FPGA Upscaler

Ben Cheng
Jay Mohile
Leo Han
Yong Da Li

PLANNING SLIDE - requirements

10min presentation (strict)

5min demo

5min questions

10min overflow (total 30min slot)

PLANNING SLIDE - topics

- what your project is about
  - HDMI into FPGA, output high resolution and high frame rate video
- what your initial goals were
  - frame rate and video upscaling 2x
- what you ended up with and why
  - what problems you had
  - changes you needed to make
  - block diagrams of originally planned and current
  - block diagram of final system
- what code blocks you coded, what you grabbed from other places
  - copied: digilient HDMI pass-through block diagram
  - custom: compute buffer, HLS to manage DMA, bilinear resolution interpolation
- design process
  - what did you do to ensure success?
  - what did you do to manage multiple designers?
- what you learned
- demo

PLANNING SLIDE - Demo Outline

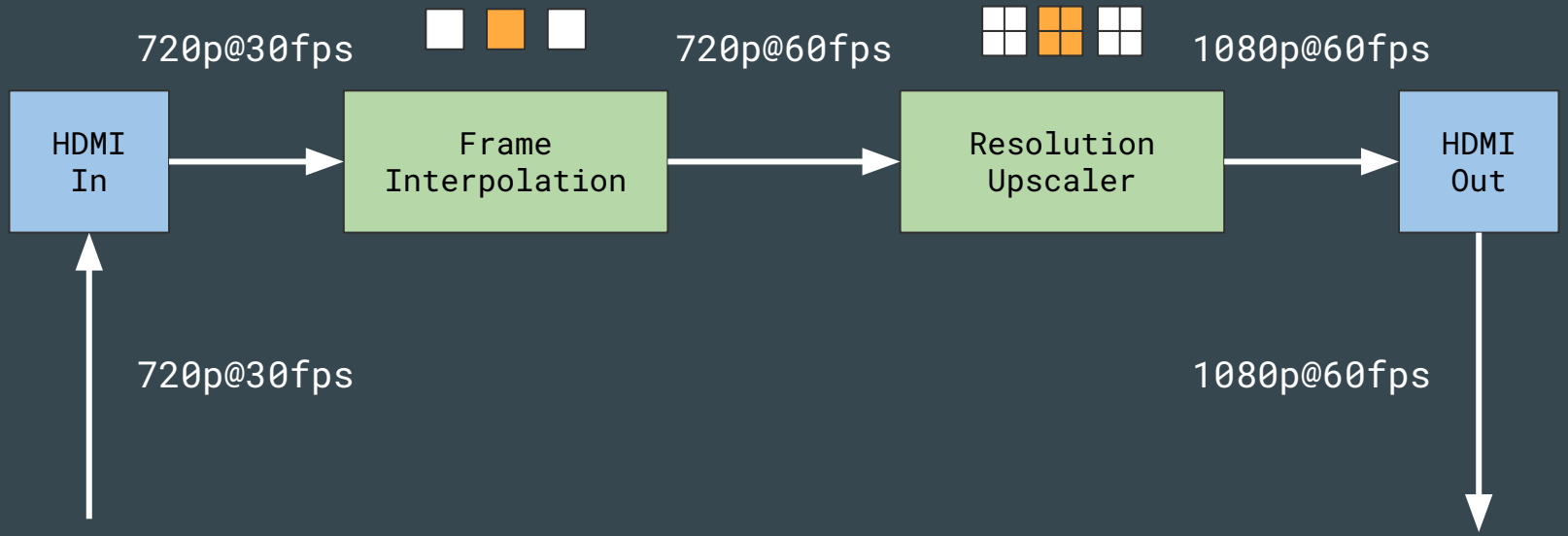
- demo

– show something working
– worst case - explain why you could not get something to work and it better be pretty good!

