Other simple constructs and state machine glue logic from the labs and class.
  - 32 to 64 bit temporal multiplexing for input
  - SM to give the next set of vertices to the line drawer when it is ready and there are two vertices available in the fifo’s

Line Drawing Simulation

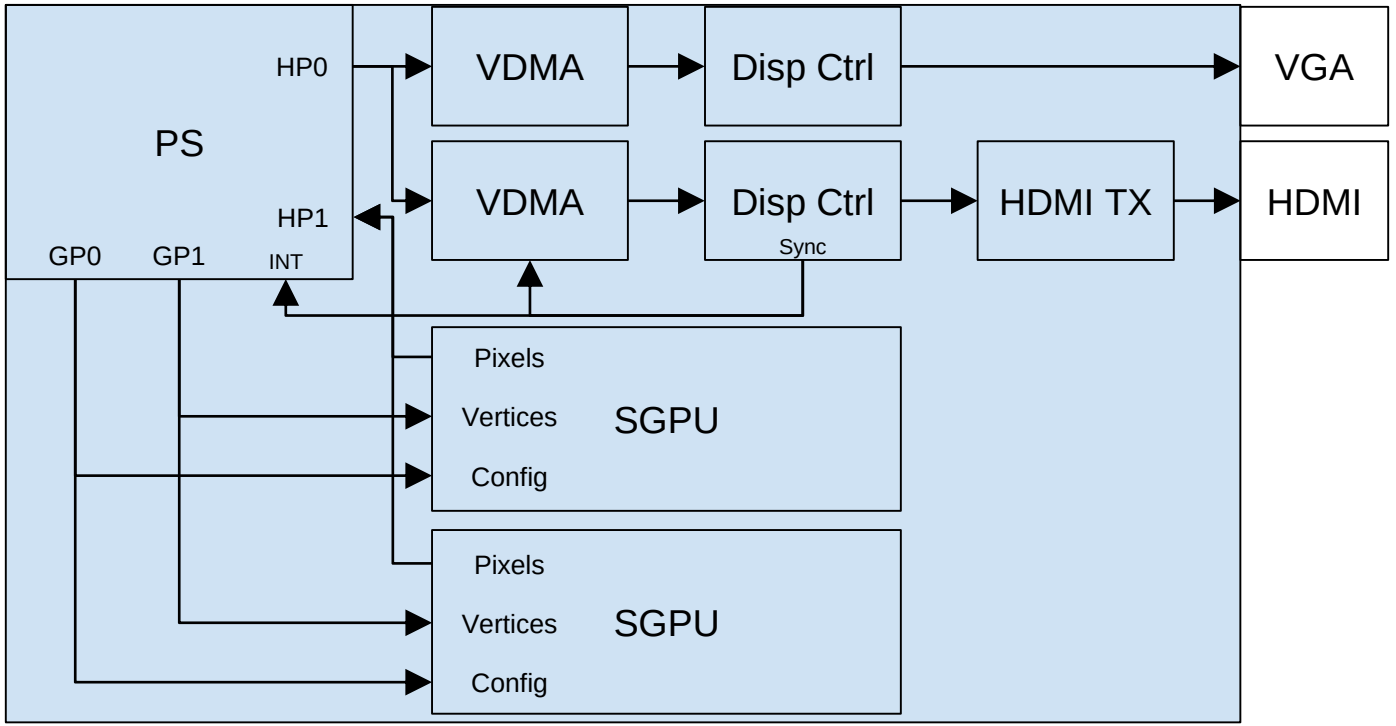


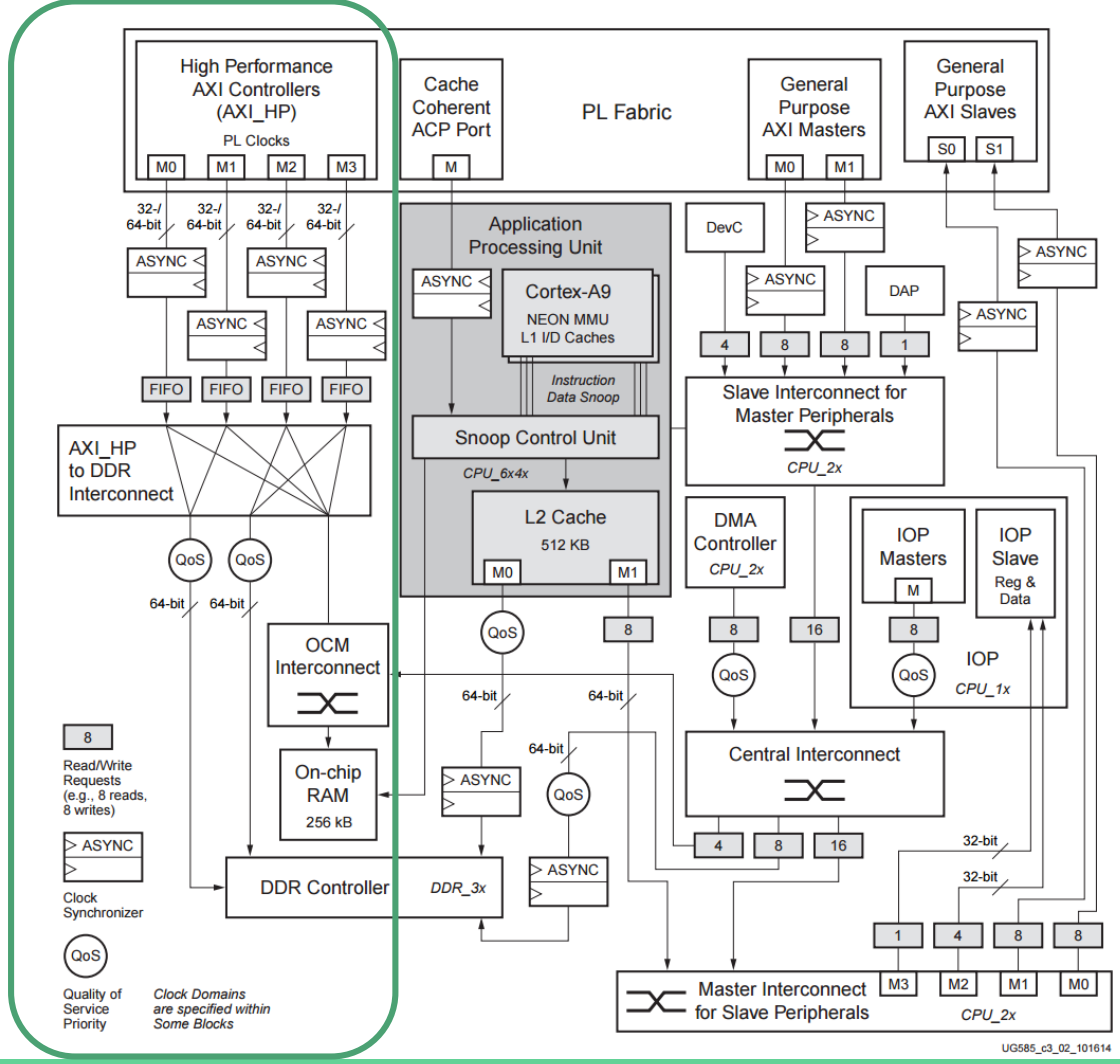
GPU in the Big Picture

Now that we've got a GPU, we add AXI Interfaces to talk to the CPU and directly to memory and then wrap it up as an IP core.



