SaarCOR-A Hardware Architecture for Ray Tracing

2008. 7. 22. 장병준

SaarCOR Chip

- Components
  - RGS
    - Ray generation and shading unit
  - RTC
    - Ray tracing unit
  - RTC-MI
    - Memory access manage unit

Component : RGS

- Sub units
  - ***master*** : determining which eye ray will be rendered next
  - ***slave*** : receiving the coordinates of a pixel from the ***master*** and managing this ray until it is fully rendered
  - ***MemInt*** : the unified memory interface, handling all memory accesses

Component : RTC

- Trace rays through the BSP acceleration structure and intersects rays with triangles found in the leaf nodes

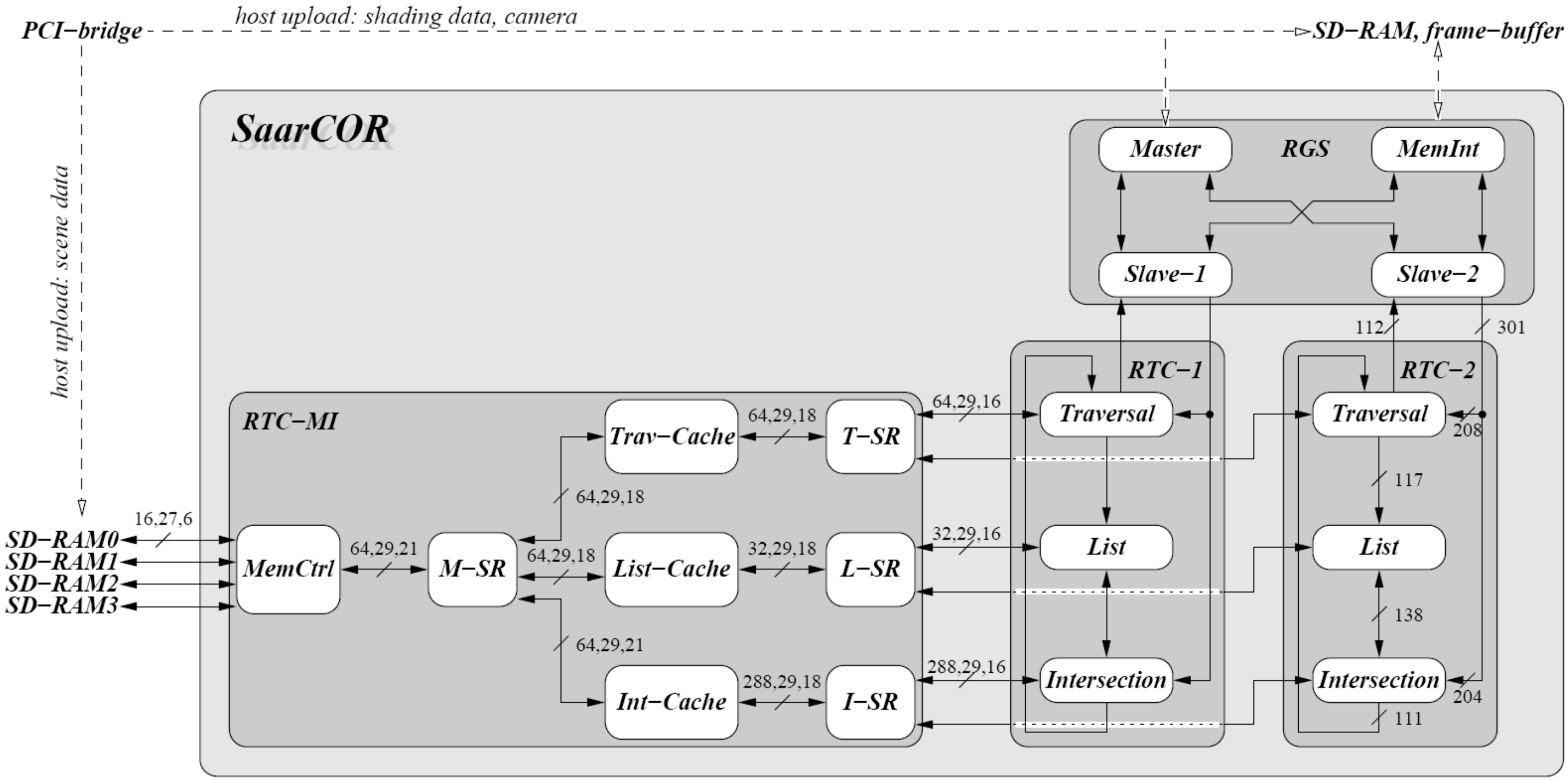
Component : RTC

- Sub units
  - ***traversal unit*** : receives rays from the RGS ,traces them until it locates a BSP node containing a list of triangles and forwards the list addresses to ***list unit***
  - ***list unit*** : fetching the addresses of the triangles and sending their addresses to the ***intersection unit***
  - ***Intersection unit*** : fetches the triangle data and performs the intersection computation

Component : RTC-MI

- Handles memory requests for all RTC cores
- Simple routing units implementing a simple but efficient routing scheme

SaarCOR hardware model



RGS implementation issues

- Phong-like shading with bilinear texture filtering
- Cost per ray is 50FP adds and 70FP muls including address calculation for texture reads
- Shading is decoupled from visibility computations and allows the architecture to be tuned for specific target markets

RTC implementation issues

- Packet of rays
  - If packet of rays are coherent and visit roughly the same items. Then traversed BSP nodes need to be fetched only once to be used for every ray in a packet.
  - Large packets also increase the overhead as they cause more rays to traverse they would not traverse.
  - Groups of 64 rays are a good compromise between bandwidth requirements, additional overhead.

RTC implementation issues

- Bit vector with each packet
  - Whether the ray is active in the current branch of the BSP tree.
  - Efficiently operate on only a subset of rays in a packet and dramatically reduces the overhead.
  - Updating and evaluating the bit vector is almost negligible.

RTC implementation issues

- Estimation of the cost
  - Traversal operations(*trav-op*)
    - 64bits of data, 3FP adds, 1FP mul
  - Intersection operations(*int-op*)
    - 288bits of data, 12FP adds, 13FP muls
  - Assume 1add equal 1sub, 1div equal 3muls

RTC implementation issues

- Load balance between the traversal and intersection
  - Varying the depth of the BSP tree
    - A ratio of 4 *trav-op* to 1 *int-op* is well suited for most scenes
- Asynchronously traversal
  - Reducing the overhead introduced by a group of idle rays to a single cycle
- Multi-threading

RTC-MI implementation issues

- Round-robin multiplexer for submitting memory requests and a labeled broadcast to return data.
- Each unit allows for several outstanding memory requests