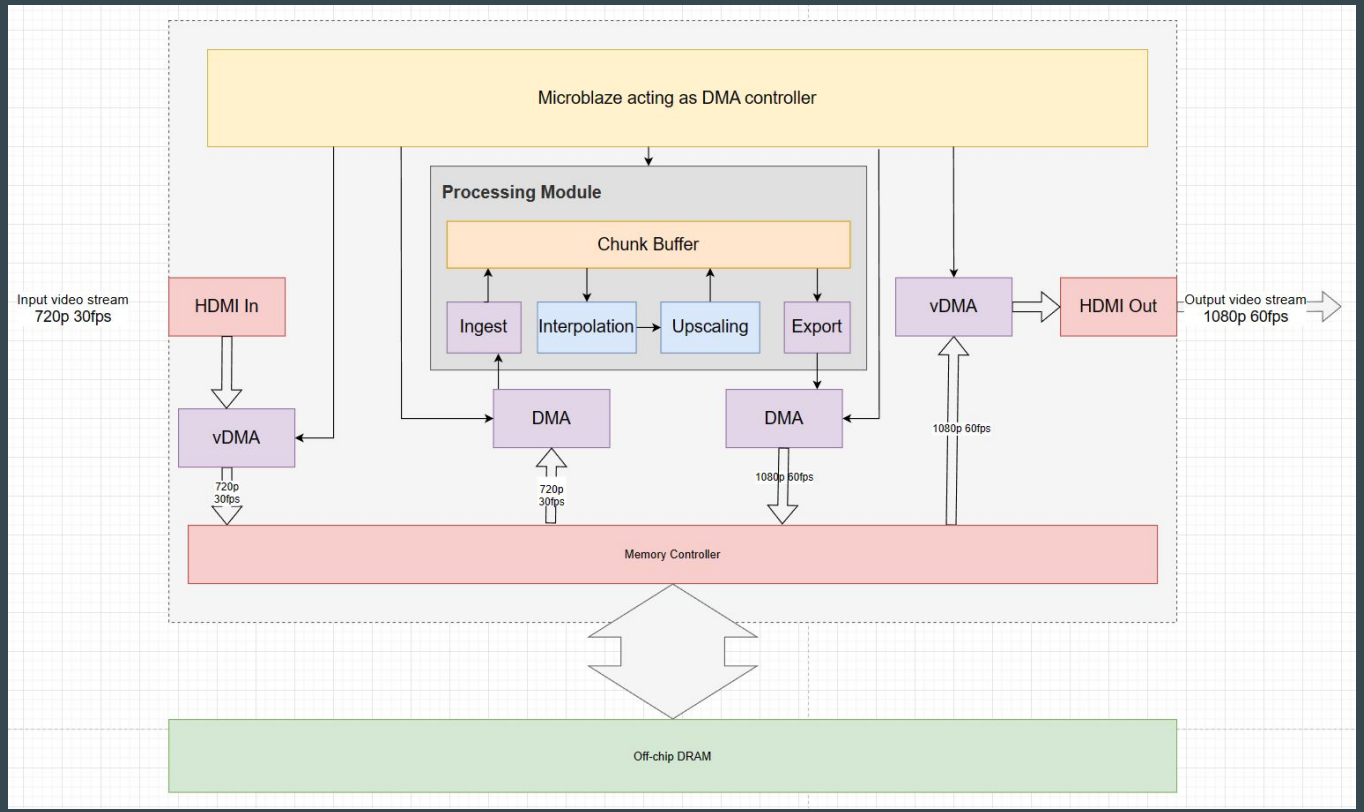
Original Project Goals - FPGA Upscaler

Input over HDMI: 1280 px by 720 px at 30 frames per second

