# Crypto On the Playstation 3™

Neil Costigan

School of Computing, DCU.

neil.costigan@computing.dcu.ie +353.1.700.6916

PhD student / 2nd year of research.

Supervisor : - Dr Michael Scott.

IRCSET funded.

## Playstation 3

- Sony, IBM, Toshiba
- Cell Broadband Engine
- Multipurpose
- Linux
- Development environment

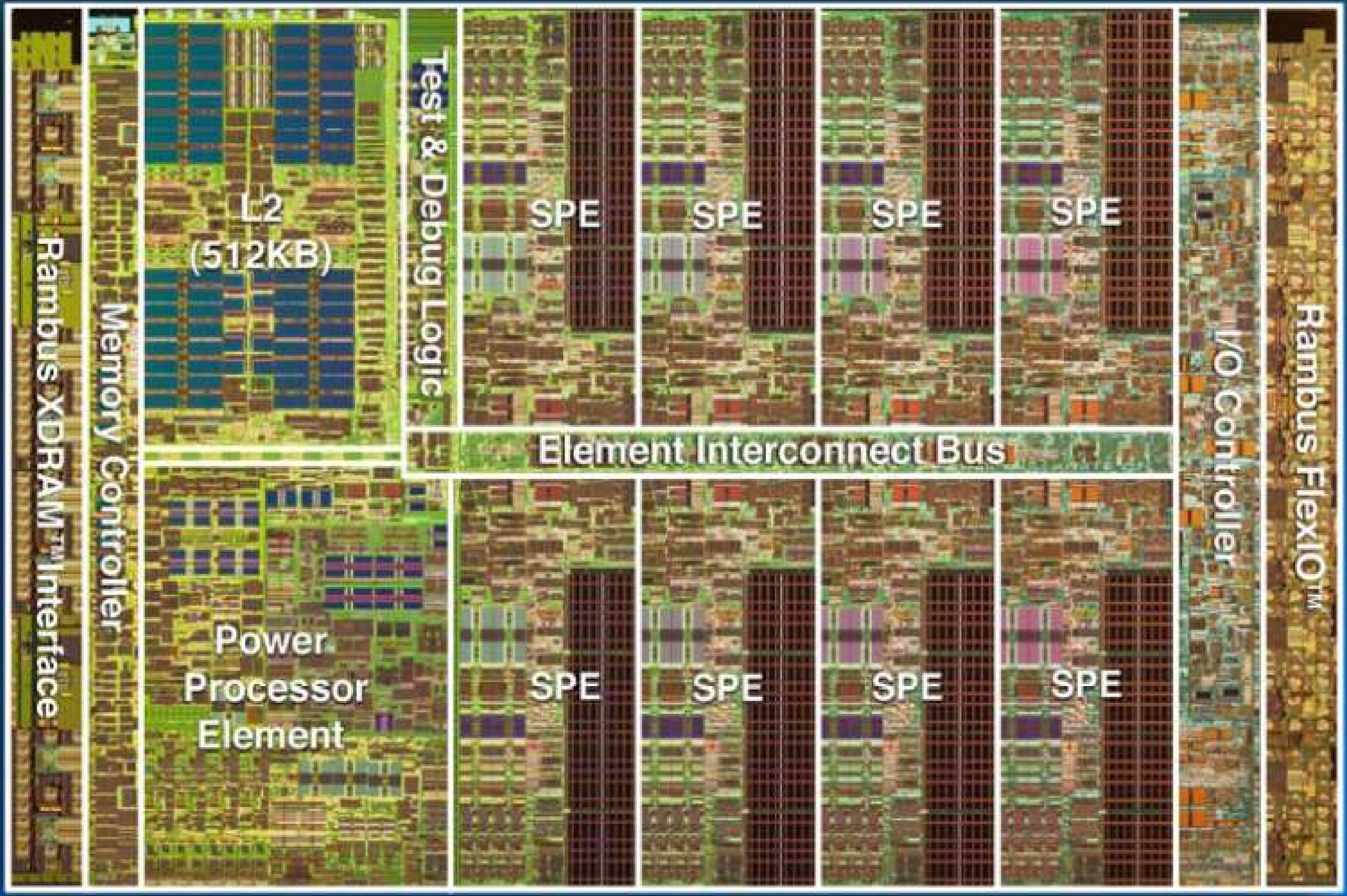
## Cell Broadband Engine

- A 9-way multiprocessor

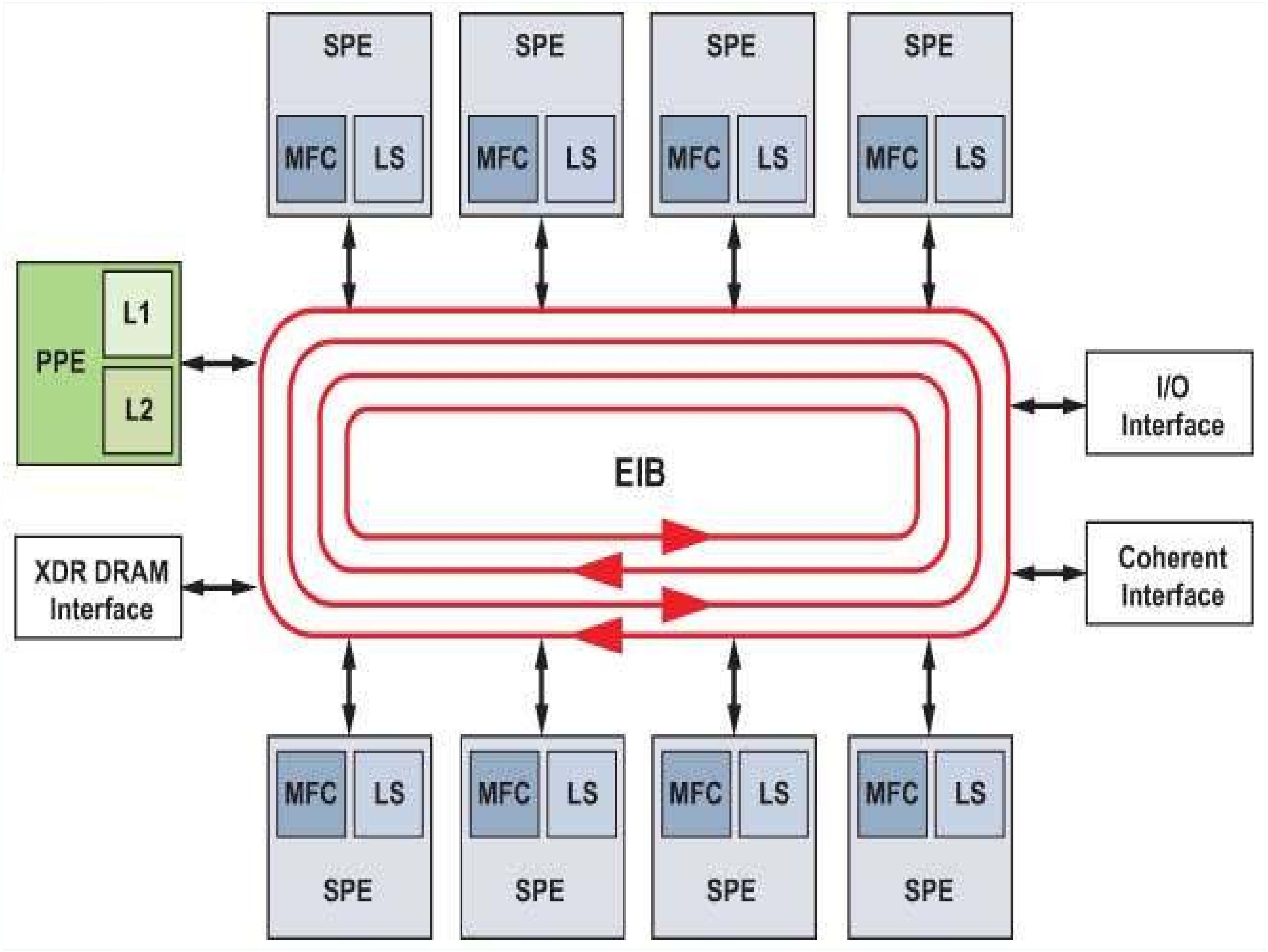
