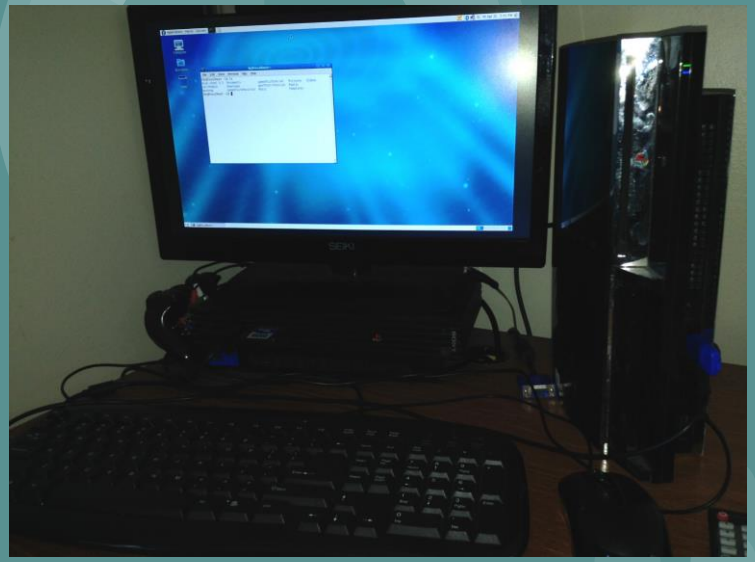
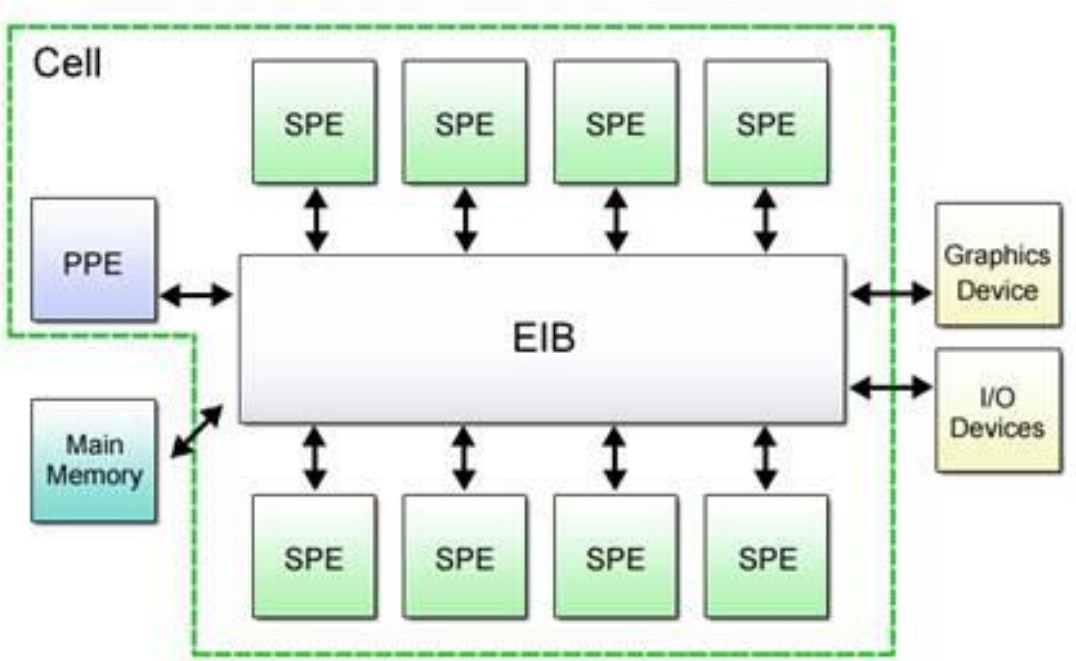
William Blair
CS 518/523: Parallel Programming