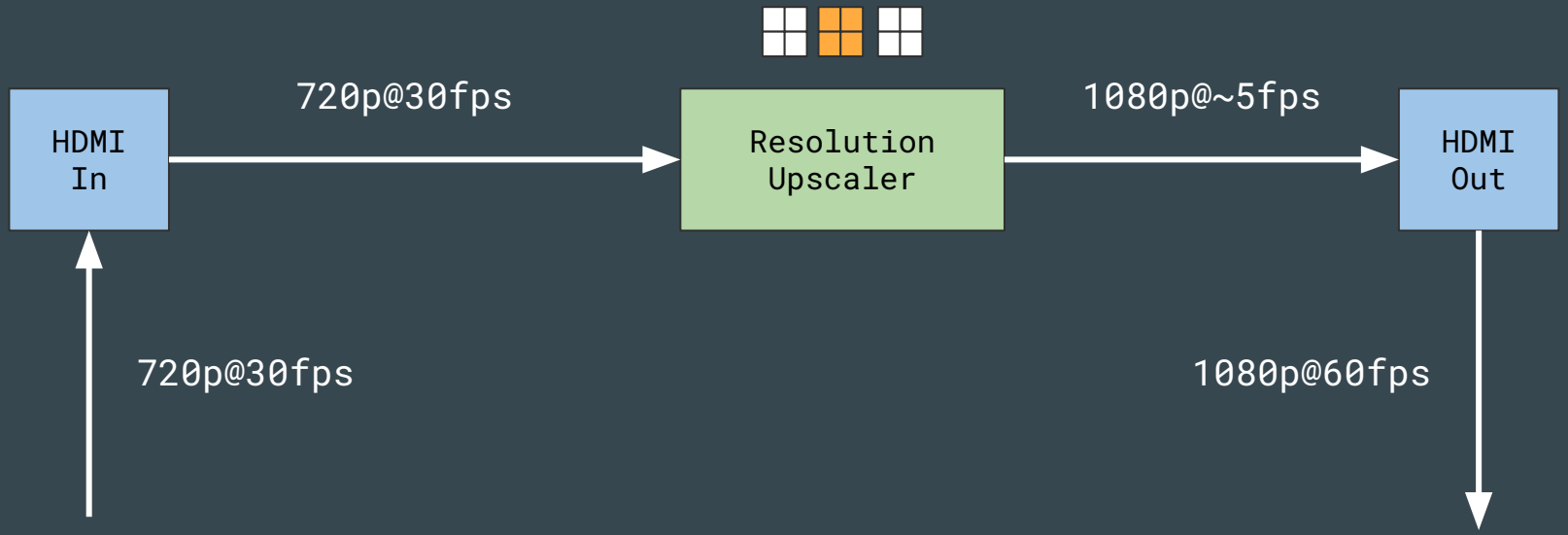
Output over HDMI: 1920 px by 1080 px at 60 frames per second