- One main 64-bit PPE processor
  - Power Processor Element, 2 hardware threads
  - Good at control tasks, task switching, OS-level code

- 8 SPE processors
  - Synergistic Processor Element
  - Good at compute-intensive tasks

# Cell Broadband Engine Processor



IBM

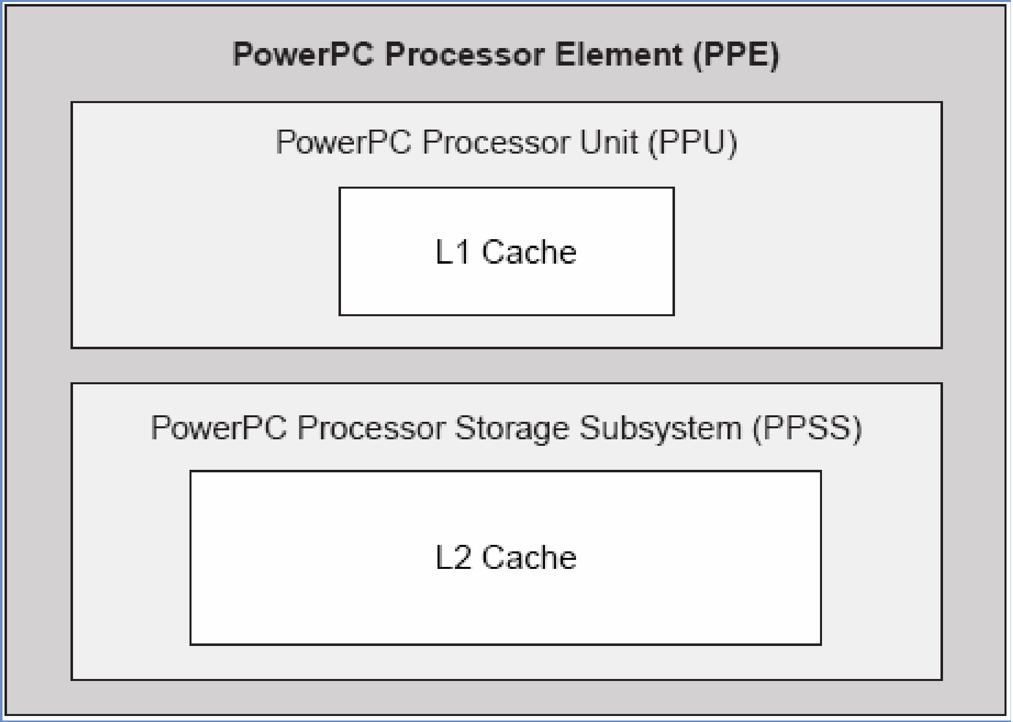

SPE
MFC
LS
SPE
MFC
LS
SPE
MFC
LS
SPE
MFC
LS
PPE
L1
L2
I/O Interface
EIB
XDR DRAM Interface
Coherent Interface
MFC
LS
SPE
MFC
LS
SPE
MFC
LS
SPE
MFC
LS
SPE

