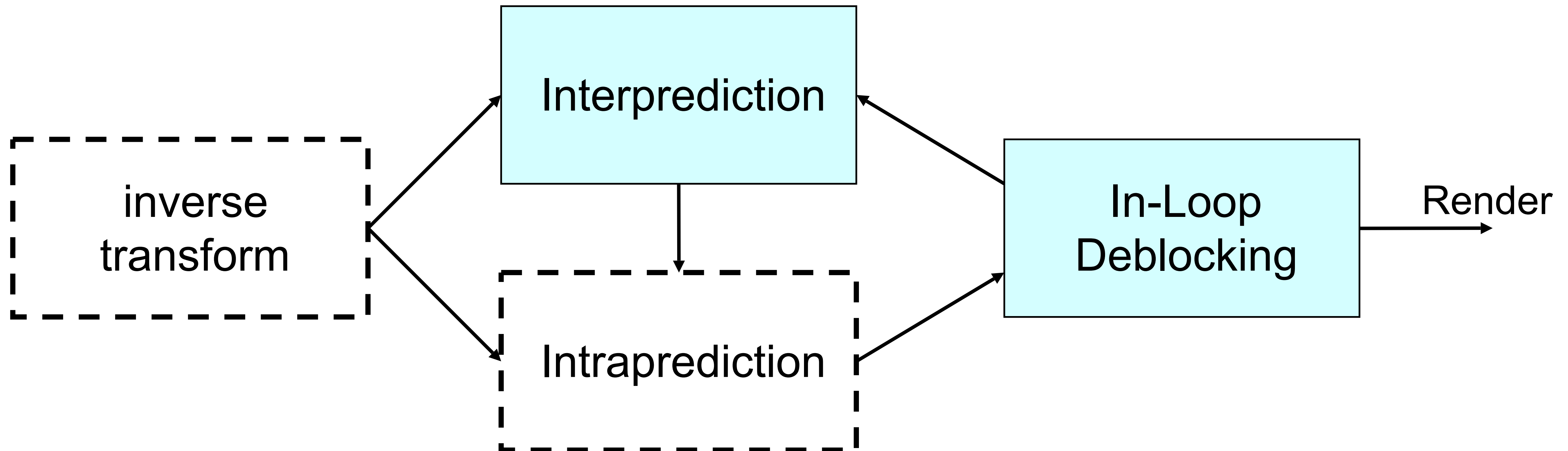
AMD Smarter Choice

H.264 Decoding

Optimization Example: H.264

AMD
Smarter Choice



Interprediction – filter data from previously decoded frames

