

# 10 Sonnets Concerning FPT

Philip Leong

phwl@cse.cuhk.edu.hk

*The Chinese University of Hong Kong*




# 1. Introduction

Reconfigurable computing is known,  
Otherwise by the two letters “RC”.  
Enormous potential it will be shown,  
Acceleration to a large degree.

Parallelism is the basic trick  
We apply to solve almost everything,  
In spatial form we arrange our logic  
Indeed to von Neumann we must not cling.

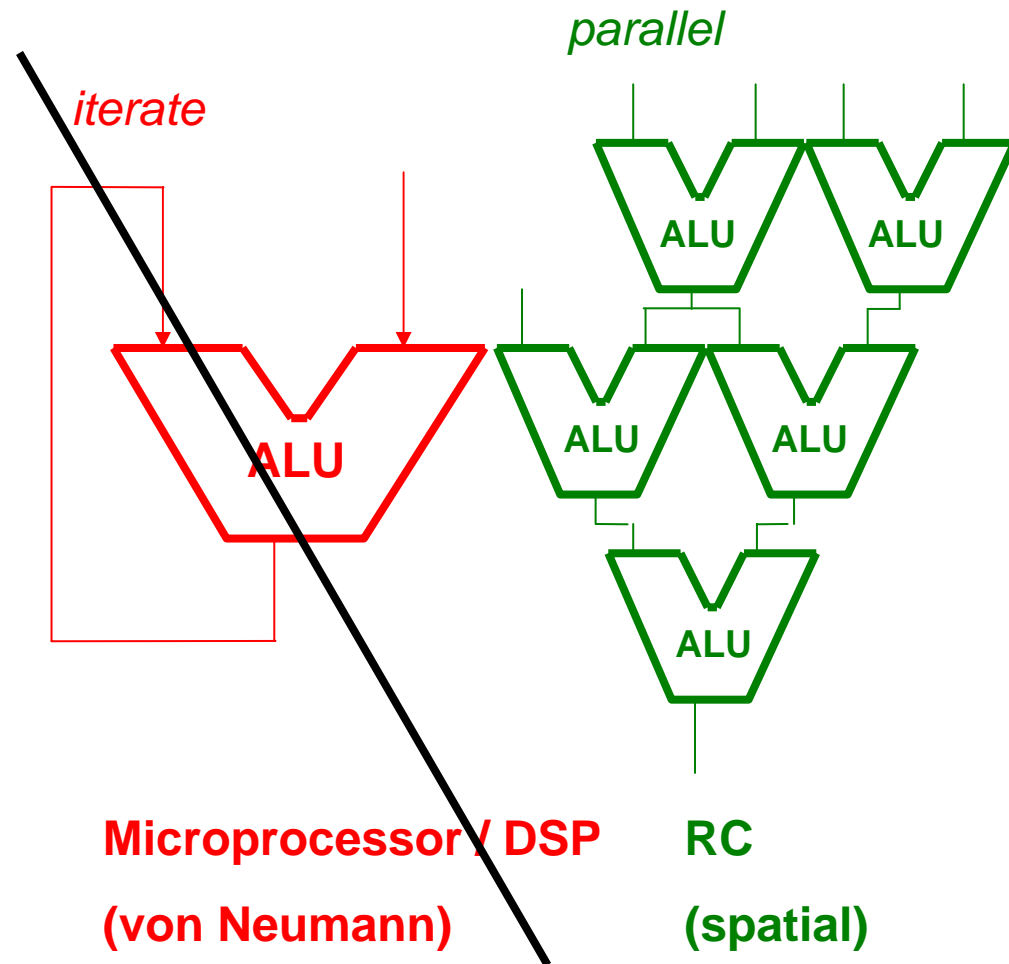
This talk, as presented in sonnet form  
Will attempt to highlight developments;  
We first describe what has become the norm  
Then we'll present some new embodiments.

Particular focus we will anoint  
To problems computing in floating point.