## Playstation Vs Cell Blade

- 6 vs 8 SPE
- Cell blade can be configured for dual Cell
- All 16 SPEs visible to each PPE

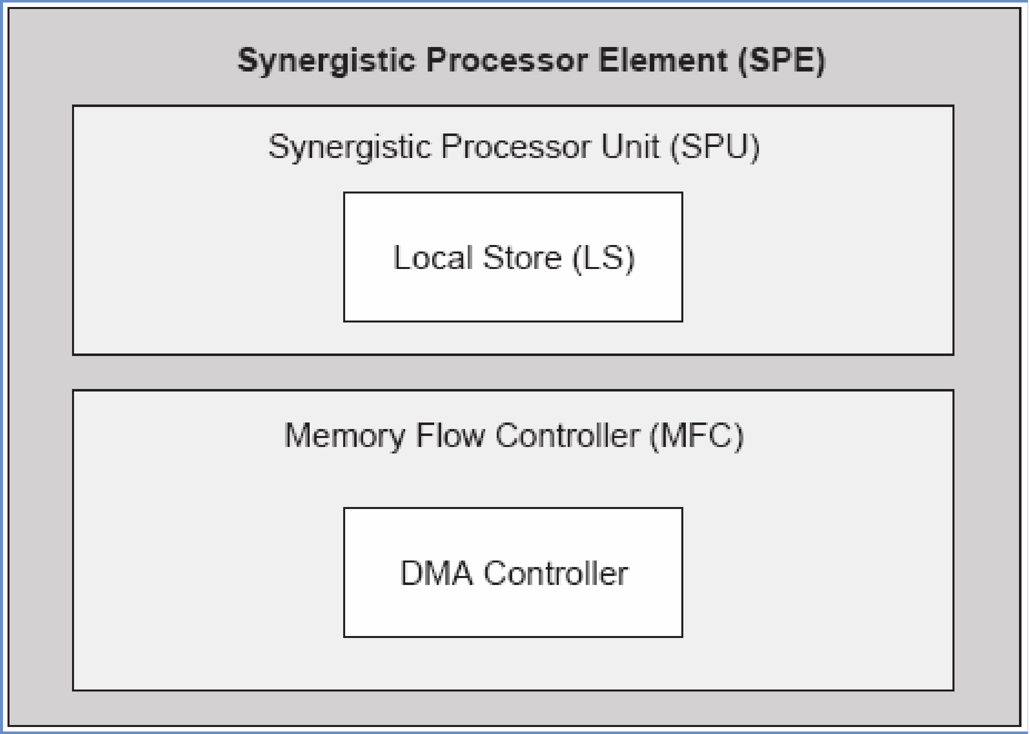
## Power Processor Element

- 64-bit PowerPC Architecture
- RISC core
- Tradition virtual memory subsystem
- Supports Vector/SIMD instruction set
- Runs OS, manages system resources etc


PowerPC Processor Element (PPE)
PowerPC Processor Unit (PPU)
L1 Cache
PowerPC Processor Storage Subsystem (PPSS)
L2 Cache

## Synergistic Processor Element

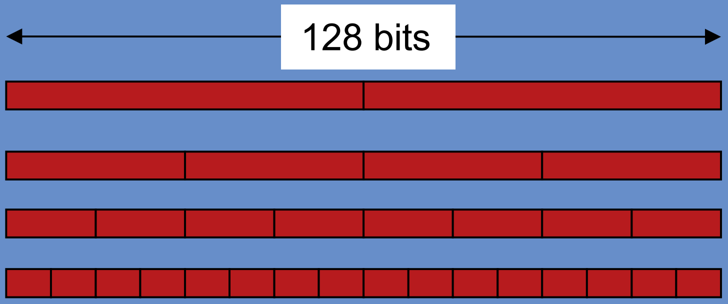
- RISC core
- 256kb local store
- 128-bit, 128 entry register file
- Vector/SIMD
- MFC controls DMAs to/from Local Store over EIB

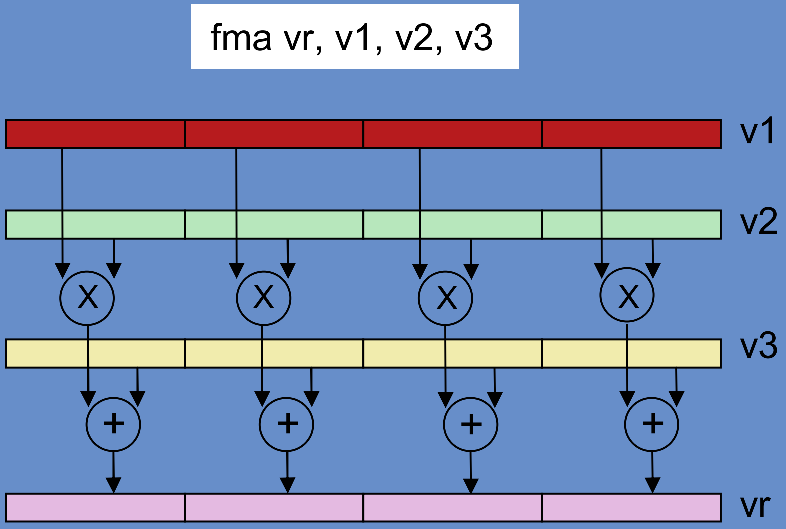

Synergistic Processor Element (SPE)
Synergistic Processor Unit (SPU)
Local Store (LS)
Memory Flow Controller (MFC)
DMA Controller

