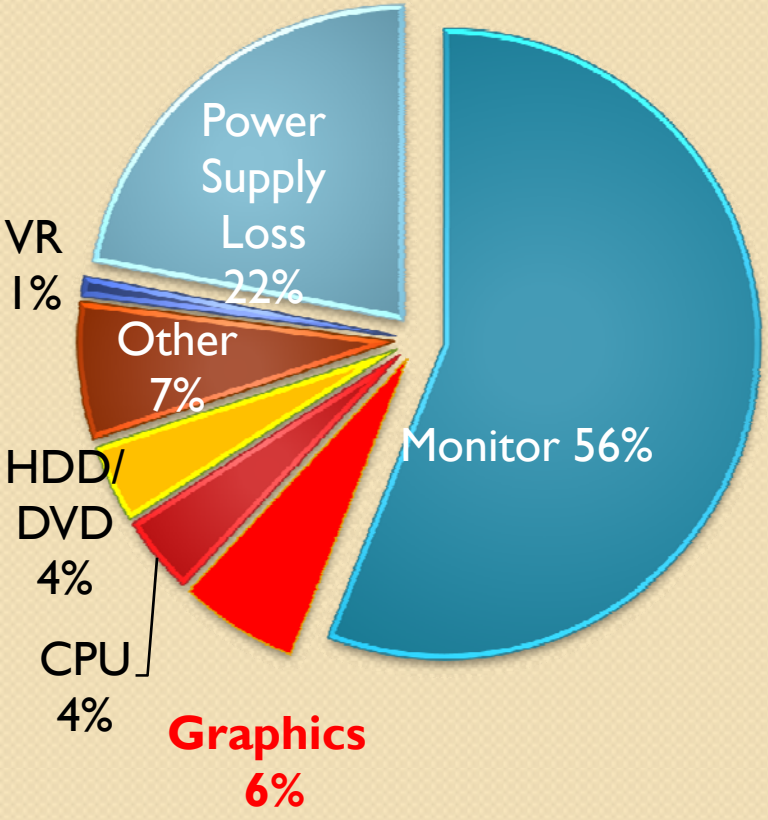
Energy Efficiency in Graphics Rendering

Preeti Ranjan Panda
Department of Computer Science and Engineering
Indian Institute of Technology Delhi

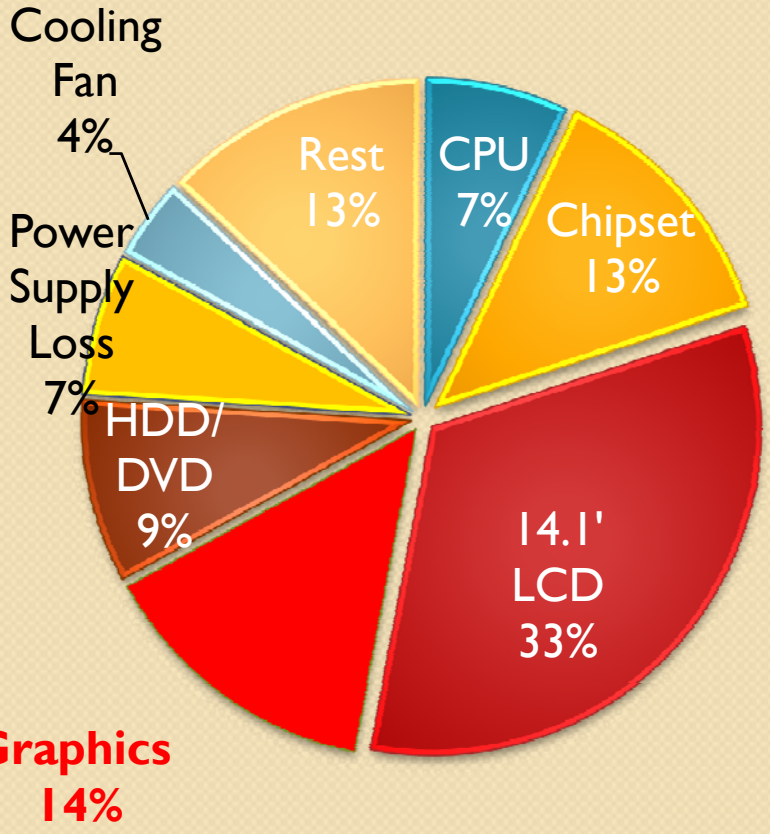
Presentation at TU Dortmund, June 2011

Graphics Power Consumption

Desktop computer



Mobile computer

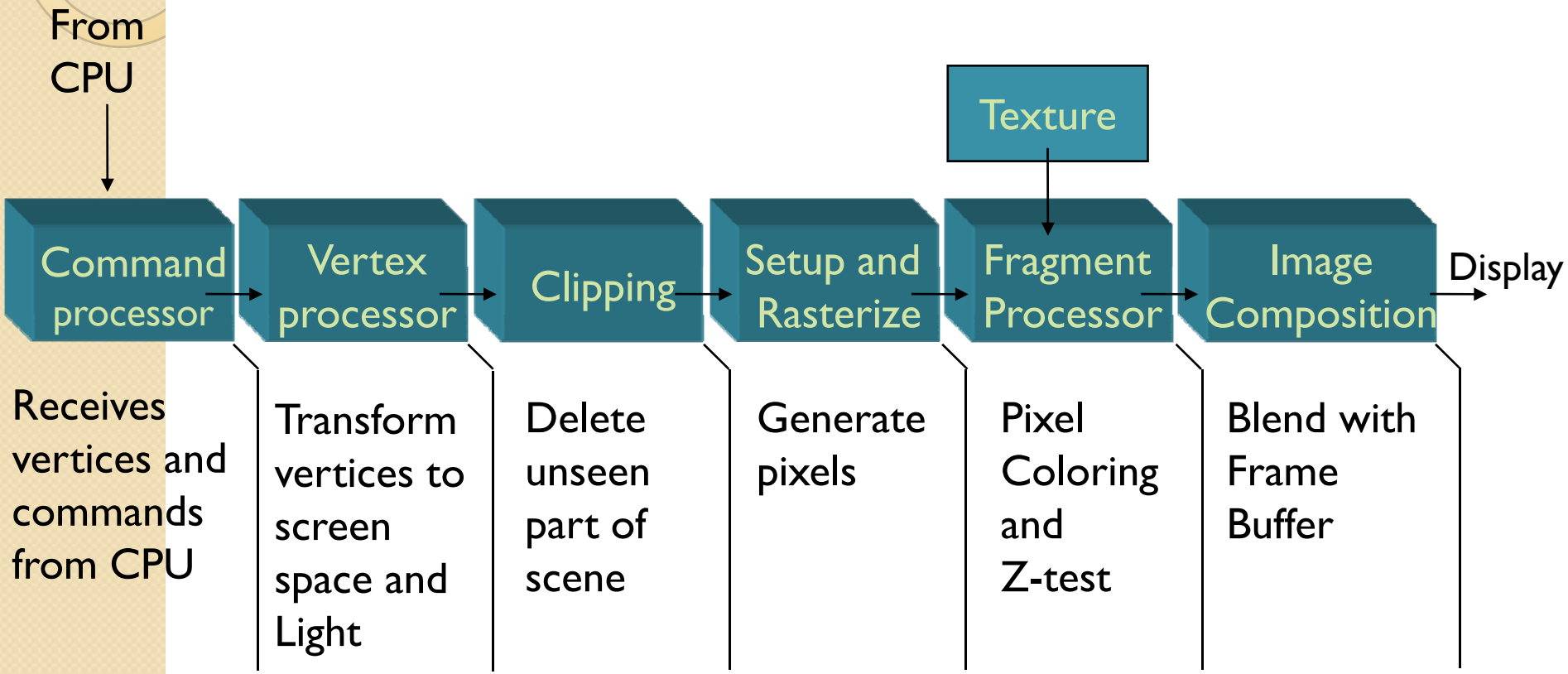


[Ref : PC Energy-Efficiency Trends and Technology, source: intel.com]