Parallel Programming on the Sony PlayStation® 3

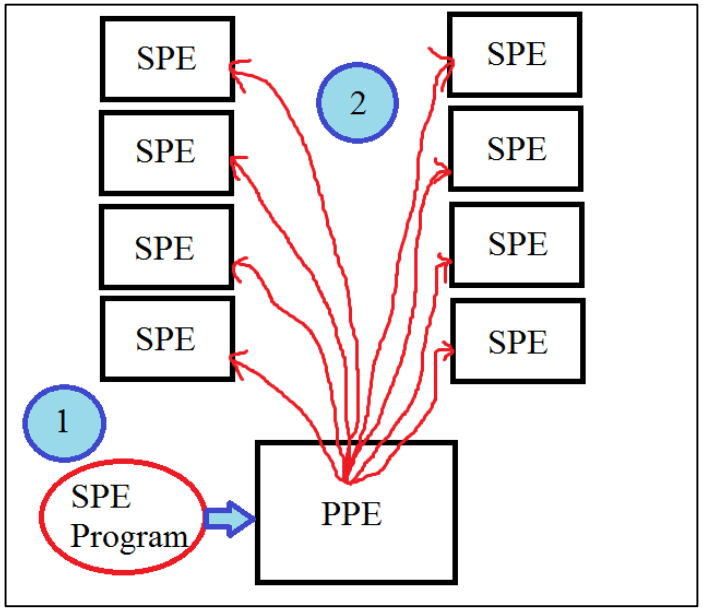
The IBM Cell B.E. Processor



The Cell Broadband PowerPC Engine



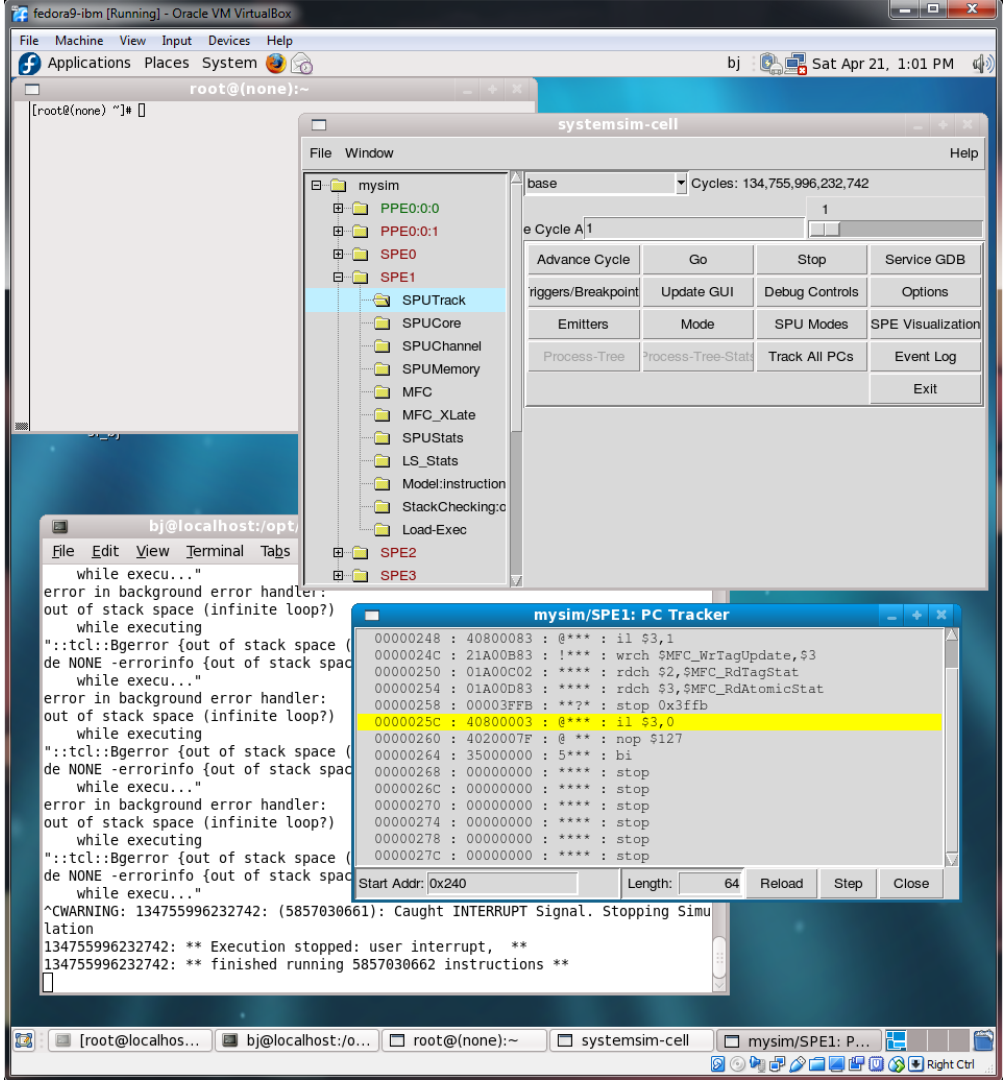
Cell internal hardware layout [1]



