**Research Goals**

- Become associated with the IBM Cell BE Processor in a Playstation 3
- Get experience with large multi-core Cell projects in my tracer, game engines etc.
- Assist the port of Sketch Synthesis to multi-core Silicon
- Retire as rich lungs in the land of multi-core programming

**Cell Architecture**

- **Power PC Unit (PPU)**: 3.2GHz
- **VMX 52 instruction set**
- **Control of eight arithmetic-logic processing units (SPUs)**
- **Main 32/64-bit L2 Cache**
- **Synaptic Processing Unit (SPU)**: 3.2GHz
- **Optimized for data streaming**
- **128 bit SIMD instruction set**
- **256KB local store**
- **128x28 256-bit store file**

**DMA (Direct Memory Access)**

- Controlled by the SPU Memory Flow Controller unit
- Transfers data directly to and from the PPU L2 Cache and the SPU local store
- Transfers data from SPU to SPU

**Spu_mul(vec2,vec3);**

**Spu_sub(vec2,vec1);**

**Spu_add(vec1,vec2);**

**vector int int_vec={1,2,3,4};**

**mfc_read_tag_status_all();**

**mfc_get(&addr_of_local_data, addr_of_remote_data, data_size, etc.);**

**Bandwidth of the Bus 96B/cycle.**

**Request queuing for 8 requests**

**DMA_bandwidth on the SPU is 128B/cycle.**

**No other sized transfers are supported.**

**Transfers data from 1 byte to 16KB by powers of 2 bytes.**

**Transfers data directly to and from the Controller by the SPU MFC (Memory Flow Controller) unit.**

**128x128 bit register file**

**256KB local store**

**Optimized for data streaming**

**Synergistic Processing Unit (SPU) 3.2Ghz**

**Power PC Unit (PPU) 3.2Ghz**

**char_vec={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};**

**MFC_TAG, ...);**

**Related Work and Results**

**Sketching Goes Concurrent: Parallel Data Structures Paper Accepted at PLDI ’08**

**MS Intel Lab For UC Berkeley Parallel Research !!**

**Mult-Core Software Reengineering and Parallel Software Synthesis (Sketch)**

**Apollo Ellis, Rastislav Bodik UC Berkeley**

**PS3DOOM**

- We wish to recreate on a multi-core platform the classic Doom game engine created at Id Software.
- Lead programmer John Carmack leaves us with some significant challenges in achieving a fast software ray casting engine on the order of Doom.
- The core technologies at our grasp came through ingenious exploration of the infamous Linear Dome source release by John Carmack and are centered around traversing of walls, doors, ceiling, and source particles.
- On the PlayStation 3 every instruction possible in our implementation will be SIMD hopefully allowing us to achieve a 4-5 time speed up over most codes.

**This is not a source port. This is my code.**

**MIT Blue Steel Ray Tracer**

- **Assigns each core one row of pixels**
- **On small scale scenes with few primitives: 15 or less**
- **Rounding is fairly easy**
- **Blue Steel Pang was created**
- **On slightly larger scale scenes: 50 to 100 primitives**
- **Blue Steel rendering can’t keep up with dynamic demands**
- **Roughly 5 ps or less**
- **Similar to single threaded ray tracing**

**Ray Tracing with The BSP Tree**

- **Cut the space with a single plane, this is our root node.**
- **Each side of the plane is child node cut with its own plane.**
- **Recursively do this or get back to a few primitives.**
- **Store the structure and traverse it with rays we need to find.**
- **Now it is possible to reduce the work done on a larger scene down to that which Blue Steel can work well with.**

**Note:** With Fast Ray-Triangle Intersection algorithms such as Badouel’s or Ieg’s Wald’s and Large Size Ray Packet Tracing (XSP) open core and have been achieved in addition to the BSP acceleration.