Observation

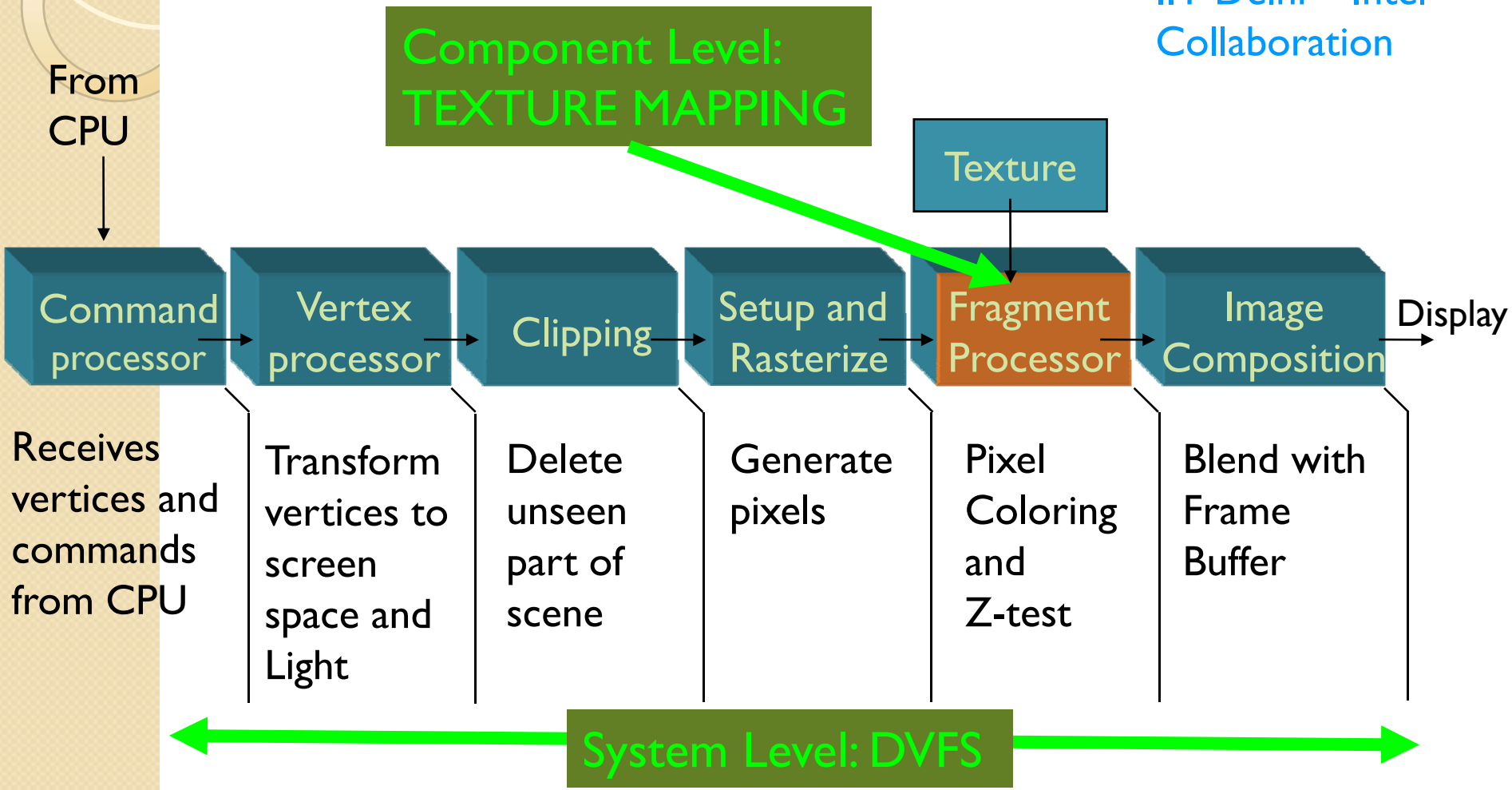
- GPU/Graphics rendering power is significant (greater than CPU)
- Yet, very little research on GPU energy efficiency!
  - GPU performance was/is primary
  - Proprietary GPU architectures