Basic structure of a Cell program

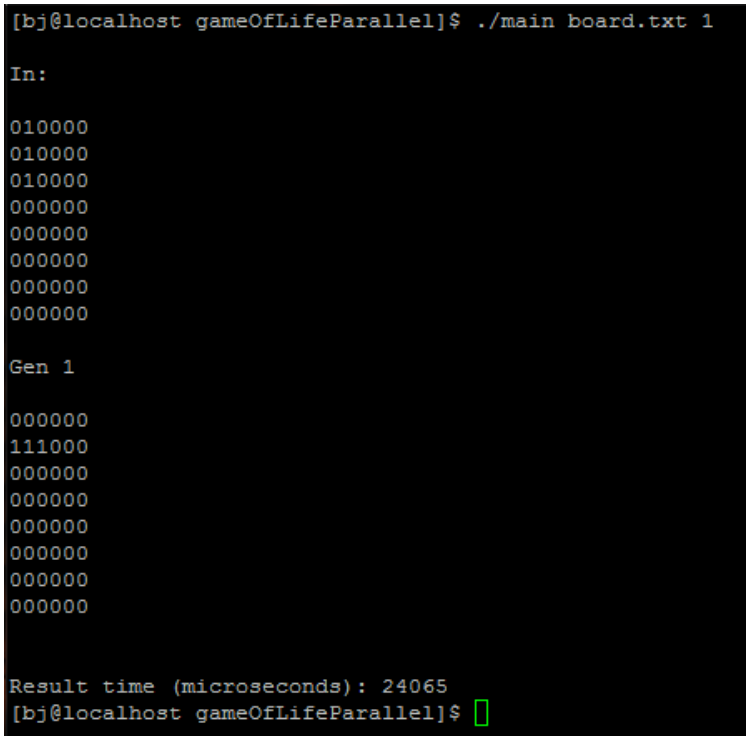
IBM Cell SDK 3.1

- IBM's Official SDK to program the Cell Processor
  - ppu-gcc, ppu-g++ to compile PPE programs (main)
  - spu-gcc, spu-g++ to compile SPU programs
    - Optional Fortran support
- Also includes a Cell simulator to work on your programs locally and analyze the program (i.e. monitor states, bottlenecks, load) (image right)
- Runs on Fedora 9 or YellowDog linux 5.0 and above, which run on both PowerPC (PS3) and PC
- Linux support dropped by Sony after PS3 firmware 3.21 on April 1st, 2010 [2]

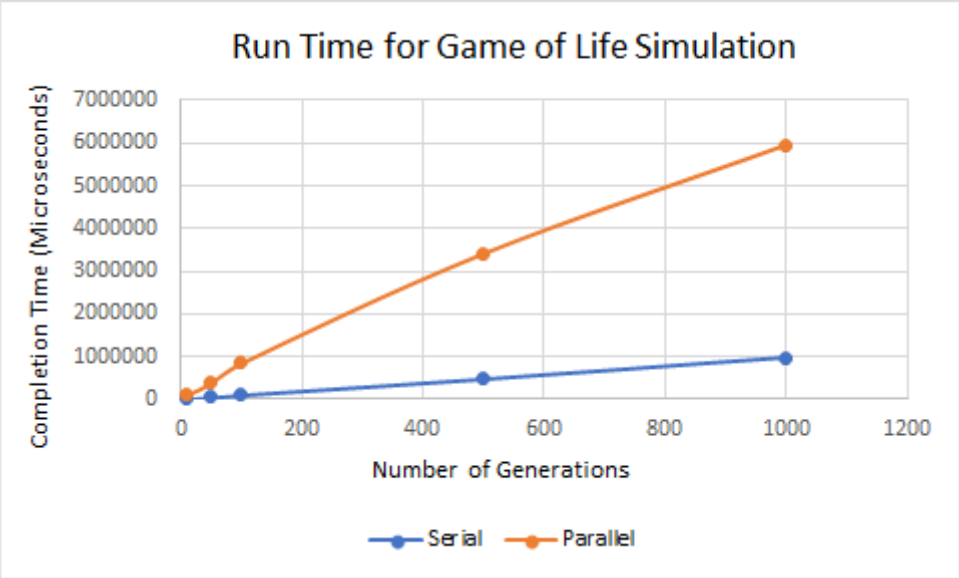


Conway’s Game of Life

- Not really a ‘game’ but a simulation of living, biological cells
- Set of rules that runs for a number of generations [3]:
  - Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
  - Any live cell with two or three live neighbours lives on to the next generation.
  - Any live cell with more than three live neighbours dies, as if by overpopulation.
  - Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- Written in C++, parallelized for the Cell B.E. Processor, and a serial version for comparison



Results



# Generations	Serial Time	Parallel Time
10	10,957 µs	89,483 µs
50	47,457 µs	375,271 µs
100	101,292 µs	841,760 µs
500	468,587 µs	3,406,050 µs
1000	957,655 µs	5,948,228 µs

Run time comparison of the Serial and Parallel implementations of Game Of Life. Calculations were performed on a grid of size 8x30 (so 1 row for each SPE thread in the parallel version) with the same layout each time.

Analysis/Future Work

- Bottlenecks:
  - Need bigger board for an improved compute/io ratio
  - Synchronization: each generation relies on the results of the previous generation, so each thread has to wait for each other thread to finish before computing the next generation
  - Each row requires the previous and next row to check for live neighbors -> sending 3x amount of data per thread
- Future work:
  - Use different method of data communication between PPE/SPEs
    - Max size to the amount of data you can send over the bus
      - The previous chart used the largest board size possible for the given amount of threads -> not big enough
    - Technique called ‘Mailbox’

References

1. https://mirrors.edge.kernel.org/pub/linux/kernel/people/geoff/cell/ps3-linux-docs/CellProgrammingTutorial/BasicsOfCellArchitecture.html
2. https://en.wikipedia.org/wiki/OtherOS
3. https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Project Web Page

https://web.cs.sunyit.edu/~blairw/CS518/final_proj/index.html