System Architecture

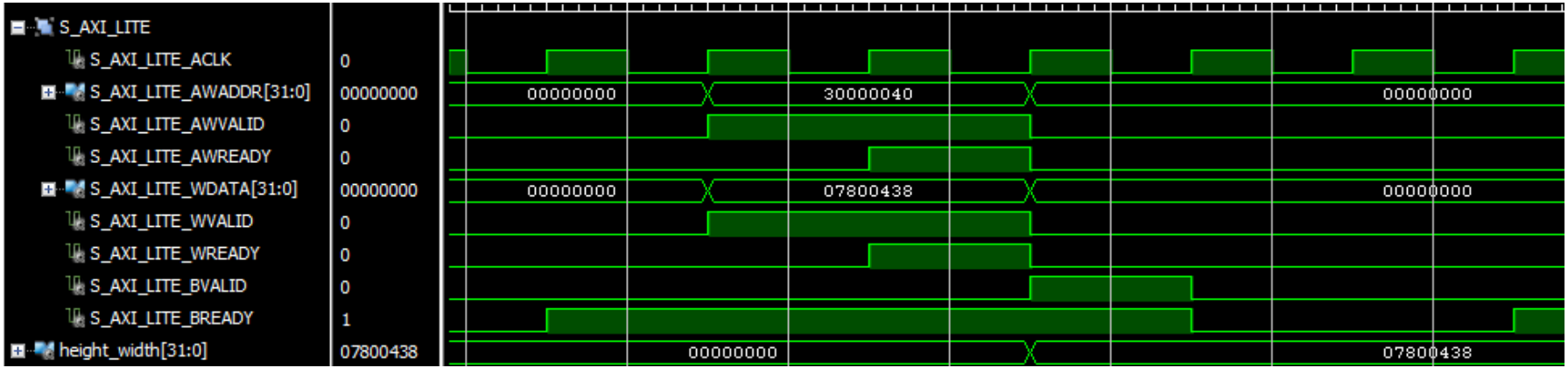




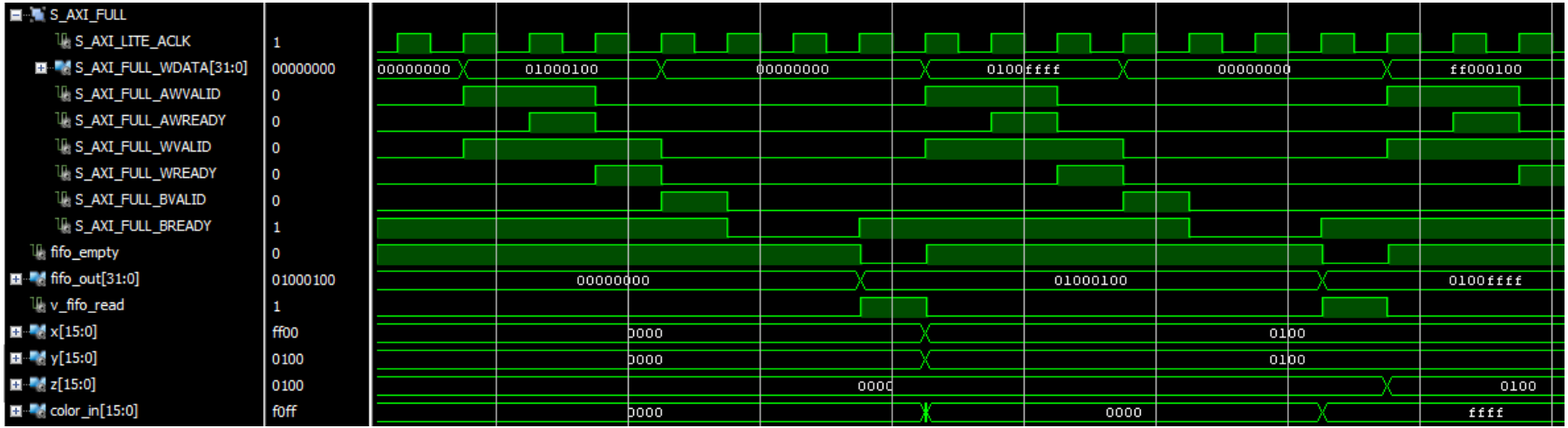
From Zynq TRM
UG585 (v1.10)
p. 64

AXI Lite Example Simulation

The AXI Lite interface is used for configuring the video frame's height and width, the base frame address, and the PV matrix.