# So what ?

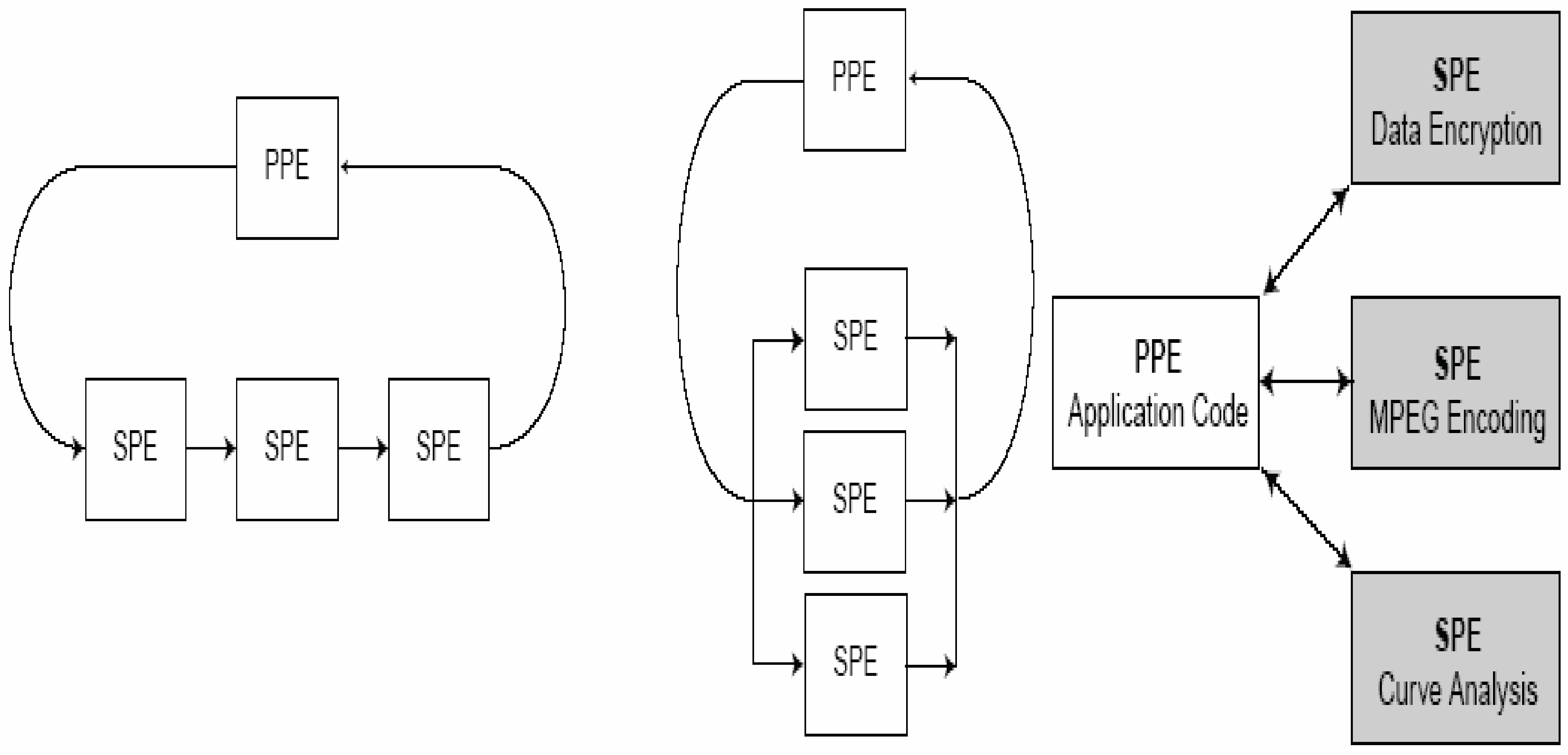
- 128 * 128-bit registers is the key.
- Up to 4 * 32-bit integer operations in ***one*** clock cycle.
- 2 instruction pipelines
- Overlapped DMA

- note
  - it is just a 16x16bit multiplier





## Usage models



Multistage parallel services

## Code : ASM like ADD

Vector Intrinsics :- Assembly-like C

```
vector unsigned int _out_s, _in_a128, _in_a64;
vector unsigned int _sum, _c0, _t0;

_c0 = spu_genc(_in_a128, _in_a64); // generate carry bits
_sum = spu_add(_in_a128, _in_a64); // add
_t0 = spu_slqwbyte(_c0, 4); // shift quadword left 4 bytes
_out_s = spu_add(_sum, _t0); // add in the carry
```

## DMA

- The major programming task is DMA management

- SPE pulls data to LS
- Processes
- Pushes results back to main memory
- Signals completion.

- Overlapping !

## What we are doing…

- Looking at the Cell BE for number crunching
- Using the vector processors to improve crypto performance.
- Currently with native vector multi-precision library. IBM-MPM (SDK 2.0)

## OpenSSL project

- PKI & crypto toolkit.

- Ships with all Unix variants (Linux, MacOSX etc.)
- Apache, MySQL etc.

- Ability to offload some algorithms via 'engine' interface (like a plug-in)

- Has 'speed' tool.