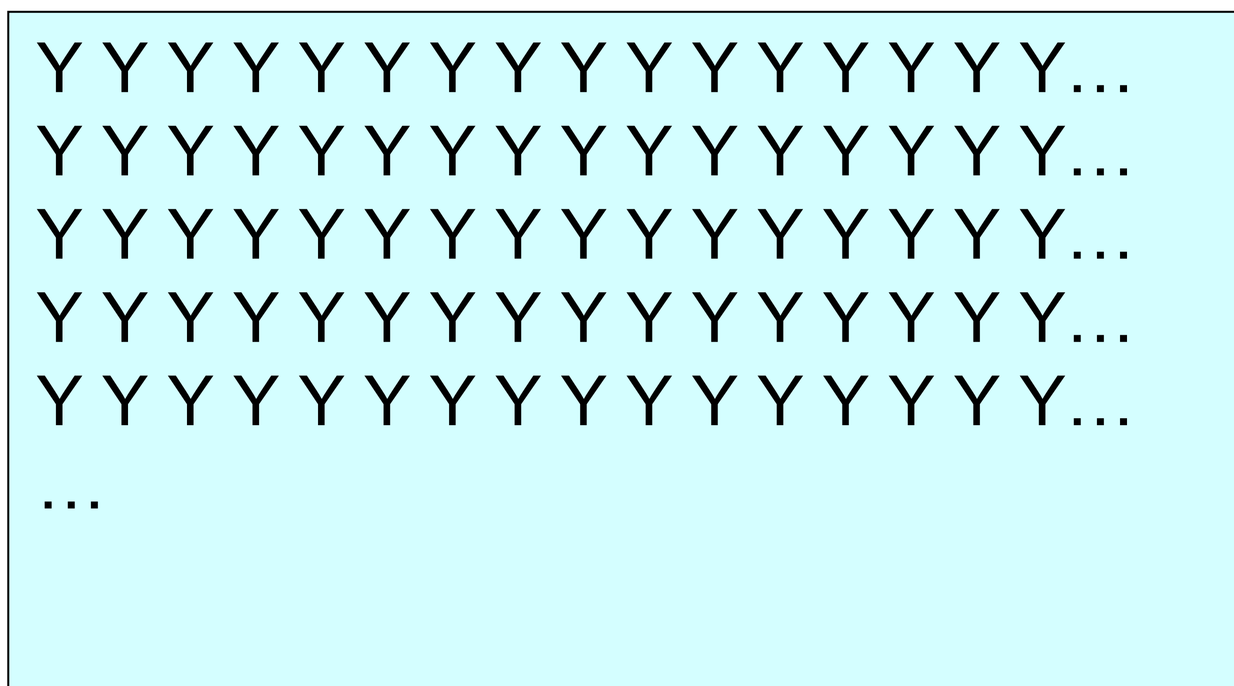
Deblocking – filter out block edges

Today: Loosely based on DX9 HW Implementation

Problem Domain

Max Resolution 1080p = 1920x1088 “RGBA” pixels (3Bpp)

