The ray tracing algorithm is well-known for its ability to generate photo-realistic rendering effects. Recent years have witnessed a renewed momentum in pushing it to real-time for better user experience. Today the construction of acceleration structures, e.g., kd-tree, has become the bottleneck of ray tracing. A dedicated hardware architecture, FastTree, was proposed for kd-tree construction by adopting a fully parallel construction algorithm. FastTree was validated by an FPGA prototype and evaluated as an ASIC implementation. Experiment result shows FastTree outperforms existing hardware construction engines by a factor of nearly 4X at a similar area and power budget.

## Hardware Implementation

### Path Compression Algorithm

Ray tracing technique emulates the vision formation process by shooting rays from observer to a 3-D scene [3][8]. A kd-tree is built by hierarchically partitioning the space according to the spatial distribution of the graphic primitives.

A. Previous construction algorithms:
- top-down approach: insufficient parallelism in upper levels
- Bottom parallel construction algorithm by Li et al. (Figure 2) [1]:
  - Starts from a complete binary tree
  - Concurrently working on all leaf nodes in a bottom-up manner
  - Compress tree by removing redundant partitions
  - Uses Morton code to identify spatial partitions
  - Leaf Node Generation
  - Internal Node Generation
  - Path Compression

B. ASIC Evaluation
- TSMC 65nm process, Synopsys DC
- Clock frequency up to 720 MHz at voltage of 1 V
- Total area 5.4mm², total power 3.9 W

## Module Implementation

### A. Scene Bound Calculation Unit (SBCU)
- Takes eight 32-bit floating point values
- Selects the largest or smallest value
- Two similar SBCUs compute lower and upper bounds in parallel

### B. Prefix Sum Unit (PSU)
- Chops a long array into shorter sections
- Performs prefix sum on each section in parallel and then merges partial results
- Computing flow consists of two pipeline stages, up- and down-sweep (Figure 4)
- Four PSUs operating in parallel

### C. Morton Code Generation Unit (MCGU)
- Generates Morton code and copy triples
- Takes nine 32-bit coordinates and computes the bounding box of the triangle
- Four MCGUs operate in parallel

### D. Radius Sort Unit (RSU)
- Based on the parallel radix sort algorithm
- PSUs are reused and RSU serves as control logic

### E. Leaf Node Generate Unit (LNGU)
- Contains 32 pairs of 2-input integer comparator and 2-to-1MUX, in parallel
- Contains 32 32-bit integer subtraction units

### F. Internal Node Generate Unit (INGU) and Bit Encoding Unit (BEU)
- INGU computes left-most different bit of leaves
- INGU also marks whether a node appears in the final kd-tree
- BEU encodes the marking bits

### G. Path Compression Unit (PCU)
- Takes a node and calculates its parent
- Also checks if its direct parent is essential (appears in the final tree).
- If not, continues to check the ancestors upwards until reaching an essential node

### H. On-chip Cache
- 16KB on-chip cache, 16-way set associative
- LRU update policy

## References