- 
- Introduction
  - Floating point FPGAs
  - Latency
  - Memory
  - Virtual logic
  - Applications
  - Conclusion

# Introduction

- Spatial computing
- FP applications can greatly broaden scope
  - Architecture?
  - Other developments needed?
  - Potential areas for research



## 2. Fixed vs Floating

RC inevitably in the past,  
Was done using fixed point arithmetic;  
Optimised to be both small and quite fast,  
To designers we are sympathetic.

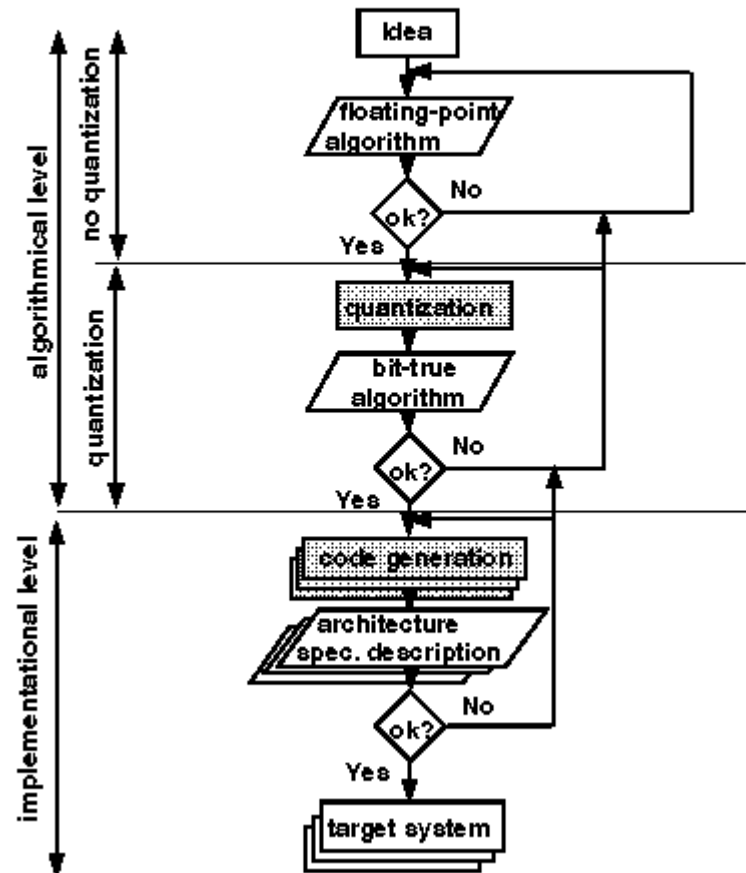
For fixed point design, a difficult art,  
Requires much patience, effort and time;  
Quantization errors right from the start,  
Overflow, underflow, we must decline.

IEEE 754 indeed,  
Will address this most troublesome issue,  
But embedded FPU cores will need,  
Much extra memory bandwidth to accrue.

This should be a focus for our research,  
For good architectures we aim to search.

## Design Flow

- Complex
- Difficult to verify
- Time consuming
- FP doesn't have these issues



Commercial FPGA  
Weaknesses (for FP)