- Luma (Y) = 1920x1088 pels (1Bpp)
- Chroma (U and V) = 960x544 pels each



Interprediction

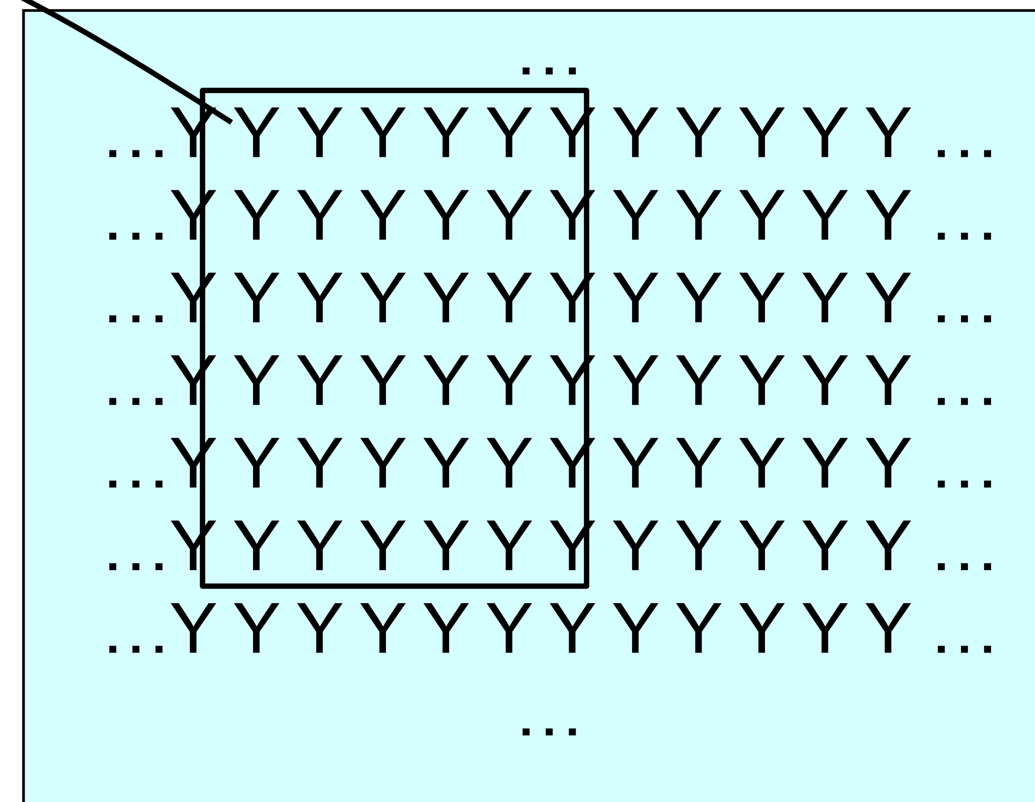
AMD
Smarter Choice

Decode each pel by interpolating a subregion of previously decoded frames

Example case: 6x6 fetches per pel

- Kernel not aligned to x4 boundary

Target



Source

Interprediction

Decode each pel by interpolating a subregion of previously decoded frames

Example case: 6x6 fetches per pel

- Kernel not aligned to x4 boundary

Target

Source

Interprediction

Decode each pel by interpolating a subregion of previously decoded frames

Example case: 6x6 fetches per pel

- Kernel not aligned to x4 boundary

Target

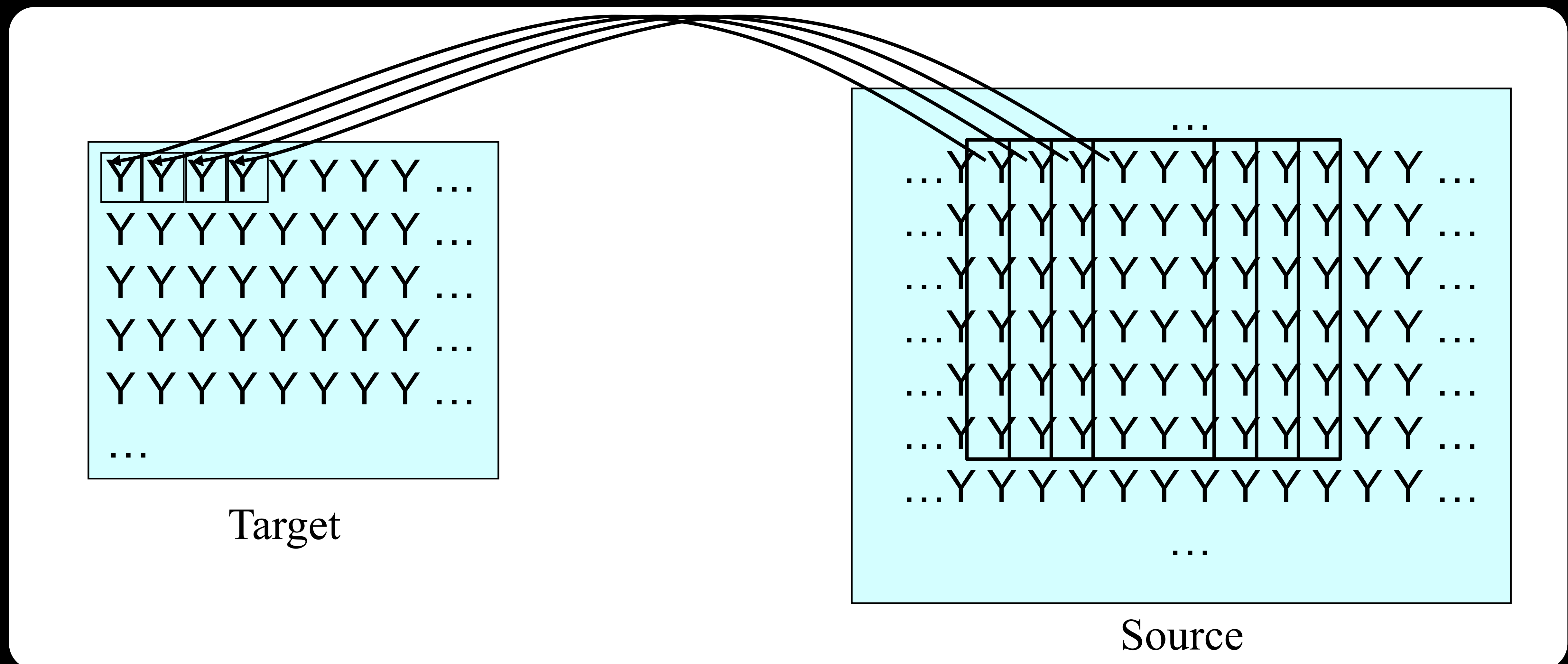
Source

Interprediction

Decode each pel by interpolating a subregion of previously decoded frames

Example case: 6x6 fetches per pel

- Kernel not aligned to x4 boundary



Do more work per pixel

Problem: Shader has a lot of texture fetches

Share already fetched data by processing four pels in one pixel (thread)

- Consolidate texture fetches & Consolidate ALU instructions