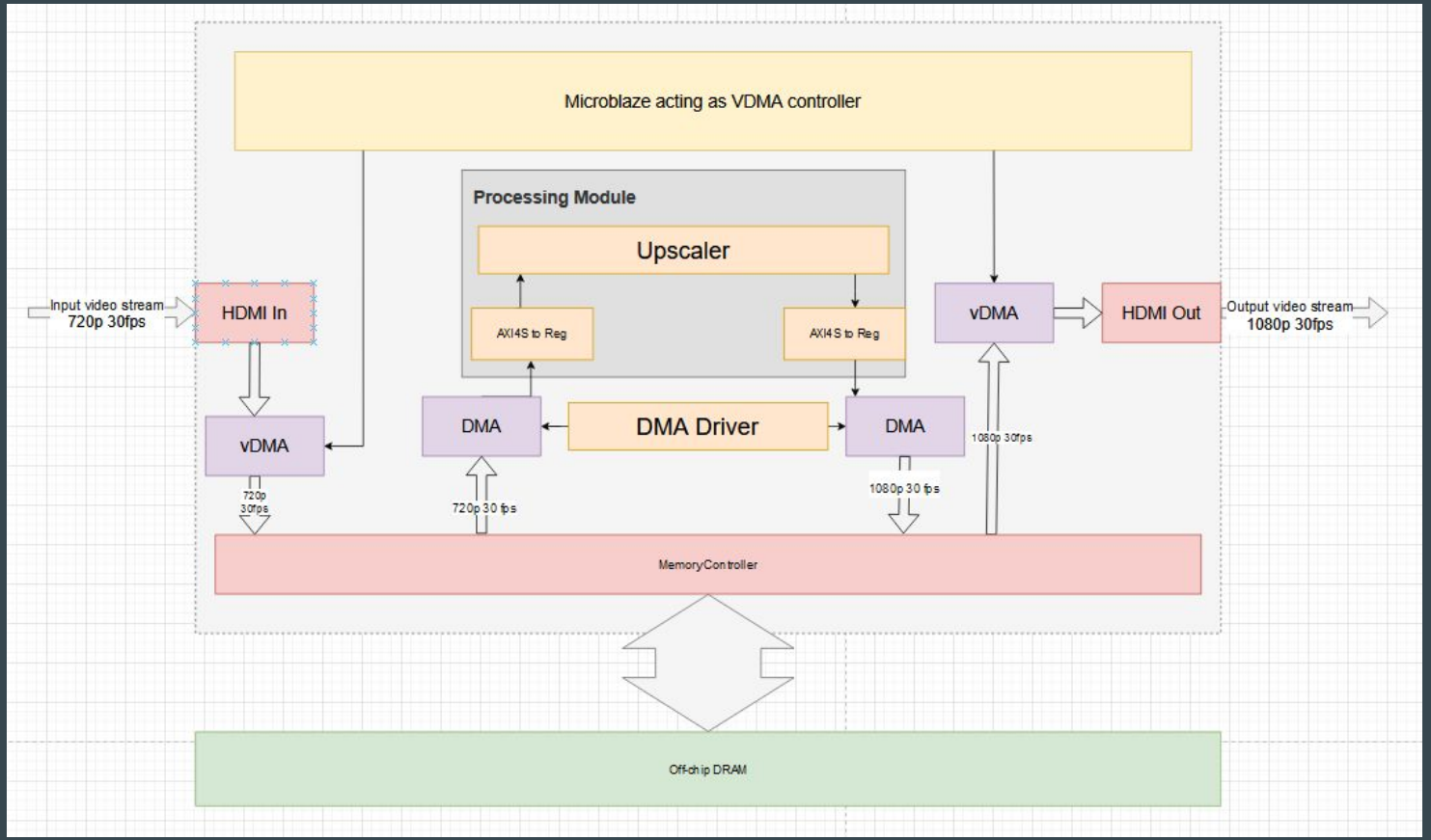
Mid-Project Block Diagram



Final Project - FPGA Upscaler



Current Block Diagram



Custom vs Pre-Canned Blocks

Custom

Combinational Upscaler

Low-Latency DMA Driver

Stream Buffer (Input / Output)

Transposer / Inverse Transposer

47% custom IP

Pre-Canned

Video In/Out IP

- dvi2rgb and rgb2dvi
- Video-In to AXI4-Stream
- AXI4-Stream to Video-Out
- AXI VDMA

Microblaze

MIG7 Memory Interface Generator

AXI DMA

Changes we made

1. No Frame Rate Interpolation

Tried output at double frame rate (60 fps) at 1080p

- VDMA experienced underflow caused by DDR bandwidth limitations
- HDMI in and out are taking 450Mbits/sec of DDR bandwidth
- Need to write/read another 450Mbits/sec for upscaling and frame rate interpolation (not possible)

Eliminated frame interpolation to reduce load on DDR

2. New Memory <-> Processing Architecture

Custom DMA driver (HLS) fetches tiles from DDR memory

Custom logic (HLS) drives DMA to write upscaled tiles to DDR memory

- Previously used microblaze to manage start/stop DMA's

Other Attempted Solutions

Full custom logic (from fetching blocks from memory to upscaling to write back to memory) using HLS

- ran out of development time to integrate with video hardware and platform

Controlling DMA with Microblaze with both Xilinx C library and direct registers

- Low throughput limited by latency of initiating DMA transactions via software

Design Process

90% in-person work sessions with all group members

parallelize everyone working on separate things, then pair-up with integration

during compile cycles, we pair-up and help each other

when debugging errors, have multiple people look at the error

Used git to track source verilog files. Added block diagrams as exported TCL scripts.