Graphics Pipeline



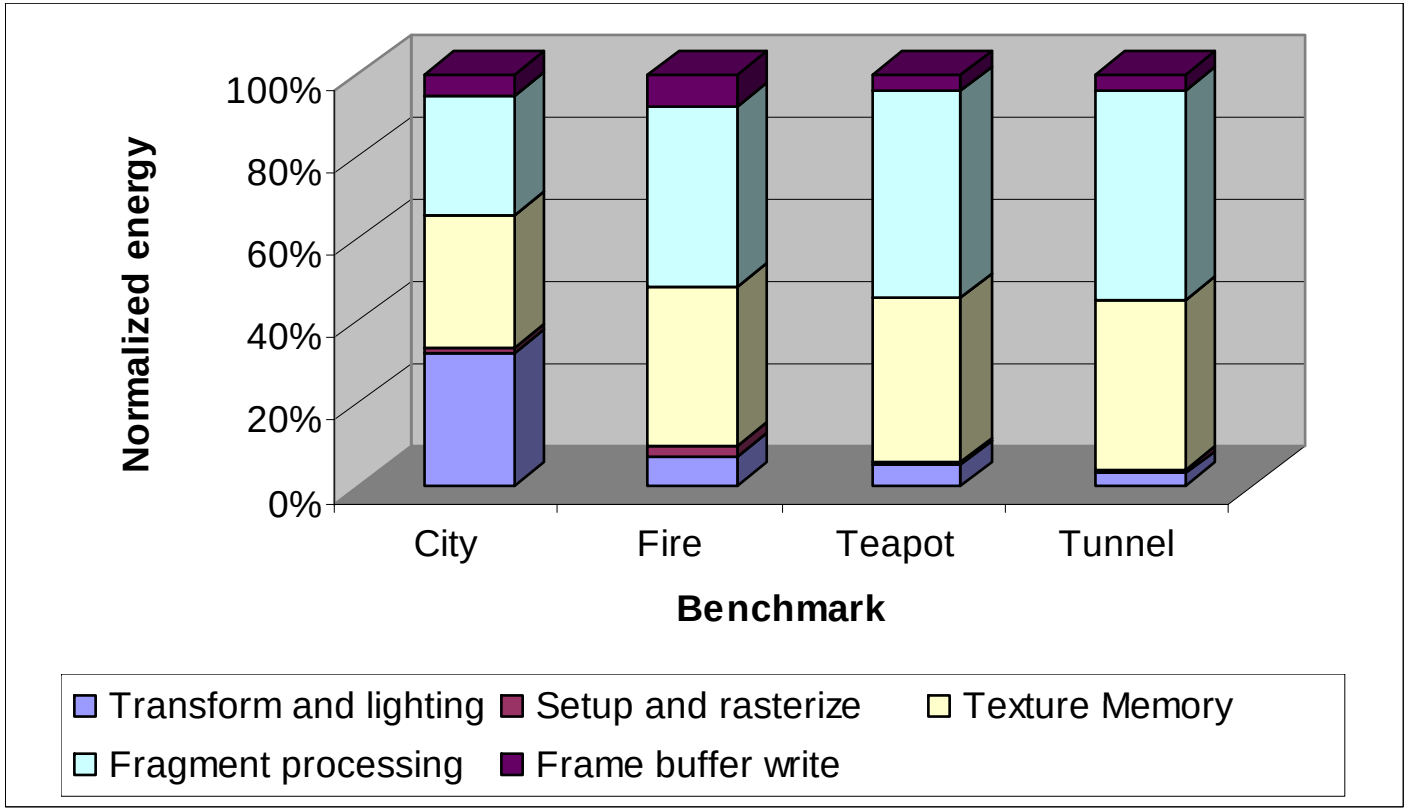
Adding Energy Efficiency

IIT Delhi – Intel Collaboration



LOW POWER TEXTURE MAPPING [ICCAD'08]

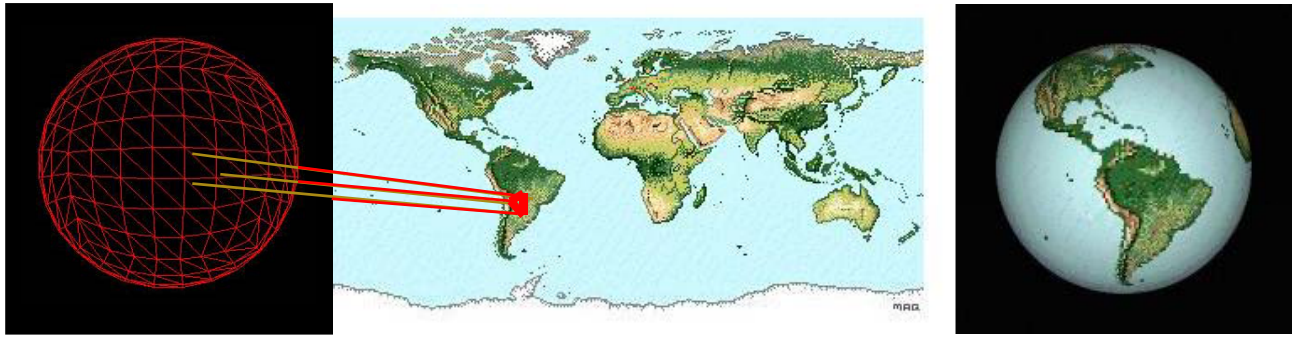
Power Profile of the Pipeline



Texture memory consumes 30-40% of total power.

Texture Mapping

- Add detail and surface texture to an object.
- Reduces the modeling effort for the programmer.

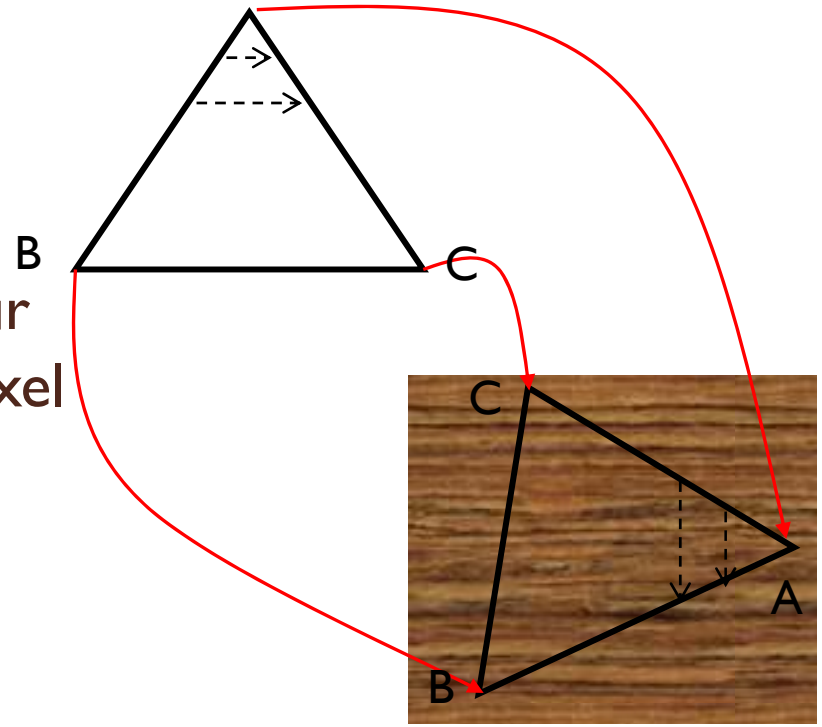


Object Texture Texture Mapped Object

Texture Filtering

- Texture space and object space could be at arbitrary angles to each other

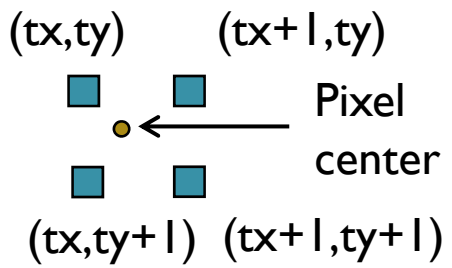
- Nearest neighbor
- Bilinear interpolation : weighted average of four texels nearest to the pixel center.