- We choose RSA with CRT
  - Analysis of SSL points to it being ~95% of session overhead.
  - Interesting as it has two independent mod_exp() that can be run in parallel.
  - By counting clock cycles we get **14.87** 4096-bit signs/sec

## Design options

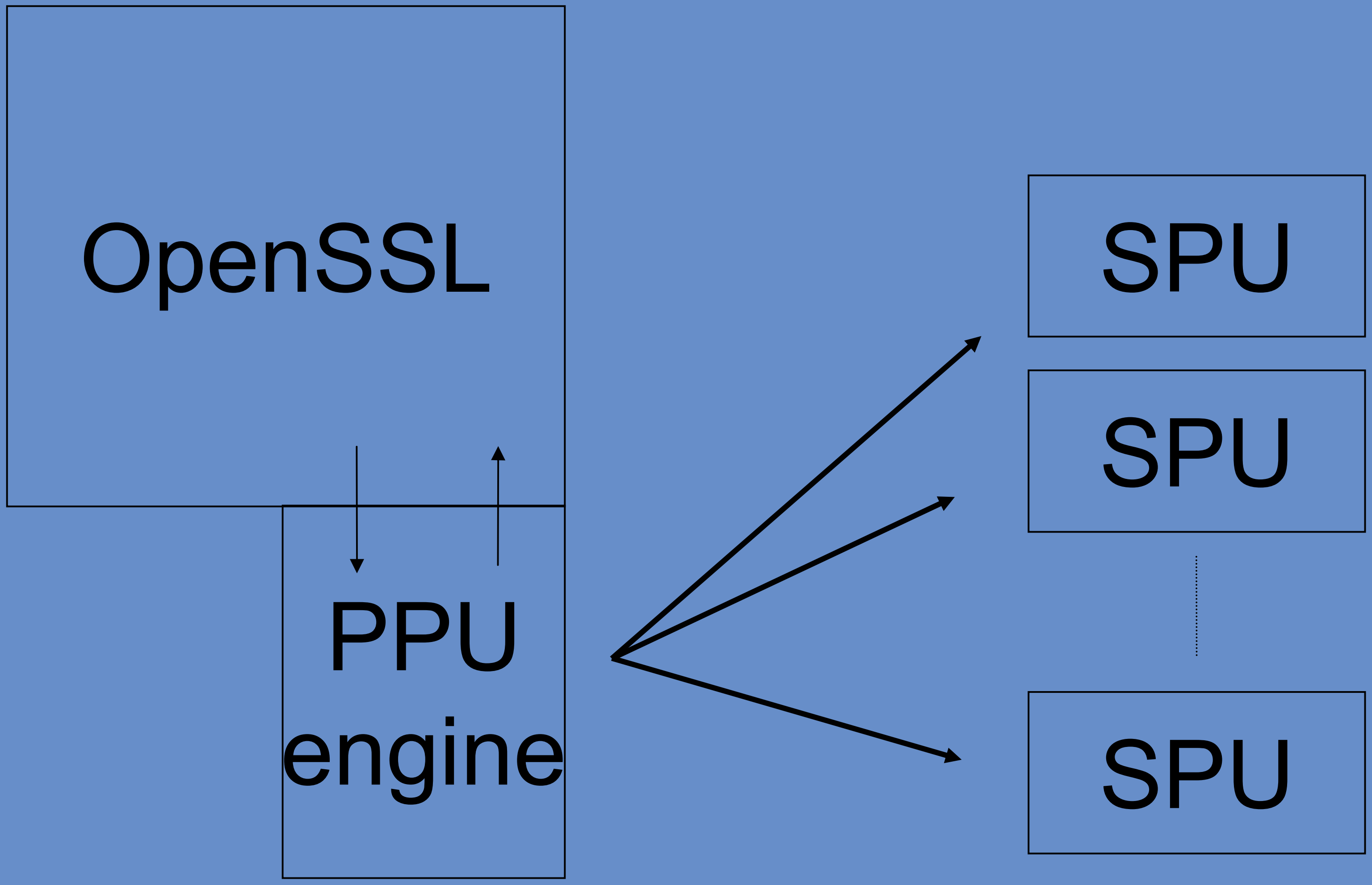
- Two options
  - Design for performance / dedicated accelerator

OR

  - Design for real world / shared system.

Real world is easier to program :-
Create 'thread' when needed but incurs overhead.

## Architecture 1. each SPU does RSA



## RSA : reference / Intel

| RSA | Linux P4 2.5Ghz | MacOSX 2.3Ghz Duo |
|---|---|---|
| key length | sign/sec | sign/sec |
| 1024-bits | 151.1 | 297.4 |
| 2048-bits | 26.7 | 50.4 |
| 4096-bits | 4.4 | 7.6 |

- OpenSSL speed / `-multi 6` / elapsed time
- Linux : Intel 32-bit P4 2.4Ghz
- MacOSX 10.4 : Intel Core 2 Duo 32-bit 2.3Ghz
- Using native OpenSSL with ASM

# RSA : PPU vs. 1 SPU

| RSA | PPU | | 1 SPU | |
|---|---|---|---|---|
| key length | sign | sign/sec | sign | sign/sec |
| 1024-bits | 0.003435s | 291.2 | 0.005655s | 176.8 |
| 2048-bits | 0.017541s | 57.0 | 0.015636s | 64.0 |
| 4096-bits | 0.109793s | 9.1 | 0.070915s | 14.1 |

- OpenSSL speed on PPU vs. 1 SPU
- using IBM-MPM
- 3.2GHz Cell.

## RSA : PPU vs. 6 SPU

| RSA | PPU | | 6 SPU | |
|---|---|---|---|---|
| key length | sign | sign/sec | sign | sign/sec |
| 1024-bits | 0.000724s | 384.5 | 0.001906s | 524.7 |
| 2048-bits | 0.002600s | 71.7 | 0.003033s | 329.7 |
| 4096-bits | 0.089455s | 11.2 | 0.011925s | 83.9 |

- OpenSSL speed on PPU vs. 6 SPUs `-multi 6`
- using IBM-MPM
- 3.2GHz Cell, 6 parallel processes.

