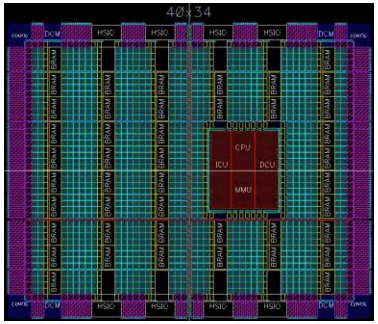
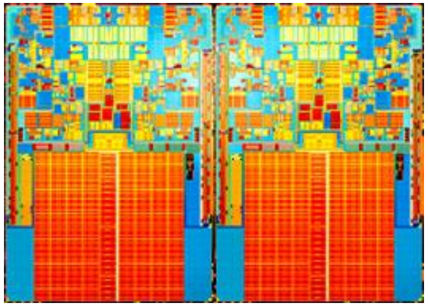
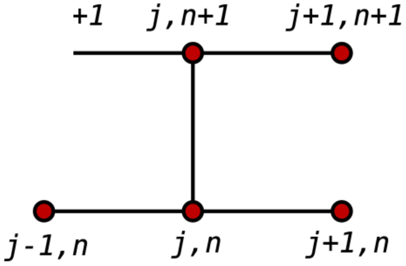
Data-Parallel Programming for FPGAs, GPUs and Mutlicore Vector Instructions

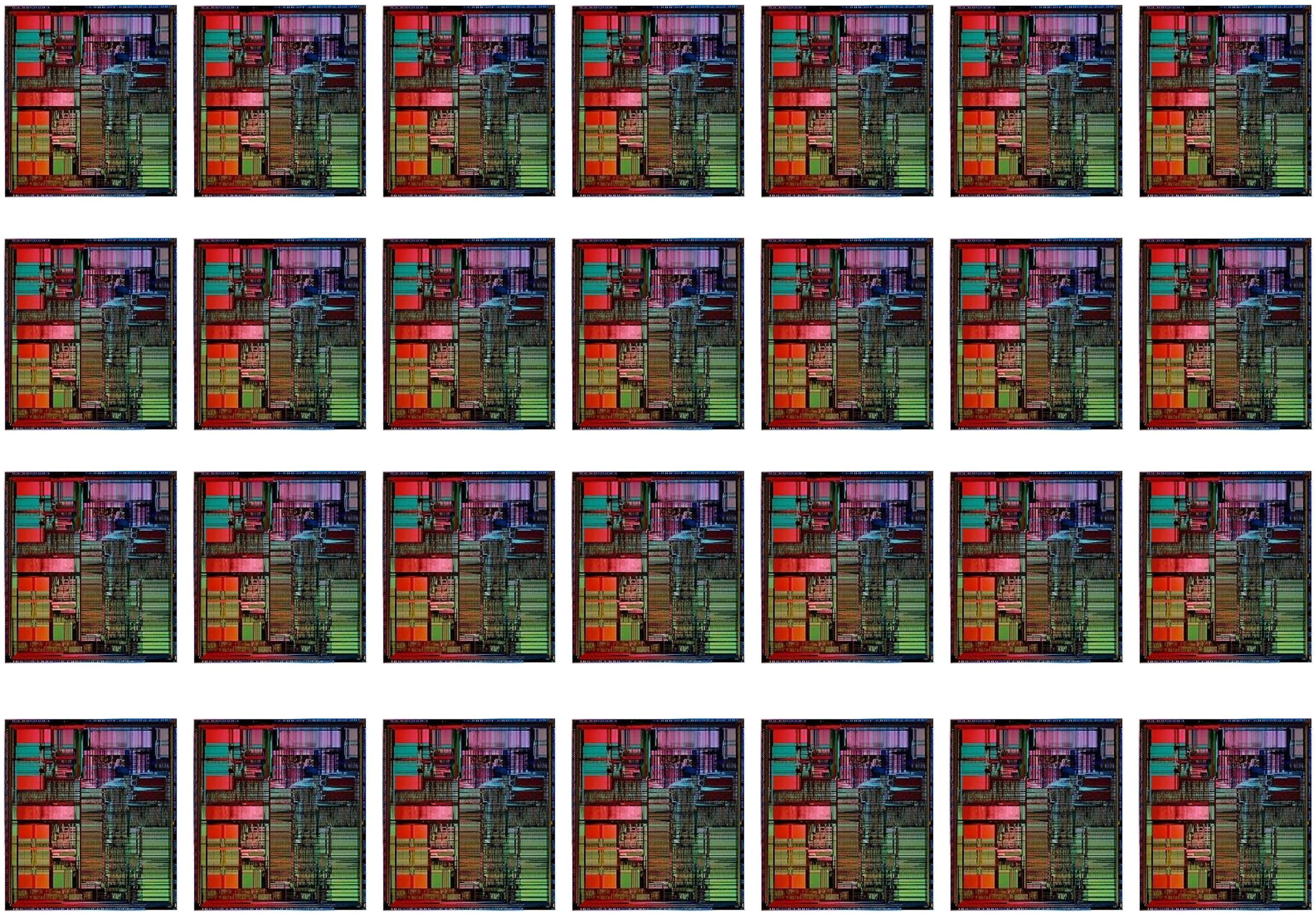
Satnam Singh
The University of Birmingham