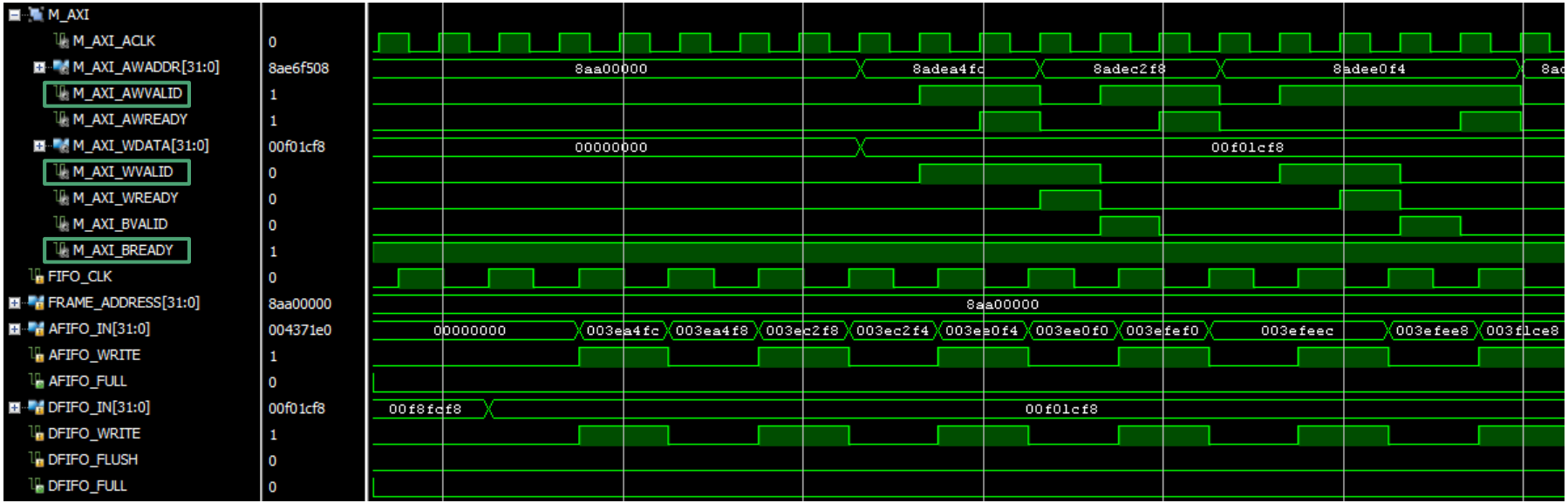
AXI Slave Full Example Simulation



First write loads X and Y

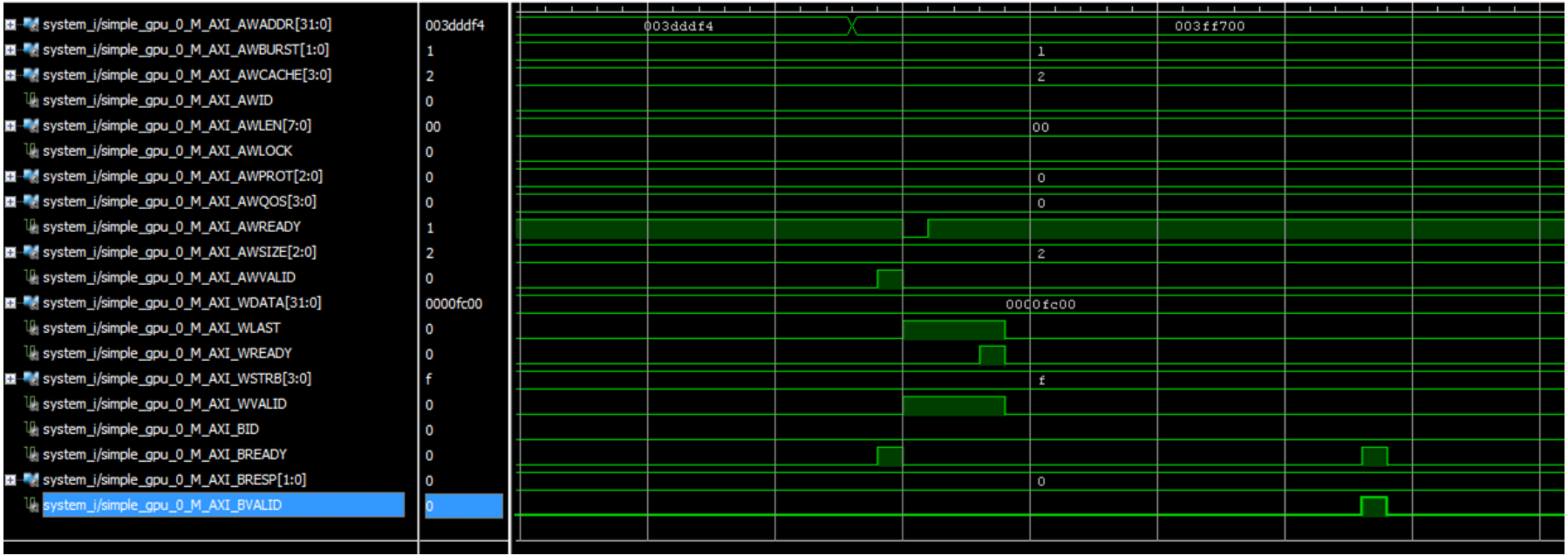
Second write loads Z and Color

AXI Master Full Example Simulation



Vivado ILA (Chipscope Pro)

The exact processor GP and HP AXI interfaces are hard to simulate since we don’t have a testbench for them. Using Vivado Integrated Logic Analyzer (ILA) we can debug hardware while it is running.



Problems

AXI has two separate handshake signals

- 1 for the address
- 1 for the data

When the system first initializes, it seems that the address “ready” goes high but the data does not causing the two output fifos to get out of sync

Line Drawing still has issues when a line leaves the screen edges

- The screen clip module ensures that a pixel is never sent to a memory location outside of the frame buffer, but the drawing starts to flicker as if vertices are being lost

Demo and Questions