Large area, low clock  
frequency

Run out of resources

Floating Point FPGA

Coarse-grain, hardwired  
FPUs

Dynamic reconfiguration

# 3. Coarse vs Fine Grain

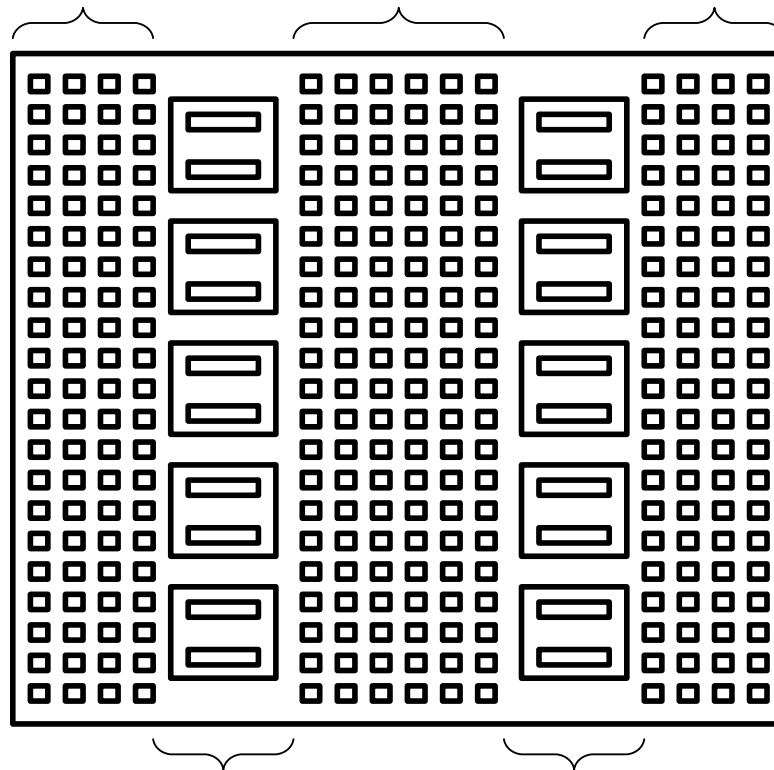
Fine grain, coarse grain, using both is the trick,  
Control and data are not quite the same;  
Hard FPU's, density like ASIC,  
Employing this scheme, performance we gain.

We found that coarse-grained blocks greatly reduce,  
FP logic, compare and multiplex;  
Using this scheme we found we could produce,  
Twenty-fold smaller area at best.

Floating-point units are better in speed,  
Which is another important factor;  
We see that they are effective indeed,  
Multiplier, adder and subtractor.

And with coarse grain chips there is less routing,  
This serves the goal for which we are shooting.