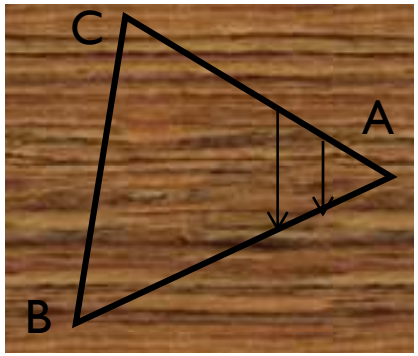


B.V. N. Silpa and P. R. Panda, 2011

Texture Access Pattern

- Texture mapping exhibits high spatial and temporal locality
  - Bilinear filtering requires 4 neighbouring texels
  - Neighbouring pixels map to spatially local texels
  - Repetitive textures



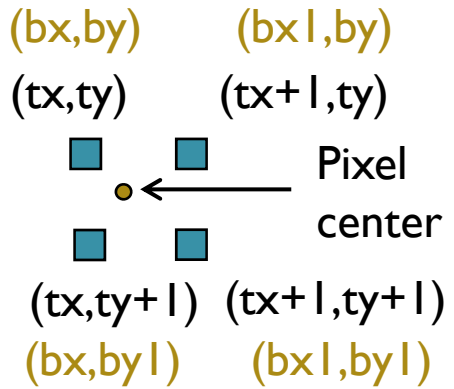


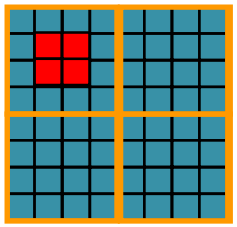
Blocking and Texture Cache

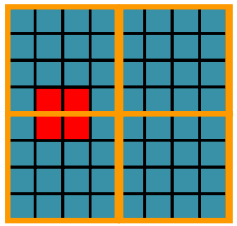
- Blocked Representation
  - Texels stored as 4x4 blocks
  - Reduces dependency on texture orientation, and exploits spatial locality
- Texture memory accessed through a Cache hierarchy ("TEXTURE CACHE")
- Familiar architectural space
- BUT, application knowledge could help improve the HW over a "standard cache"

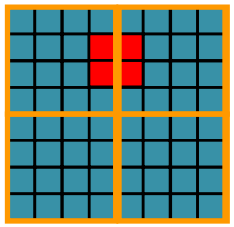
Predictability in Texture Accesses

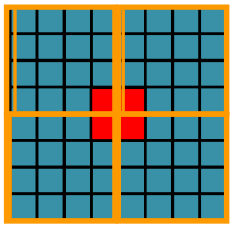
- Access to first texel gives information about access to the next 3 texels
- The four texels could be mapped to either one, two or four neighbouring blocks.




Case 1


Case 2


Case 3

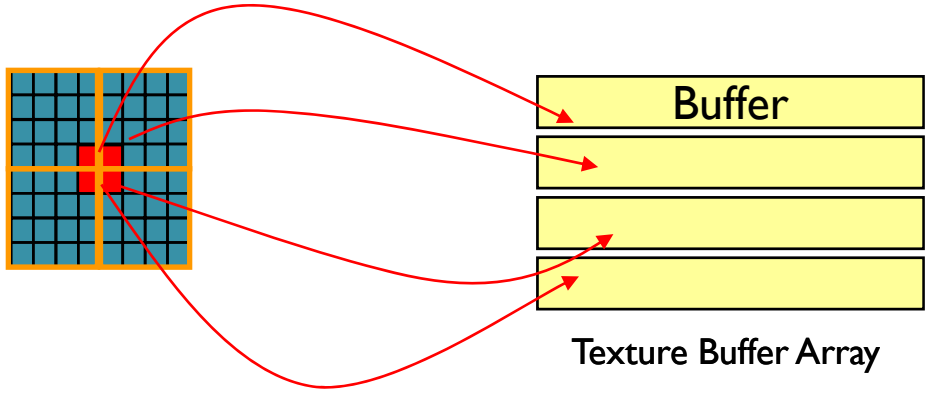

Case 4

Low Power Texture Memory Architecture

- Lower power memory architecture than Cache for texturing
  - Use a few registers to filter accesses to blocks expected to be reused
  - Access stream has predictability - controlled access mechanism reduces tag lookups

How many blocks to buffer?

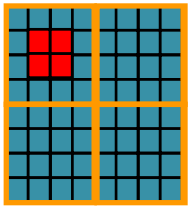
- Need to buffer up to 4 blocks



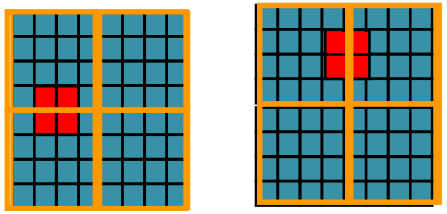
- A buffer is a set of 4x4 registers, each 32 bit
- Texture Buffer Array is a group of 4 such buffers

Texture Lookup

- Case 1:
  - Lookup (block0)
    - Get the 4 texels from the block using offsets
    - SAVING: 3 LOOKUPS

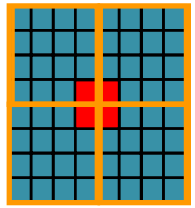


- Cases 2 & 3:
  - Lookup (block 0)
    - Get texel0 and texel1 from this block
  - Lookup (block 2)
    - Get texel2 and texel3 from this block
  - SAVING: 2 LOOKUPS



Contd..

- Case 4:
  - Lookup all 4 blocks and get the texels from the respective blocks using offsets

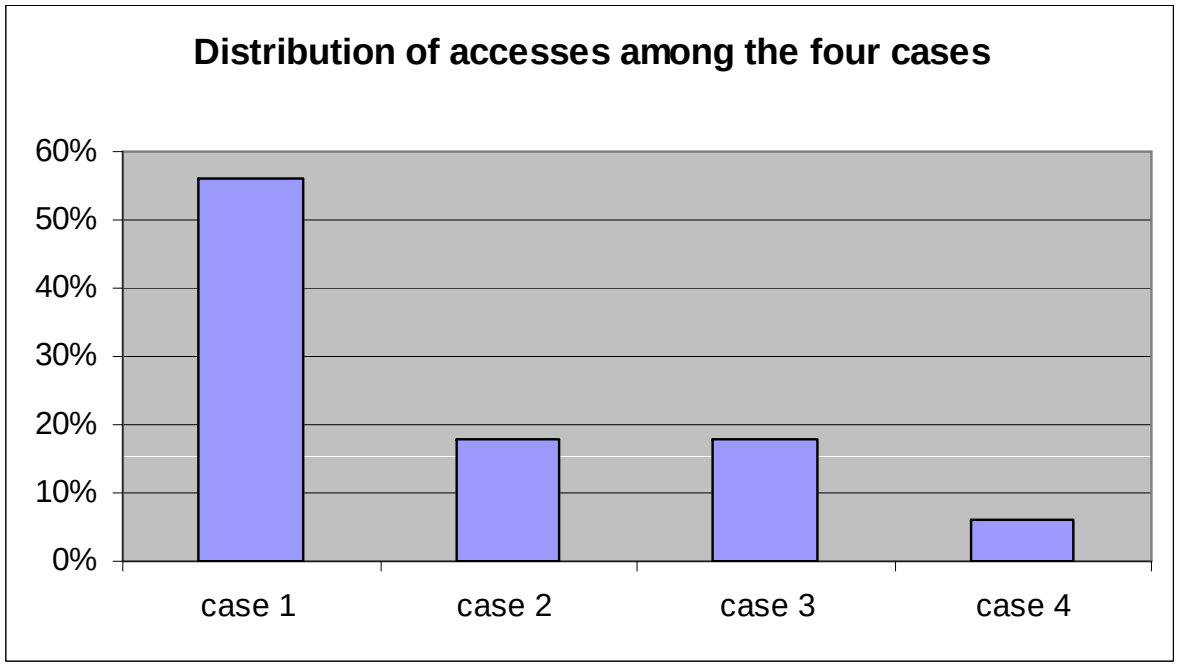


Power Savings from:

