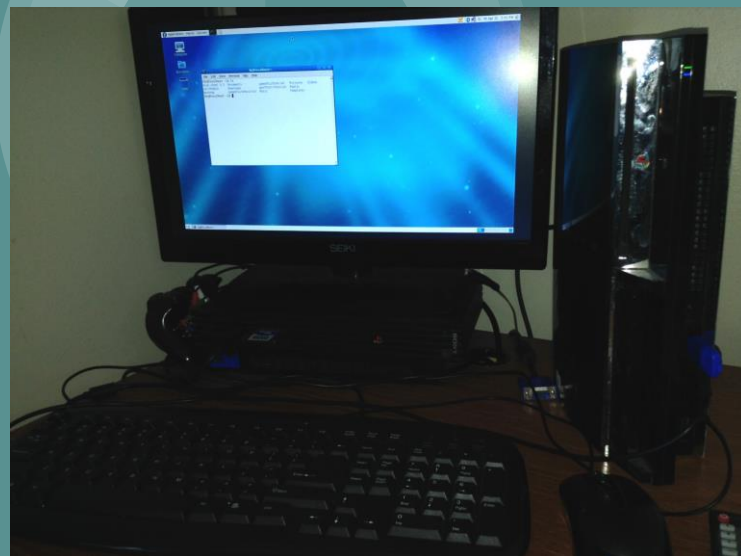


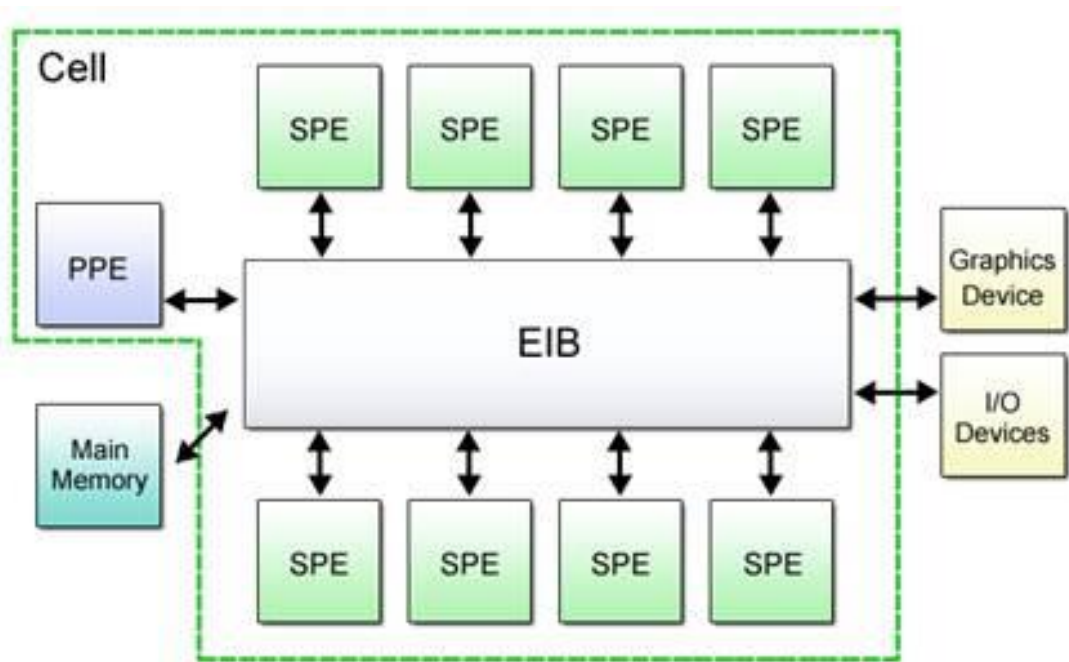
William Blair
CS 518/523: Parallel Programming

Parallel Programming on the Sony PlayStation[®] 3

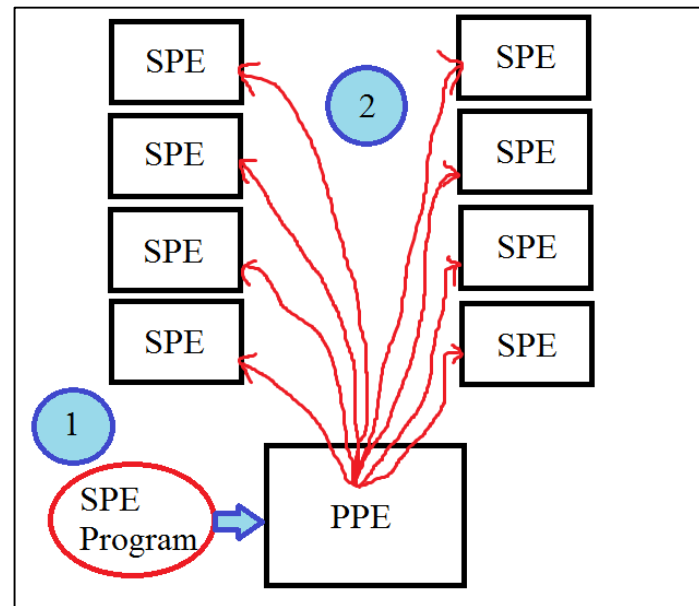
The IBM Cell B.E. Processor



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]