- Consolidate writes (four bytes per pixel)
- Increase number of threads

Source

Target

Saved 96 fetches!

Optimizing the algorithm

AMD Smarter Choice

Theoretical Model for 4 pels per pixel

- 115 ALU, 56 TEX, 66 Bytes In/Out
- 480x1088 pixels processed
- From previous equations:
  - ALU time = 2 ms
  - TEX time = 3 ms
  - Mem time = 0.67 ms
  - Theoretical time = 3 ms

Using the theoretical model we can find the problems and fix it

Do even more work per pixel!

Use multiple render targets

- 16 pels per pixel

Y Y Y Y Y Y Y Y …
Y Y Y Y Y Y Y Y …
Y Y Y Y Y Y Y Y …
Y Y Y Y Y Y Y Y …
Y Y Y Y Y Y Y Y …
…

Target

…
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…Y Y Y Y Y Y Y Y Y Y Y Y …
…

Source

Saved 522 fetches!

What happens to the model?

AMD Smarter Choice

Theoretical Model for 16 pels per pixel

- 304 ALU, 85 TEX, 107 Bytes In/Out
- 480x272 pixels processed
- From previous equations:
  - ALU time = 1.32 ms
  - TEX time = 1.12 ms
  - Mem time = 0.29 ms
  - Theoretical time = 1.32 ms

Shader goes from TEX bound to ALU bound

Oops...