Reduced Tag lookups

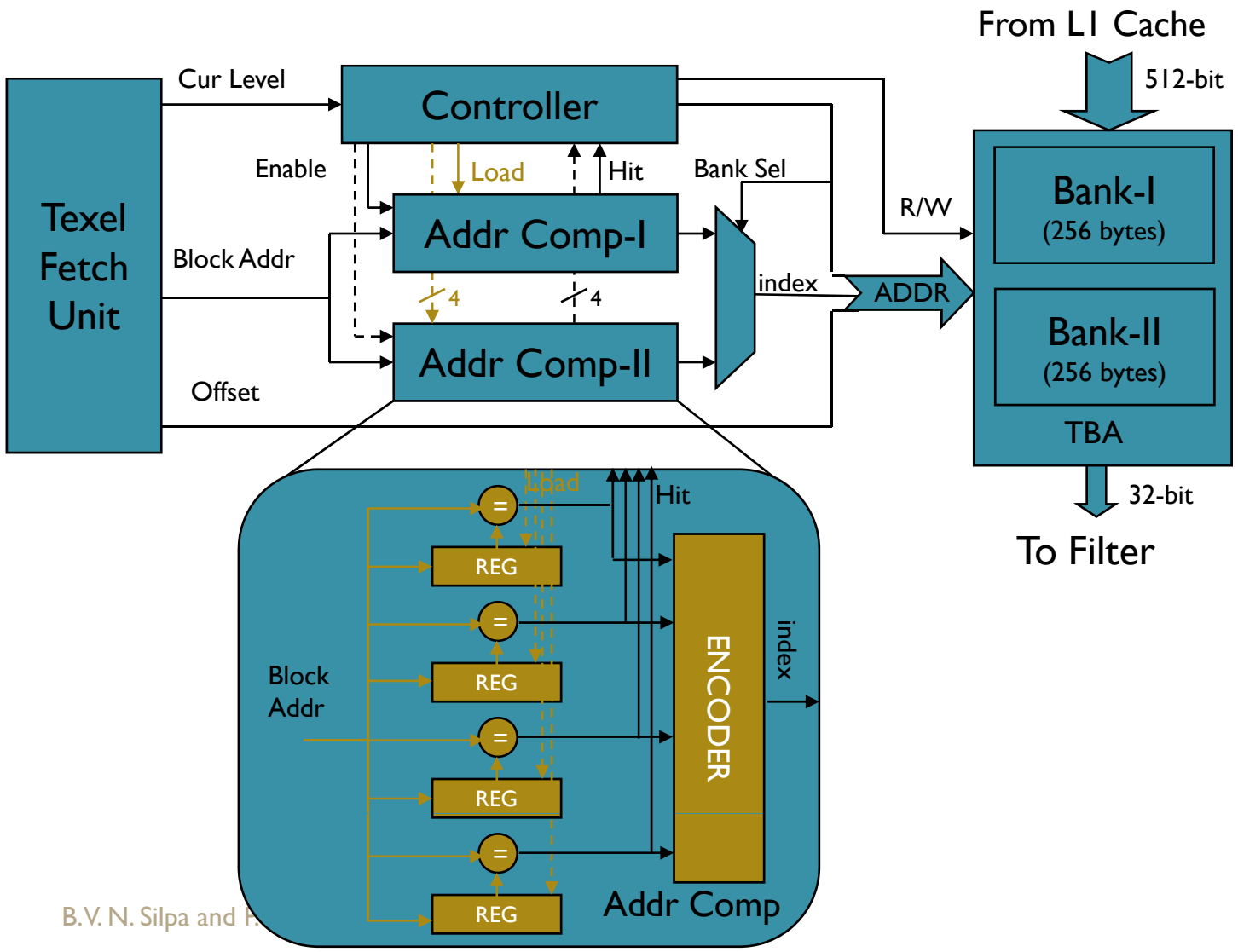
Smaller buffer than cache

Distribution of access among various cases

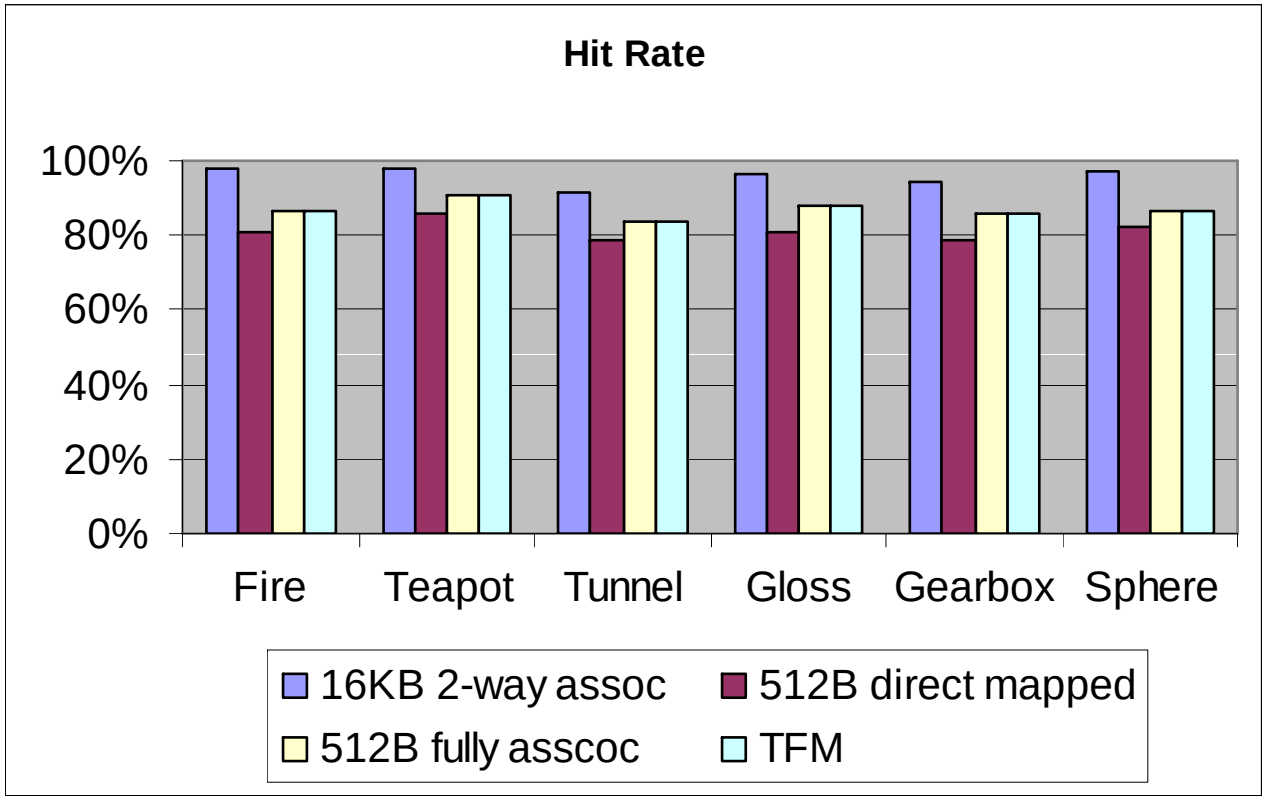


Number of comparisons per access is 1.38 instead of 4

Architecture of Texture Filter Memory

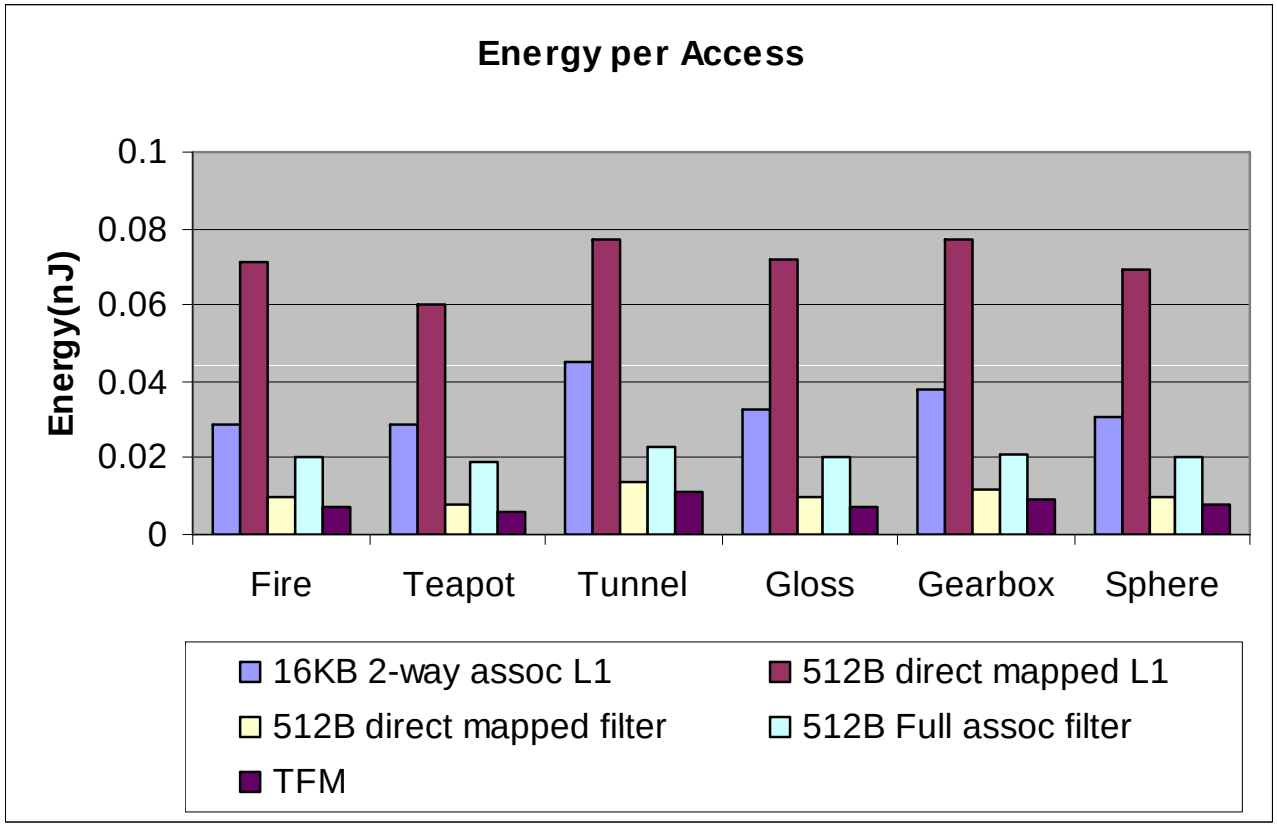


Hit Rate into TFM



TFM gives 4.5% better hit rate than a direct mapped filter of the same size

Energy per Access



TFM consumes 75% lesser energy than the conventional Texture cache

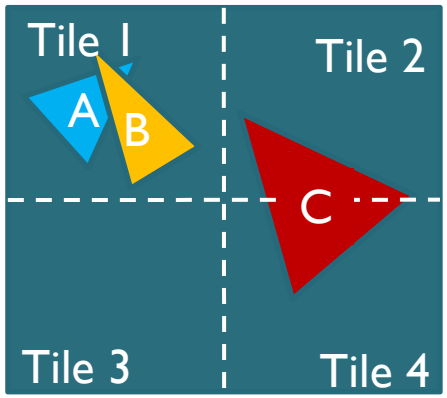