What we learned

Creating and packaging custom IPs with Vivado HLS

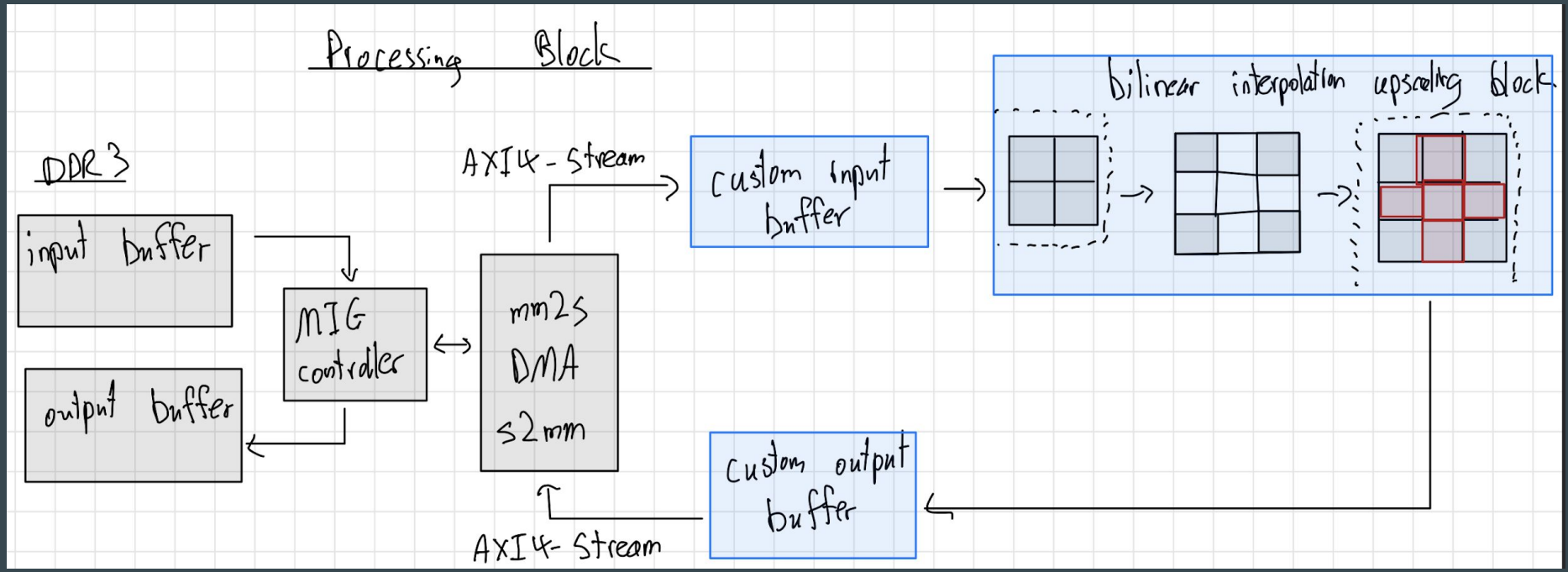
- HLS can save time and reduce development
- HLS requires good understanding of HLS tool and programming model