Fine-grained units



Coarse-grained FP units

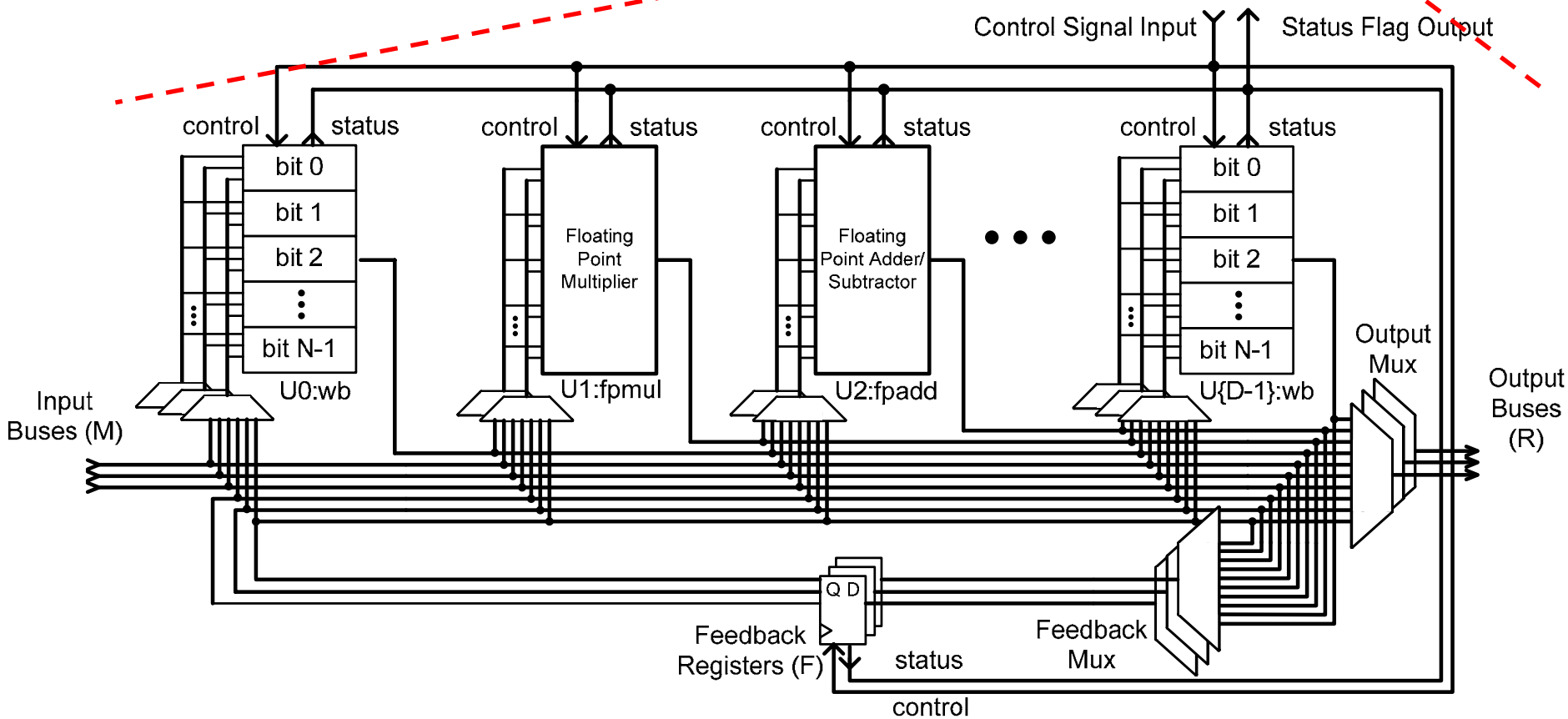
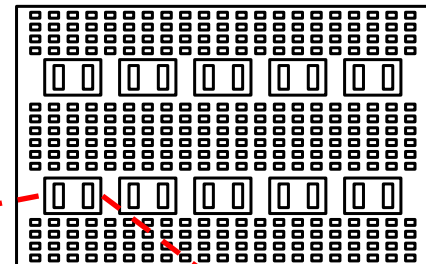
# 4. A Coarse-grained FP Fabric

The coarse-grained block is directional see,  
Data in from the left, out on the right;  
Transformed by blocks of which we have three,  
We are able to customise each site.

The first is a 4-LUT for general use,  
It is good for multiplex and compare,  
A register too so we can produce,  
Memory and latched values of output there.

In floating point blocks there are mult and add,  
Single or double precision in size;  
For speed and size the word block is not bad,  
At compile time we parameterise.

Compared to island-style less routing note,  
Faster, smaller and supporting the float.



D=9, M=4, R=3, F=3, 2 add, 2 mul: best density over benchmarks

	Floating Point hybrid FPGA		XC2V3000-6			
	Area (slices)	Delay (ns)	Area (slices)	Delay (ns)	Area (times)	Delay (times)
bfly	565	9.02	13733	24.57	<b>24.3</b>	<b>2.72</b>
dscg	661	10.11	9614	22.78	<b>14.5</b>	<b>2.25</b>
fir4	371	9.06	11290	23.68	<b>30.4</b>	<b>2.61</b>
mm3	642	8.90	8889	23.4	<b>13.8</b>	<b>2.63</b>
ode	545	9.74	8238	21.93	<b>15.1</b>	<b>2.25</b>
bgm	1810	10.00	30207	24.34	<b>16.7</b>	<b>2.43</b>

Coarse-grained vs XC2V3000  
(floating point benchmarks)