Texture Filter Memory Summary

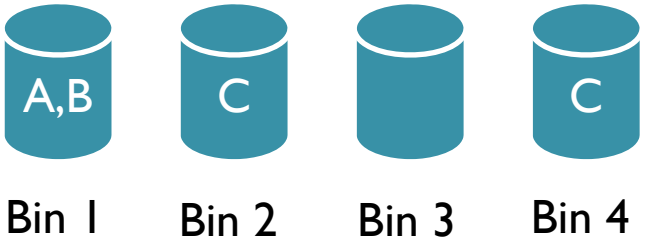
- In addition to high spatial locality, texture mapping access pattern also has predictability
- Replaced high energy cache lookups with low energy register buffer reads
- TFM consumes ~75% lesser energy than conventional texture mapping system
- Overheads:
  - TFM access 4x faster than cache access
  - 0.48% area overhead over texture cache subsystem

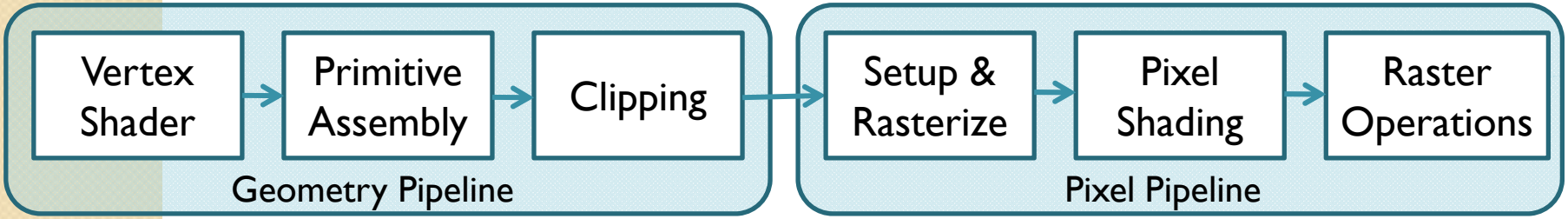
DYNAMIC VOLTAGE AND FREQUENCY SCALING (DVFS)

[CODES+ISSS'10]