Render targets split the image

0
1
2
3

0

1

2

3

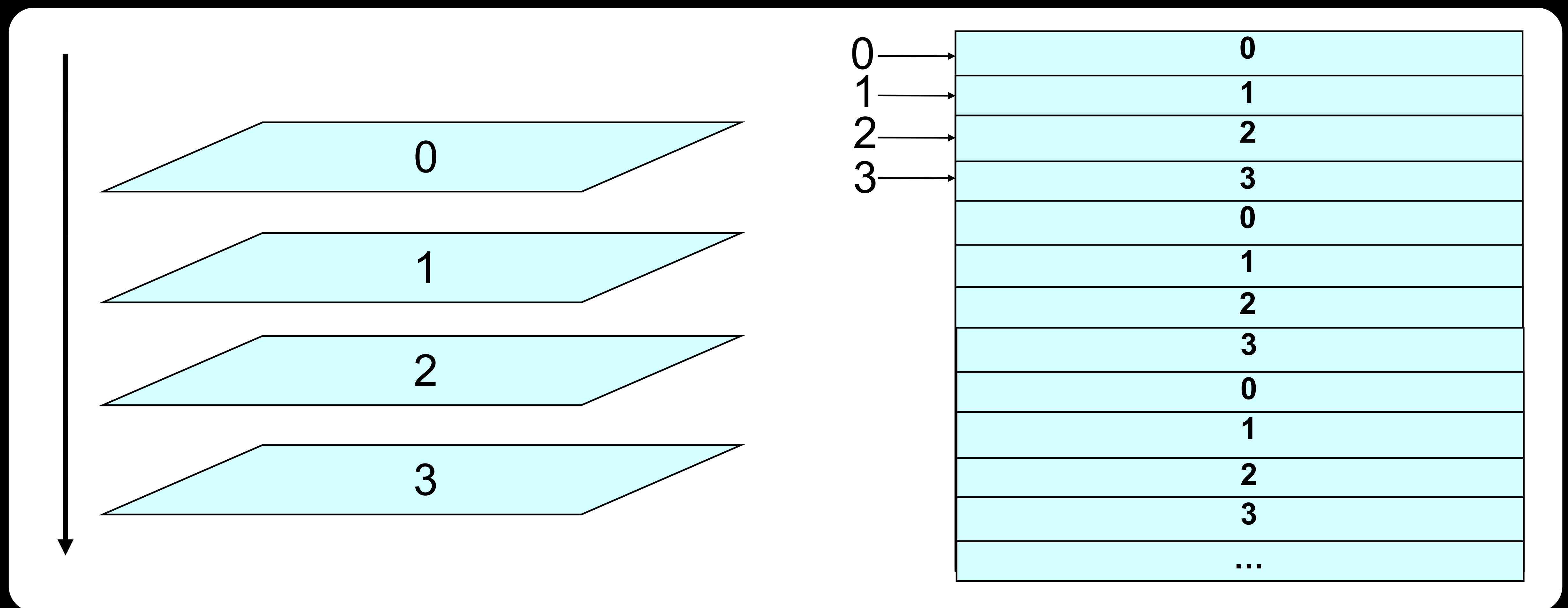
Could do a copy shader at the end, or...

Interleave Render Targets

AMD
Smarter Choice

Hardware supports interleaved render targets

Four surfaces appear as one



Scatters are slow, MRTs are fast

Back to the real world...

AMD Smarter Choice

Compare actual performance with theoretical

Easy way to determine next step

- If you're close, you probably have to redesign if you've missed your performance target
- If you're off, time to look at the hardware

Interprediction Filtering

AMD Smarter Choice

Each block of 4x4 pels are processed differently

- Up to 6 filter cases
- Up to 2 prediction directions
- Up to 2 block types (frame/field)
- Up to 16 input textures (max 4 for 1080p)

With up to up to 384 possible cases, that's a pretty big shader!

Branching isn’t perfect

AMD Smarter Choice

Shader length

Branch overhead

Divergent paths between pixel blocks
- One pixel branch can stall the others
- “Branch Granularity”

Our old friend the Z-Buffer

AMD
Smarter Choice

Use a fast z-pass to initialize the buffer

Render each “case” separately

Fast because of top-of-the-pipe pixel kills

Very little branching overhead

Potentially shorter shaders

More threads

Deblocking

AMD
Smarter Choice

Filters block edges

Performance can suffer due to render dependencies

- Input to next pixel is output from previous pixel
- Frame divided into MB of 16x16 pels
- 4 vertical and 4 horizontal passes per MB
- For 1080p frame, up to 748 passes