# RSA : 2 PPU vs. 16 SPU*

| RSA | 2 PPU | | 16 SPU | |
|---|---|---|---|---|
| key length | sign | sign/sec | sign | sign/sec |
| 1024-bits | 0.001270s | 787.5 | 0.001509s | 745.1 |
| 2048-bits | 0.006805s | 146.9 | 0.001664s | 601.0 |
| 4096-bits | 0.053944s | 22.8 | 0.005762s | 173.6 |

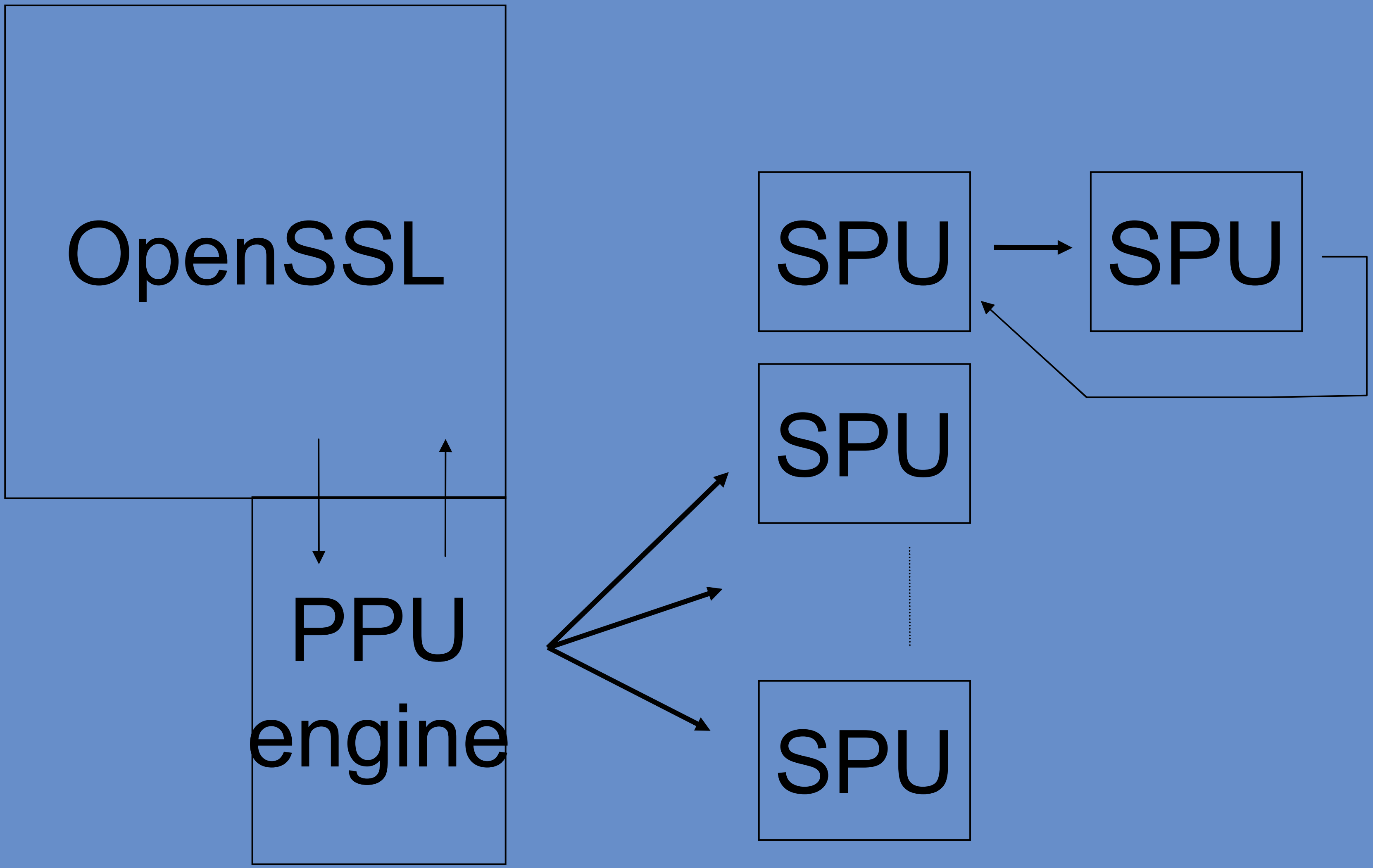
- OpenSSL speed on cell dual PPU with 16 SPUs
- using IBM-MPM
- 3.2GHz Cell, 16 parallel processes. `-multi 16`

- * numbers are from one run by IBM.

## Architecture 2 exploit parallel.

```
INPUT: $p$, $q$, $I0$, $dmq1$, $dmp1$, $iqmp$
OUTPUT: $r0$
       PPU                             SPU 1                     SPU2
$r1$ <- $I0$ mod $q$

THREAD                          $m1$ <-($r1$ ^$dmq1$)mod $q$

$r1$ <- $I0$ mod $p$

THREAD                                                 $r0$ <-($r1$ ^$dmp1$) mod $p$

WAIT

$r0$ <- $r0$ - $m1$
if $r0$ < 0 then
   $r0$ <- $r0$ + $p$
end if
$r1$ <- $r0$ · $iqmp$
$r0$ <- $r1$ mod $p$
$r1$ <- $r0$ · $q$
$r0$ <- $r1$ + $m1$
```