Geometric  
Mean

**18.3**      **2.48**

# 5. Latency

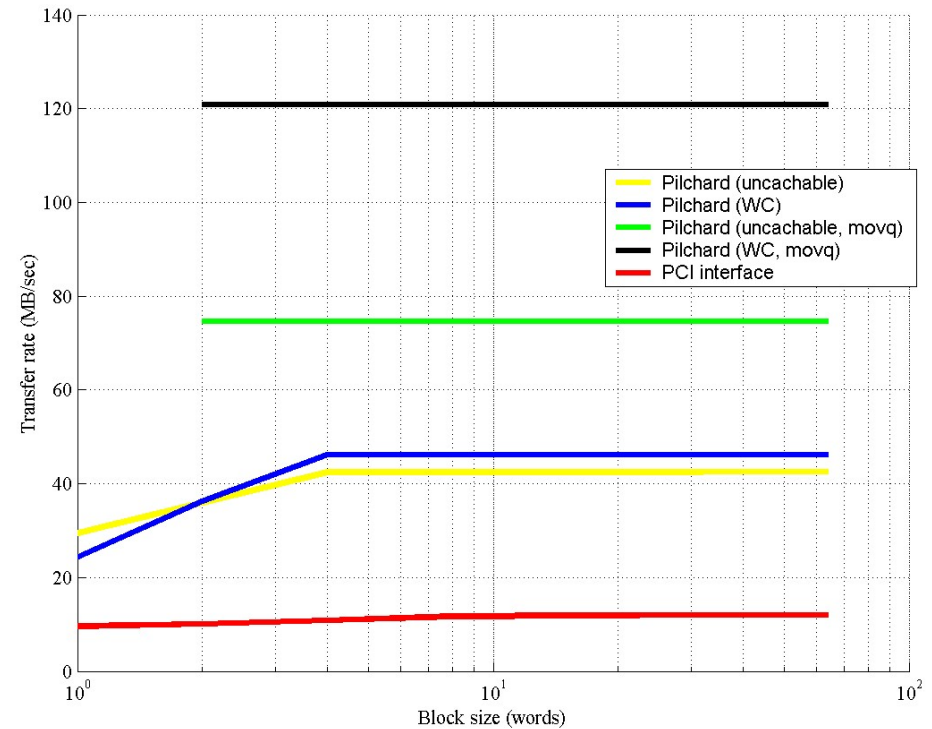
I/O is an unfortunate evil,  
Without it our speedups would be higher;  
We must have high bandwidth for retrieval,  
Very low latency we aspire.

We hence desire a low latency bus,  
Tightly coupled to the host CPU;  
This is not a desire but a must,  
So the data can come fast and be true.

Pilchard was an answer which used a DIMM,  
In the memory bus speedups we did see;  
Today some other connections are in,  
Such as FPGA to FSB.

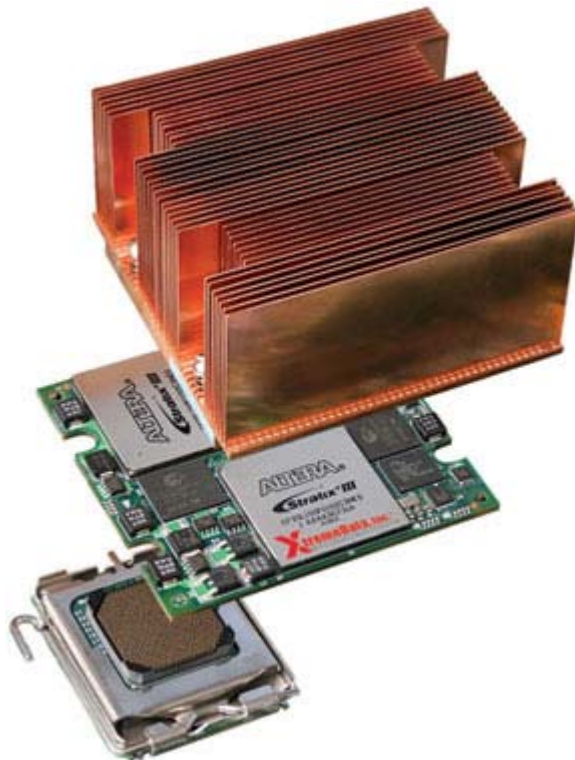
Intel and IBM's Geneseo,  
Hopes to also address RC IO.