Example…

Vertical Deblocking

AMD
Smarter Choice

8 pels

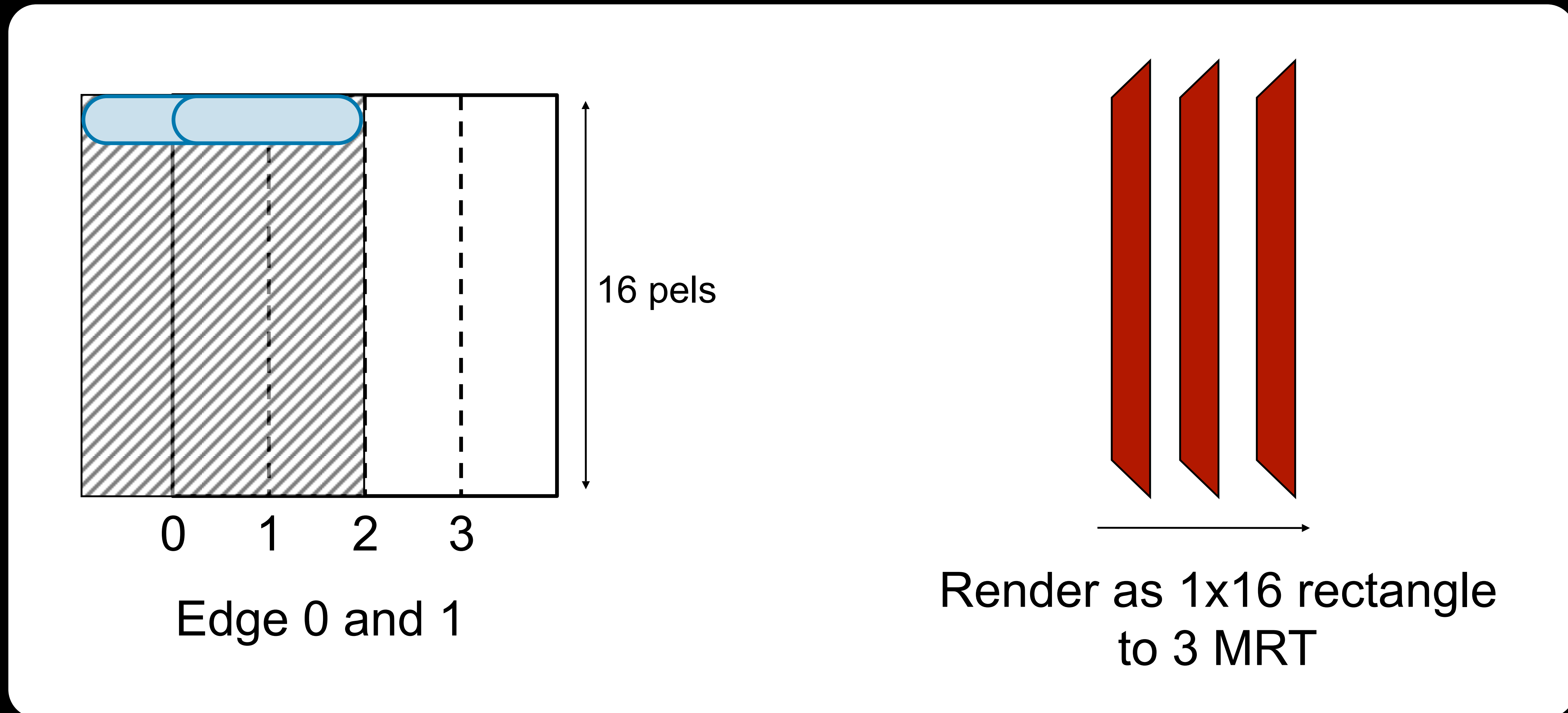
Edge 0	Edge 1	Edge 2	Edge 3
0 1 2 3	0 1 2 3	0 1 2 3	0 1 2 3