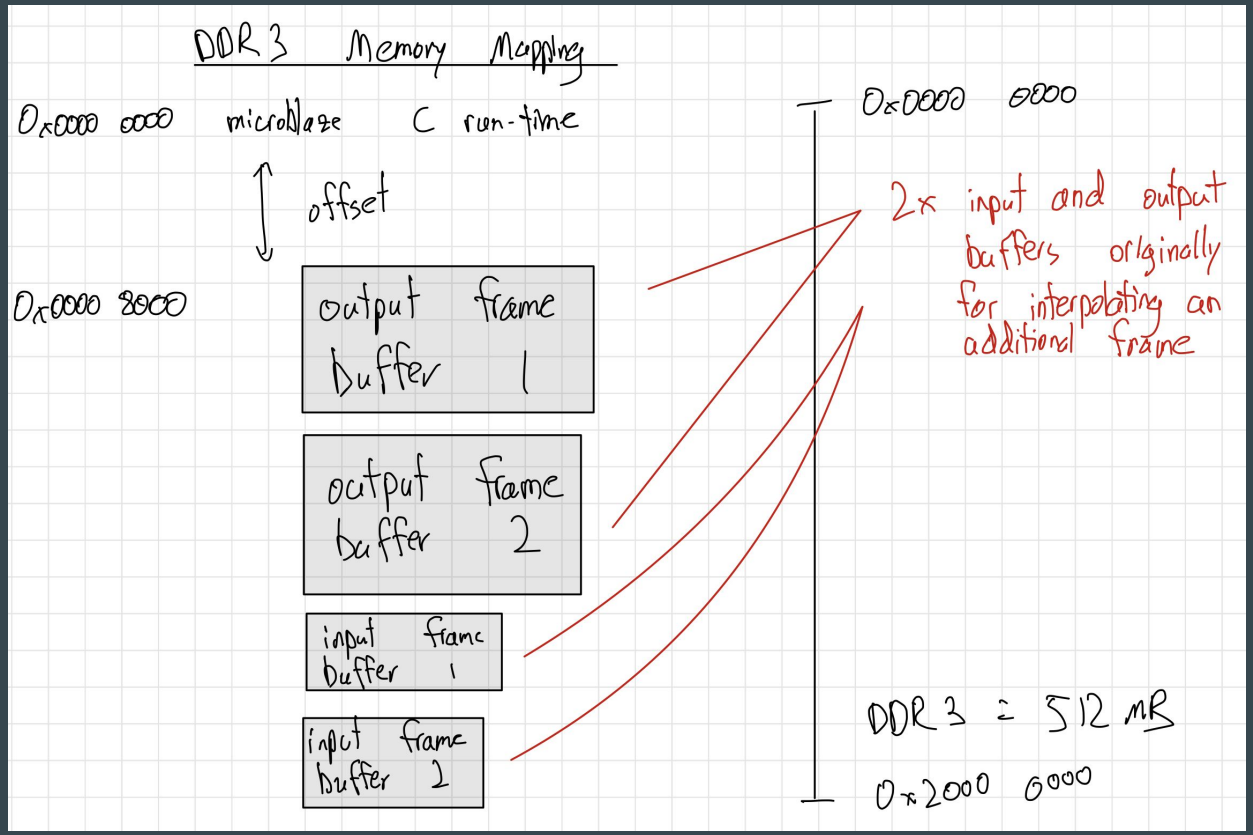
Interfacing with HDMI video input and output (incl. VDMA)