- Memory bus has high bandwidth and low latency
- Pilchard circa 2000





Acceleration technologies for high-performance computing



Front side bus RC systems from DRC, Xtremedata and Nallatech

# 6. Geneseo

A major issue with hardware we hate,  
Is portability with host, we know;  
It's great Intel has stepped up to the plate  
With new technology "Geneseo."

This will be a standard which helps enhance,  
RC interfaces to host machine,  
Supporting coprocessors will advance,  
PCI express to achieve our dream.

This will combine good portability,  
With low latency, high bandwidth and more,  
Without support for full coherency,  
Interoperability furthermore.

For remember FSB coherent,  
And moreover, Geneseo isn't.

# 7. Memory

Memory is so expensive you know,  
Five or six transistors to make a RAM;  
Flash is a single transistor and so  
A reduced area design we plan.

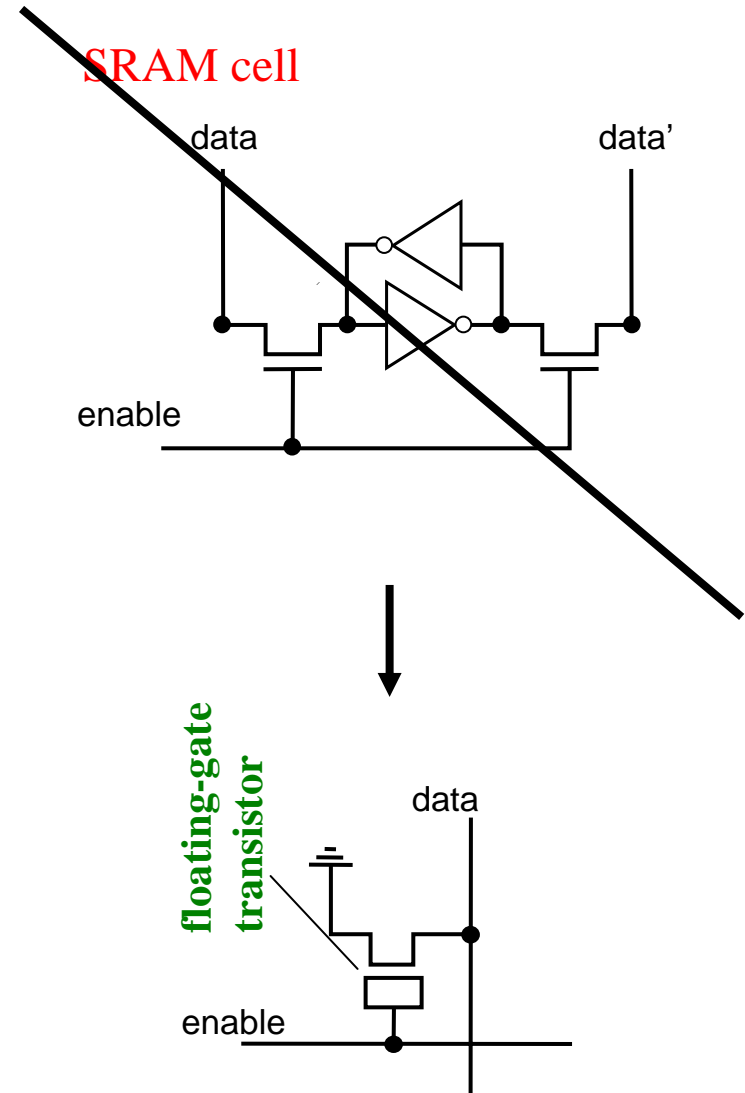
Unfortunately though flash very small,  
Program and erase overhead is high  
Currently large blocks make sense furthermore  
No solution on which we can rely.