fedora9-ibm [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places System

bj Sat Apr 21, 1:01 PM

root@(none):~

root@(none) ~

File Window Help

mysim

- mysim
 - PPE0:0:0
 - PPE0:0:1
 - SPE0
 - SPE1
 - SPUTrack
 - SPUCore
 - SPUCchannel
 - SPUMemory
 - MFC
 - MFC_XLate
 - SPUStats
 - LS_Stats
 - Model:instruction
 - StackChecking:c
 - Load-Exec
 - SPE2
 - SPE3

base Cycles: 134,755,996,232,742

1

e Cycle A1

Advance Cycle	Go	Stop	Service GDB
riggers/Breakpoint	Update GUI	Debug Controls	Options
Emitters	Mode	SPU Modes	SPE Visualization
Process-Tree	Process-Tree-Stat	Track All PCs	Event Log
Exit			

mysim/SPE1: PC Tracker

```
00000248: 40800083: @***: il $3,1
0000024C: 21A00B83: !***: wrch $MFC_MrTagUpdate,$3
00000250: 01A00C02: ****: rdch $2,$MFC_RdTagStat
00000254: 01A00D83: ****: rdch $3,$MFC_RdAtomicStat
00000258: 00003FFB: ***?: stop 0x3ffb
0000025C: 40800003: @***: il $3,0
00000260: 4020007F: @ **: nop $127
00000264: 35000000: 5***: bi
00000268: 00000000: ****: stop
0000026C: 00000000: ****: stop
00000270: 00000000: ****: stop
00000274: 00000000: ****: stop
00000278: 00000000: ****: stop
0000027C: 00000000: ****: stop
```

Start Addr: 0x240 Length: 64 Reload Step Close

```
error in background error handler:
out of stack space (infinite loop?)
while executing
"::tcl::Bgerror {out of stack space (
de NONE -errorinfo {out of stack spac
while execu..."
error in background error handler:
out of stack space (infinite loop?)
while executing
"::tcl::Bgerror {out of stack space (
de NONE -errorinfo {out of stack spac
while execu..."
error in background error handler:
out of stack space (infinite loop?)
while executing
"::tcl::Bgerror {out of stack space (
de NONE -errorinfo {out of stack spac
while execu..."
^WARNING: 134755996232742: (5857030661): Caught INTERRUPT Signal. Stopping Simu
lation
134755996232742: ** Execution stopped: user interrupt, **
134755996232742: ** finished running 5857030662 instructions **
```



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

```
[bj@localhost gameOfLifeParallel]$ ./main board.txt 1

In:

010000
010000
010000
000000
000000
000000
000000
000000
000000

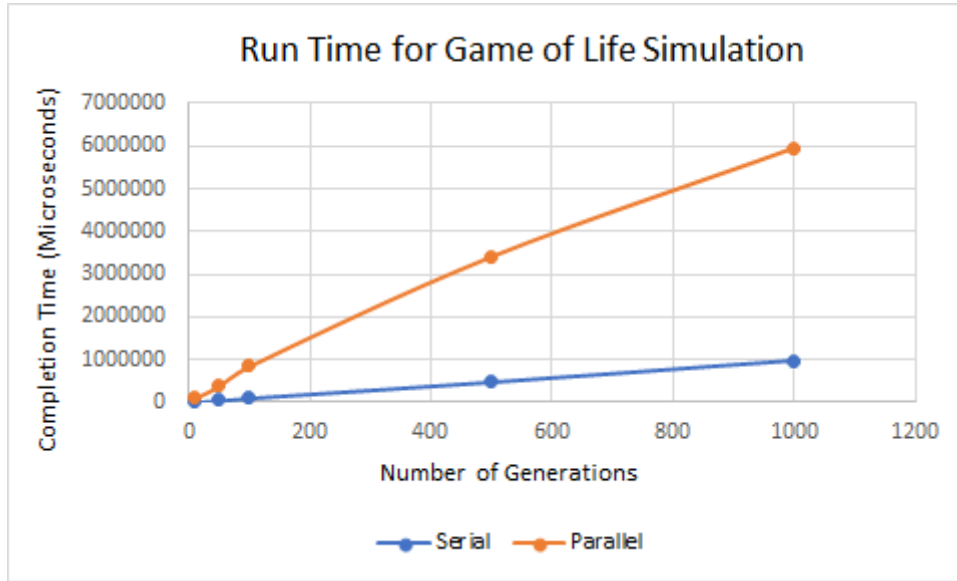
Gen 1

000000
111000
000000
000000
000000
000000
000000
000000
000000

Result time (microseconds): 24065
[bj@localhost gameOfLifeParallel]$
```



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