## Architecture 2
## SPU does mod_exp()



## RSA : PPU vs. 2 SPU parallel

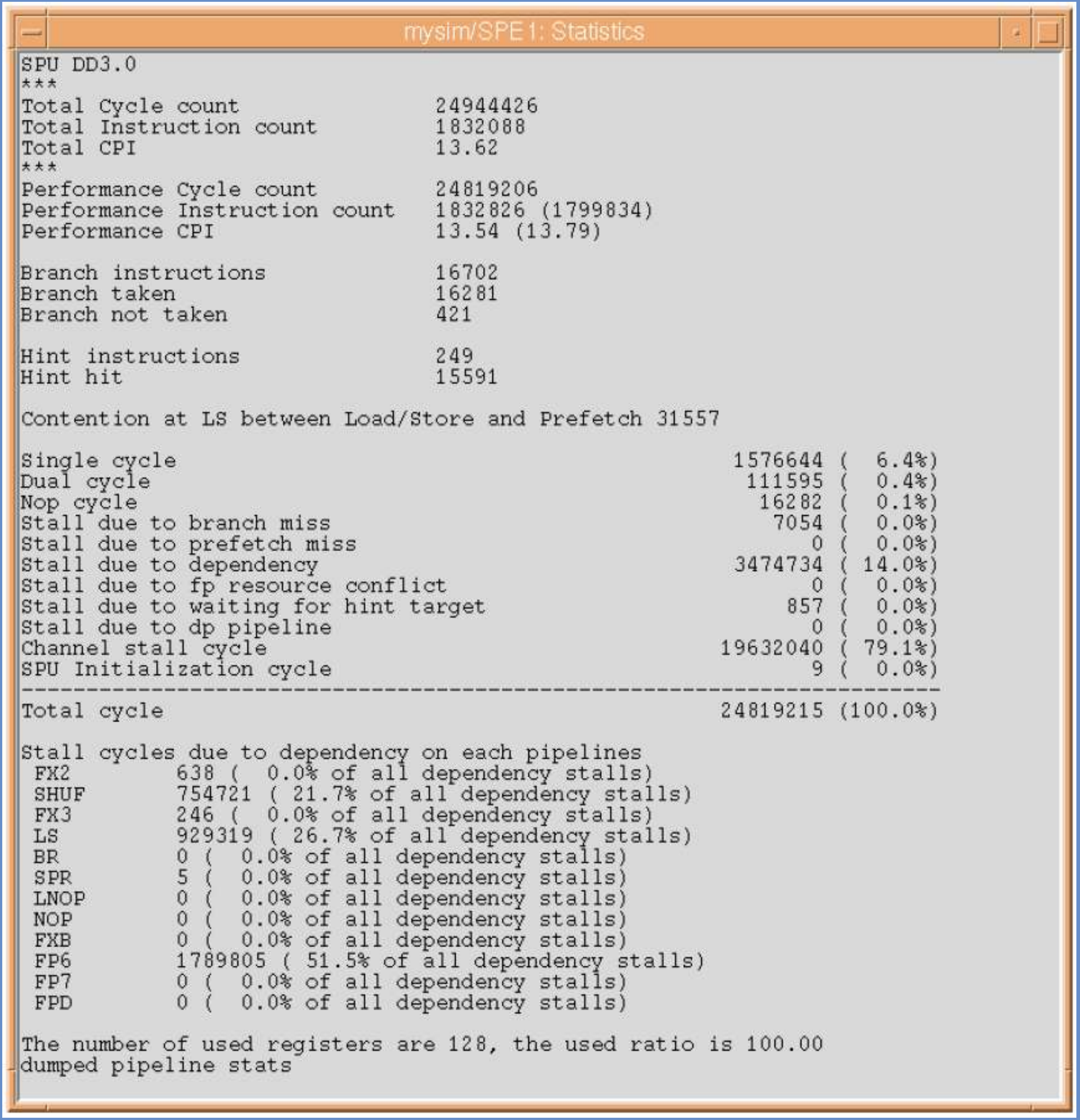
```
RSA                  PPU                        2 SPU
key length  sign         sign/sec     sign         sign/sec
1024-bits   0.003435s    291.2        0.005208s    192.0
2048-bits   0.017541s    57.0         0.009775s    102.3
4096-bits   0.109793s    9.1          0.037392s    26.7
```

- OpenSSL speed on PPU vs. 2 SPU parallel mod_exp()
- using IBM-MPM
- 3.2GHz Cell.