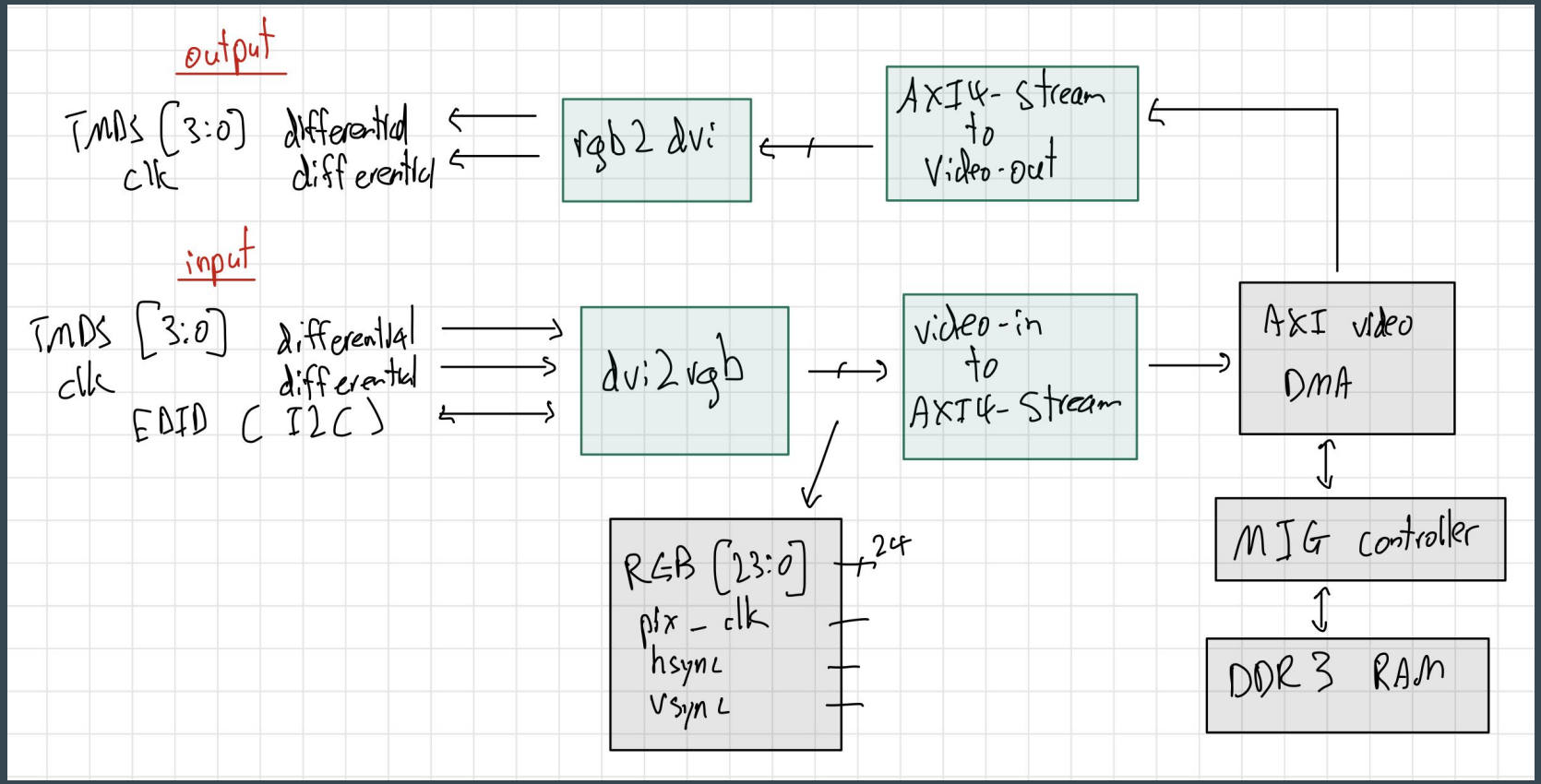
Creating custom AXI-Stream interface with Verilog

Hardware takes very long to iterate

- RTL simulation and test benches save development time and effort


Processing Block
DDR3
input buffer
output buffer
MIG controller
mm2s DMA
s2mm
AXI4-Stream
AXI4-Stream
custom input buffer
custom output buffer
bilinear interpolation upscaling block


DDR3 Memory Mapping
0x0000 0000
microblaze C run-time
offset
0x0000 8000
output frame buffer 1
output frame buffer 2
input frame buffer 1
input frame buffer 2
0x0000 0000
2x input and output buffers originally for interpolating an additional frame
DDR3 = 512 MB
0x2000 0000


output
TMDS [3:0]
clk
differential
differential
rgb2 dvi
AXI4-Stream
to
Video-out
input
TMDS [3:0]
clk
differential
differential
EDID (I2C)
dvi2rgb
video-in
to
AXI4-Stream
AXI video
DMA
MIG controller
DDR3 RAM
RGB [23:0]
24
pix_clk
hsync
vsync