The great advantage is high density,  
And has the static power problem beat,  
FPGAs of such propensity;  
To make most current designs obsolete.

Non-volatile too, so you shouldn't see,  
As many designs with CPLDs.

## ■ Use flash for FPGA storage

- Improved density
- Improved power
- Non-volatile



# 8. Virtual Logic

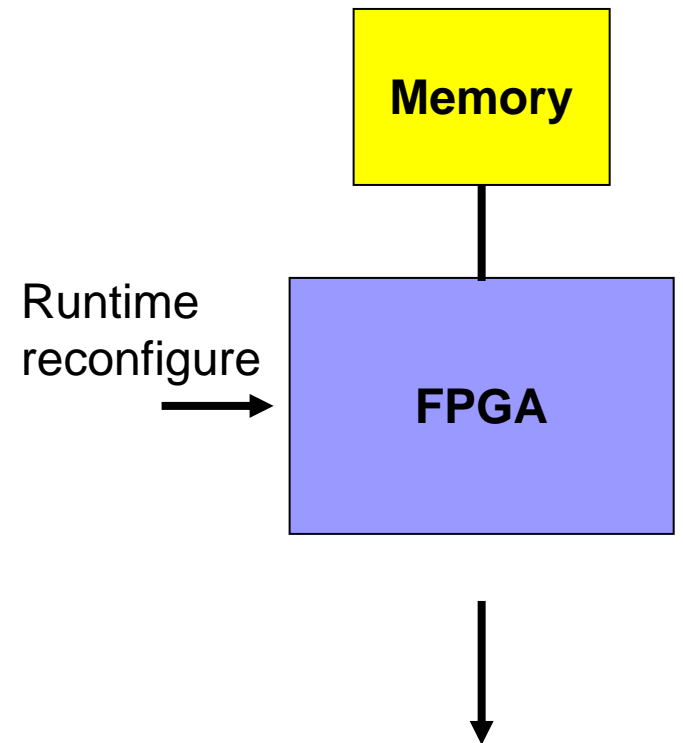
Virtual memory is used by everyone,  
So why not the same scheme for RC;  
To reuse logic would be lots of fun,  
For coarse-grained blocks it makes more sense to me.

Hence designs would never run out of space,  
With CPUs we could better compete;  
Multiple contexts could do it with grace,  
Reduced design time would make this complete.

Large applications we could demand page,  
To designers complexity reduced;  
I'm sure that this scheme would be all the rage,  
Resulting in simpler designs produced.

Till now the vendors are not playing ball,  
A challenge for this community all.

- Obstacle to dynamic reconfiguration is in commercial architectures, we should figure out whether it is a good idea or not
- Coarse-grain has less state to reconfigure



Virtual logic


# 9. Applications

We know that this technology is great,  
But also need the right problems to solve,  
To triumph over the micros we hate,  
So what applications do we involve?

Well science, finance and oil are my bet,  
All are computational in nature,  
With parallel datapaths we will get,  
Low power and performance much greater.

Unfortunately, we have other foes,  
E.g. GPUs, multicores and Cell,  
Easy to program, low power as well,  
They all do run like a bat out of hell.

We'll save the planet, less power, less space,  
And win the CPU/RC arms race.

- 
- Oil and gas exploration
  - Financial engineering
  - Bioinformatics
  - Scientific computing
  - VLSI simulation and verification

# 10. Conclusion

Smaller, faster and less latency we,  
All know what FPT really does need;  
So include flash, coarse-grain and FSB,  
To these research challenges we should heed.

For flash is small with low dissipation,  
Coarse-grain gives us less routing and switches,  
Tools for latency elimination,  
RC applications bring the riches.

Logic devices were there from the start,  
Along with some programmable routing,  
Some future chips will be FP in heart,  
For DSP and other computing.

One final area, out of the norm,  
FPT poetry as an art form.