Tiled Graphics Rendering



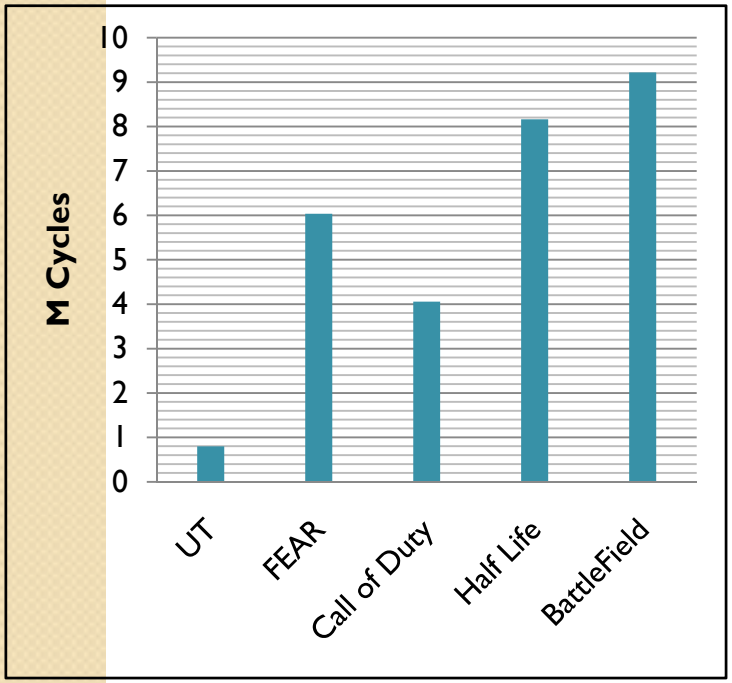


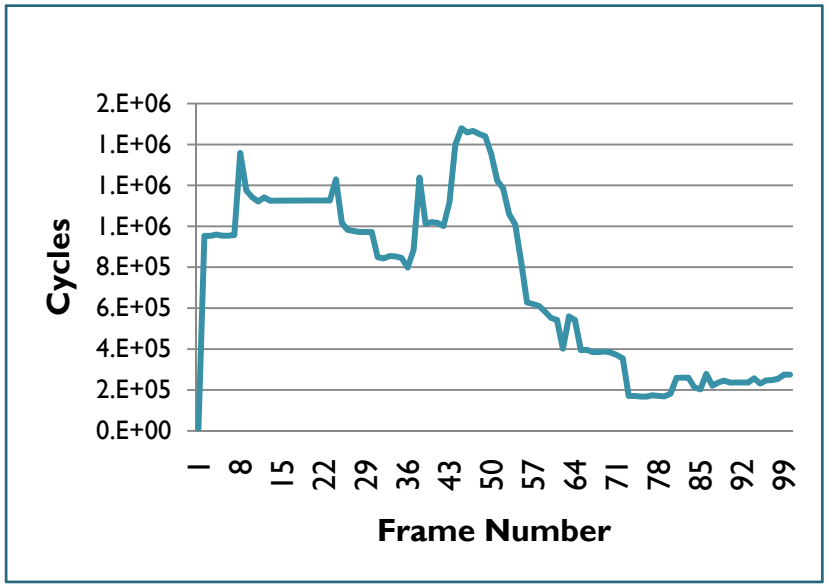


B.V. N. Silpa and P. R. Panda, 2011

Workload of games

Different games have significant but gradual workload variation within a game





Spatial Correlation in frames

Continuity of motion leads to frame level spatial correlation, resulting in slow workload variation

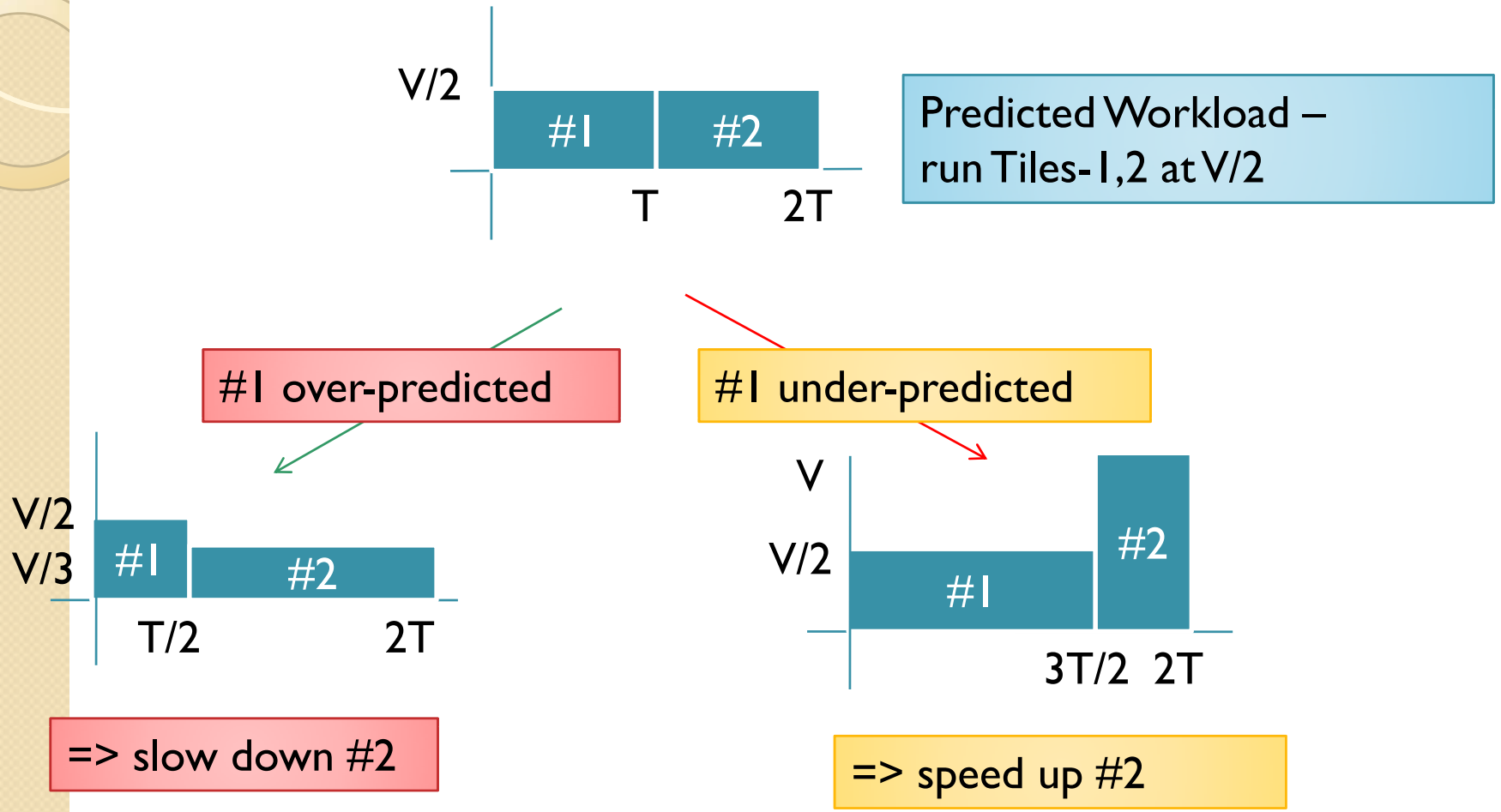
B.V. N. Silpa and P. R. Panda, 2011

Temporal Correlation of Tile Workload

- Many tiles are correlated, even if workloads of consecutive frames differ
  - 80% tiles within 10% diff