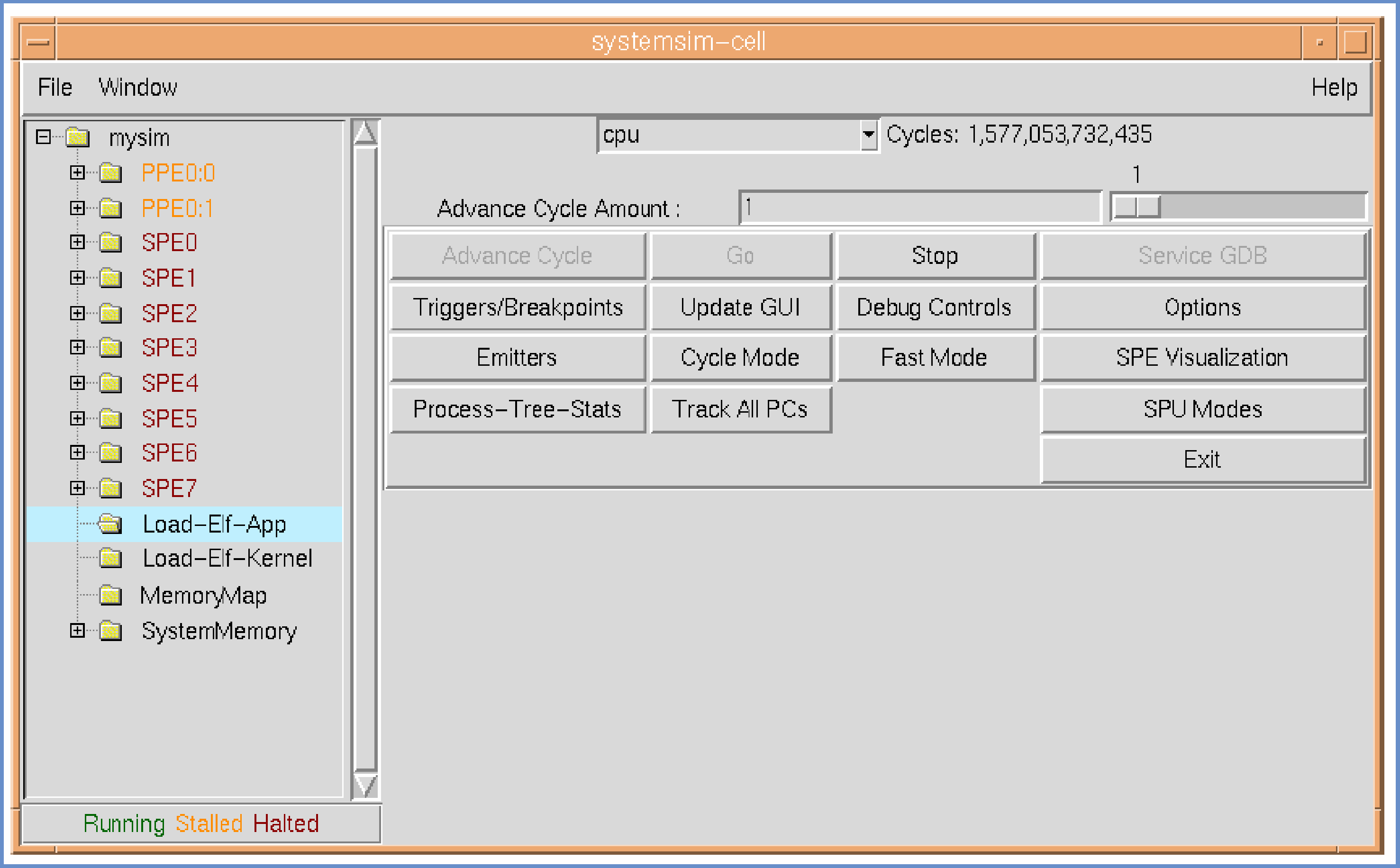
## Next steps…

- OpenSSL native uses Karatsuba method but IBM Library doesn't. (x3 ?)
- Plan is to port MIRACL to get this
- Look at OpenSSL pre-release for improving ALL algorithms
- recent SDK announcement hits at improvements

## SUPPORT SLIDES



```
SPU DD3.0
***
Total Cycle count                24944426
Total Instruction count          1832088
Total CPI                        13.62
***
Performance Cycle count          24819206
Performance Instruction count    1832826 (1799834)
Performance CPI                  13.54 (13.79)

Branch instructions              16702
Branch taken                     16281
Branch not taken                 421

Hint instructions                249
Hint hit                         15591

Contention at LS between Load/Store and Prefetch 31557

Single cycle                                              1576644 (  6.4%)
Dual cycle                                                 111595 (  0.4%)
Nop cycle                                                   16282 (  0.1%)
Stall due to branch miss                                     7054 (  0.0%)
Stall due to prefetch miss                                      0 (  0.0%)
Stall due to dependency                                   3474734 ( 14.0%)
Stall due to fp resource conflict                               0 (  0.0%)
Stall due to waiting for hint target                          857 (  0.0%)
Stall due to dp pipeline                                        0 (  0.0%)
Channel stall cycle                                      19632040 ( 79.1%)
SPU Initialization cycle                                        9 (  0.0%)
-----------------------------------------------------------------------
Total cycle                                              24819215 (100.0%)

Stall cycles due to dependency on each pipelines
 FX2        638 (  0.0% of all dependency stalls)
 SHUF       754721 ( 21.7% of all dependency stalls)
 FX3        246 (  0.0% of all dependency stalls)
 LS         929319 ( 26.7% of all dependency stalls)
 BR         0 (  0.0% of all dependency stalls)
 SPR        5 (  0.0% of all dependency stalls)
 LNOP       0 (  0.0% of all dependency stalls)
 NOP        0 (  0.0% of all dependency stalls)
 FXB        0 (  0.0% of all dependency stalls)
 FP6        1789805 ( 51.5% of all dependency stalls)
 FP7        0 (  0.0% of all dependency stalls)
 FPD        0 (  0.0% of all dependency stalls)

The number of used registers are 128, the used ratio is 100.00
dumped pipeline stats
```



Cycles: 1,577,053,732,435

Advance Cycle Amount : 1

# Time tool…

```
0D      45                        ilhu        $9,32896
1D      456789                    stqd        $1,-112($1)
0D       56                       ai          $1,$1,-112
1D       567890                   lqa         $10,$CONSTANT_AREA+0
0D        67                      ilhu        $11,32896
1D        678901                  lq a         $12,$CONSTANT_AREA+16
0D         78                     iohl        $9,1543
1D         7890                   cdd         $7,8($1)
0D          89                    iohl        $11,1029
1D          8901                  cdd         $16,0($1)

0          78                            a            $3,$3,$5
0d          89                           a            $4,$4,$8
1d          --0123                       shlqbyi      $4,$4,4
0d             ---45                     a            $3,$3,$4
1d                --6789                 rotqbyi      $4,$3,8
1                    ---0123             shufb      $2,$4,$7,$2
1                        ---456789       stqd         $2,0($6)
```

## MIRACL

- Mike Scott / Shamus Software
- Small footprint & very fast. Optimal for crypto
- 2 steps
  - Muldvd()
  - KCM
    - 3 x speed up ?
- exercise ;
  - Vectorise to SPU Intrinsics
  - Examine generated ASM code and optimise.
  - Count cycles, fix pipelining etc.