Previous macroblock

Do more work in a pixel revisited

Save on number of passes

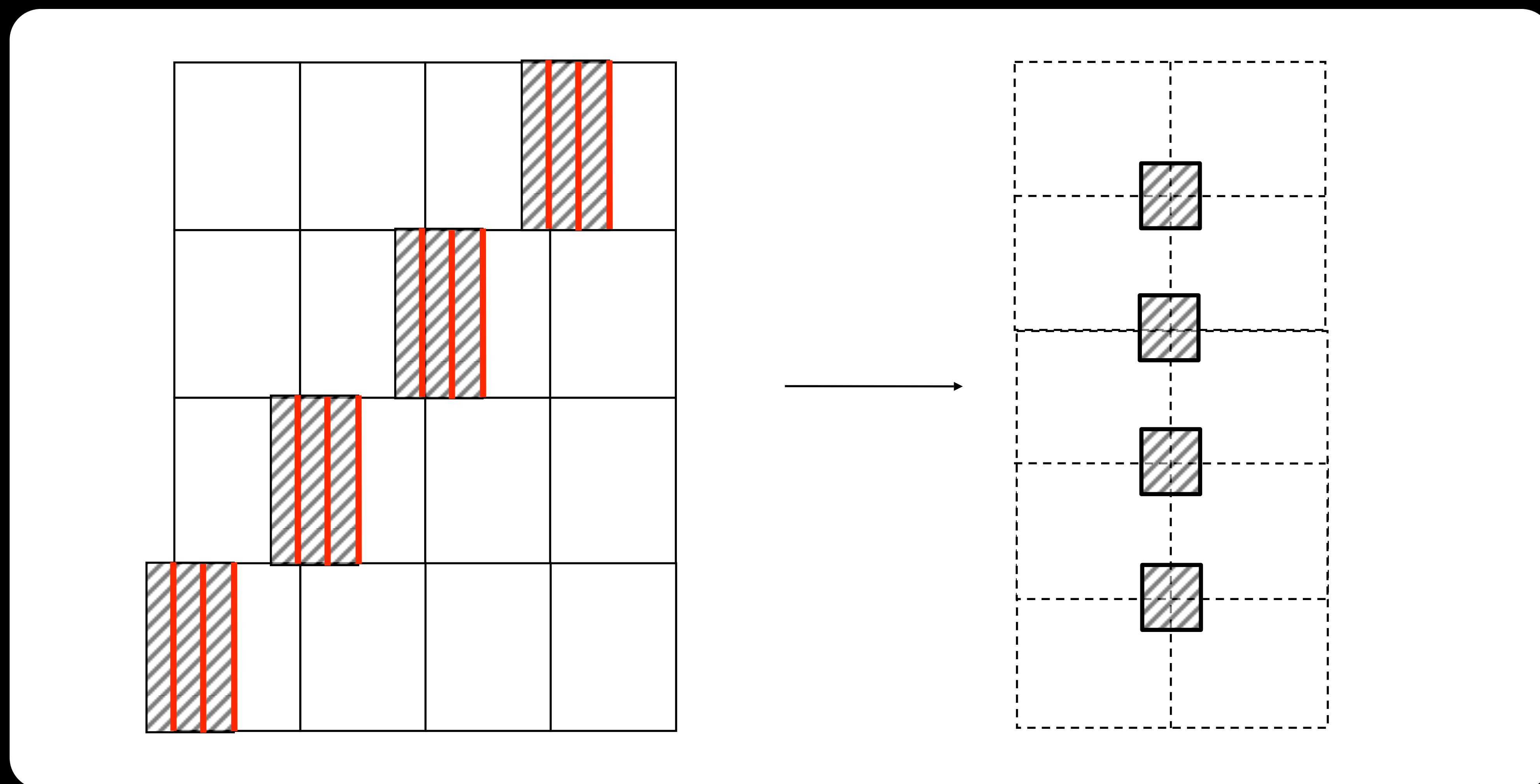
- If I have the data, why not use it?
- Filter two edges in one pixel



Hmmm....

Performance improves, but is still off

- Hardware units arranged in 2x2 pattern
- 1x16 strip only uses half the pipes
- Rearrange the rendering



Read/Write to one texture

Because of dependencies the output is the input to the next pass.

Hardware allows reading and writing from the same texture

No need to reset states

Getting more info

Look at performance counters

- HW Utilization
- Texture cache miss
- Z-buffer statistics

Cache and Memory Tips

AMD
Smarter Choice

Rearrange the data

Play with the memory layout

- CAL API allows more fine grain control

Memory performance is hard to predict

- Access pattern
- Memory subsystem is unknown

Tip: Try replacing input textures with a very small texture (1x1)

Doesn't help with dependent texture fetches