B.V. N. Silpa and P. R. Panda, 2011

Dynamic Voltage and Frequency Scaling



Continuously track and take corrective action after rendering each tile

Frame Rank (R_G, R_p, R_t, R_r)

- Vertex processing workload of a primitive of V vertices using a Shader Nv instructions long
  - Shader workload ~ V * Nv
  - Clipping and Binning ~ V

$$R_g = \sum_{Batches} VertexCount \times VertexShaderLength + PrimitiveCount$$

- Pixel shading workload
  - Number of pixels per primitive ~ Area of bounding box of the primitive

$$R_p = \sum_{Batches} \sum_{Primitives} PrimitiveArea \times PixelShaderLength$$

Frame Rank (R_G, R_p, R_t, R_r)

- Texture mapping workload
  - Texture footprint – number of texels to be filtered per pixel

$$R_t = \sum_{Batches} \sum_{Primitives} PrimitiveArea \times TextureCount \times TextureFootPrint$$

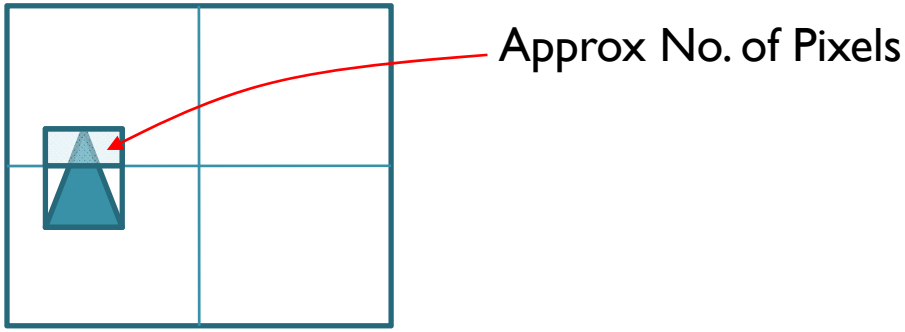
- Raster operations workload
  - Each raster operation results in a read and write to frame buffer

$$R_r = \sum_{Batches} \sum_{Primitives} 2 \times PrimitiveArea \times RasterOps$$

B.V. N. Silpa and P. R. Panda, 2011

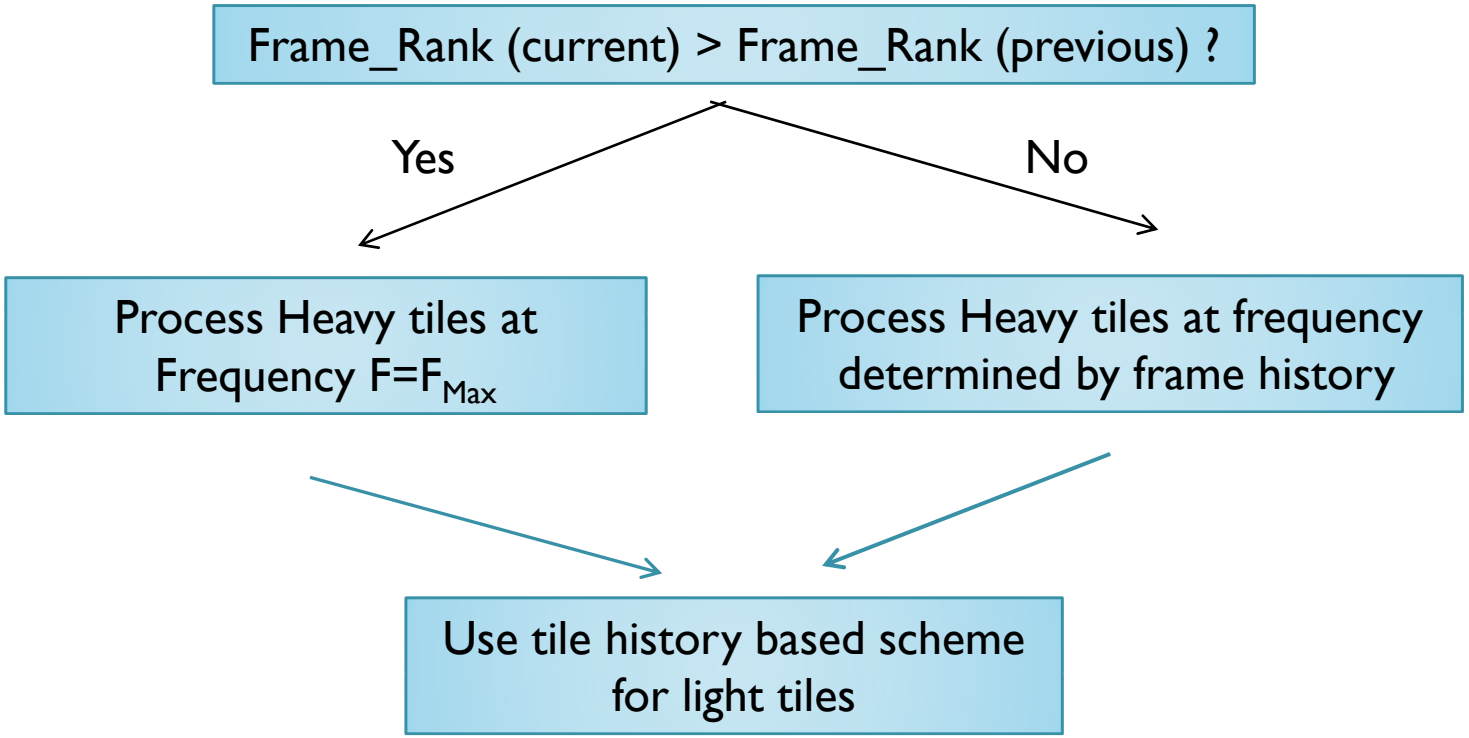
Tile Rank (T_p, T_T, T_r)

- Tile rank computation is similar to frame rank computation
- Pixel count is computed as overlap area of the bounding box and the tile.



Rank Based DVFS Scheme

- Divide the tiles into set of Heavy tiles (Tile rank in current frame greater than its rank in previous frame) and Light tiles.



Tile Rank Based DVFS Summary

- Tile Rank Based DVFS gives 75% better performance than history based scheme
- Energy/FrameRate minimum for Tile Rank based DVFS scheme
- Overheads
  - < 0.01% computation
  - < 0.01% storage

Future Work

- Extension to multi-core GPUs
- Other stages of the graphics pipeline

Thank You!