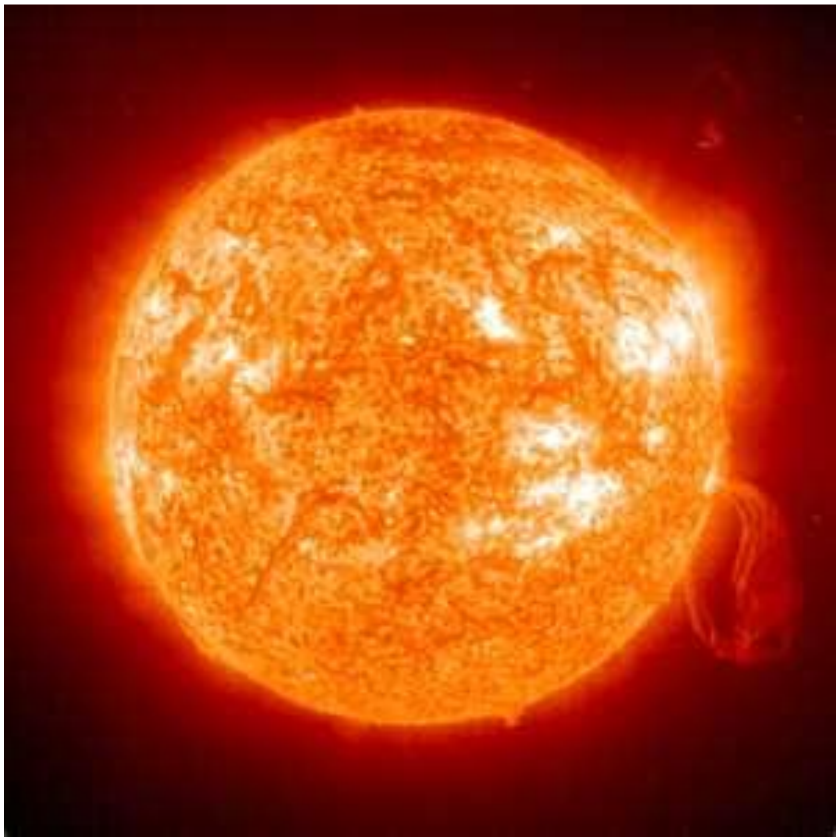


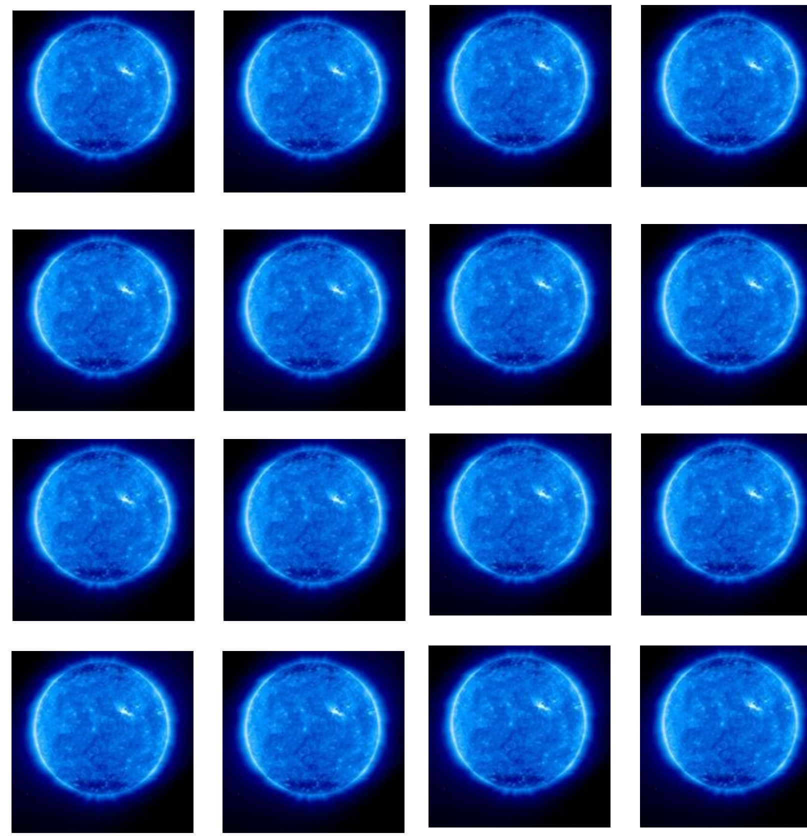




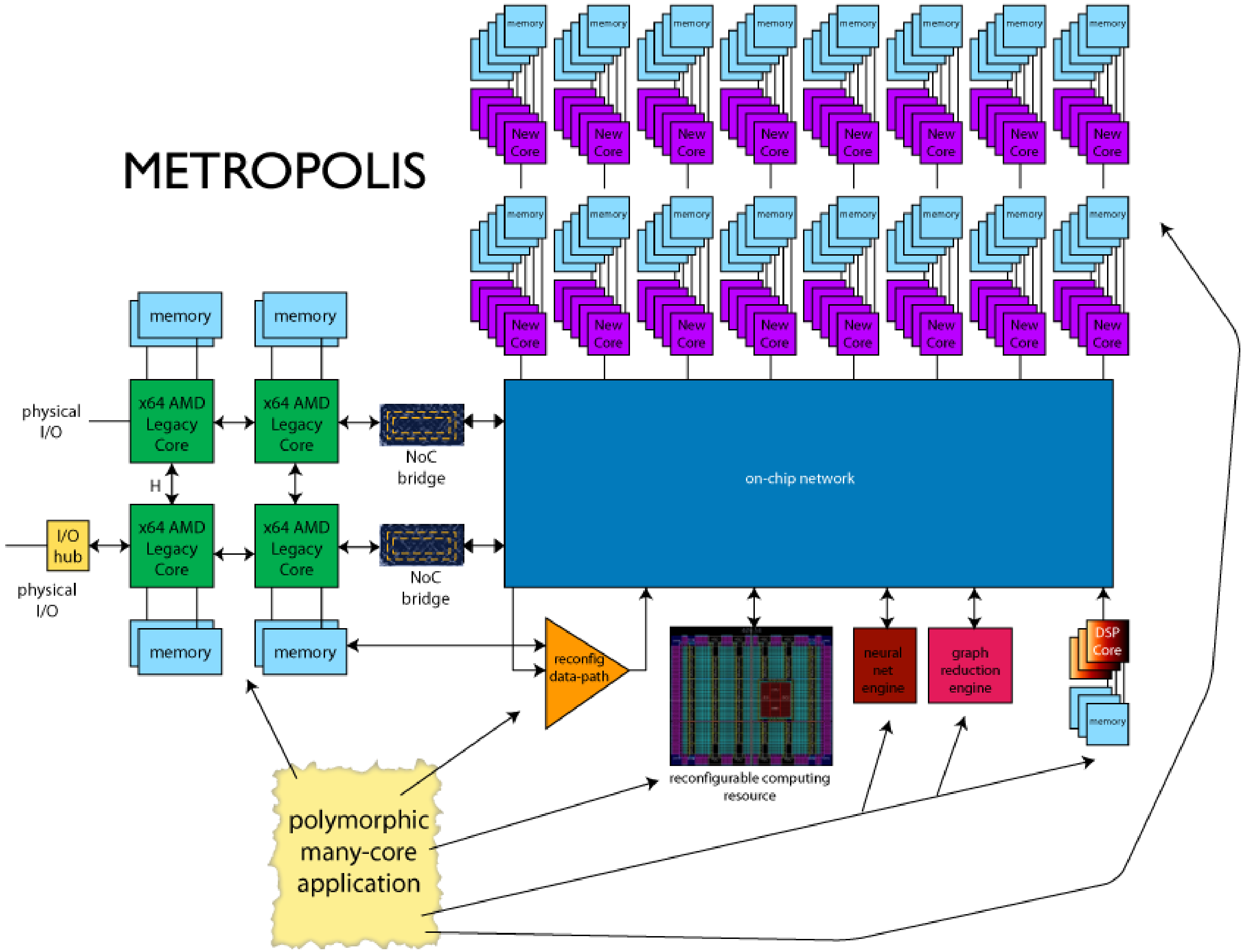
IRQ, NMI







METROPOLIS


memory
New Core
memory
memory
memory
memory
x64 AMD Legacy Core
x64 AMD Legacy Core
x64 AMD Legacy Core
x64 AMD Legacy Core
physical I/O
I/O hub
physical I/O
H
NoC bridge
NoC bridge
on-chip network
reconfig data-path
reconfigurable computing resource
neural net engine
graph reduction engine
DSP Core
polymorphic many-core application

What's New?

Announcing Cluster GPU Instances for Amazon EC2

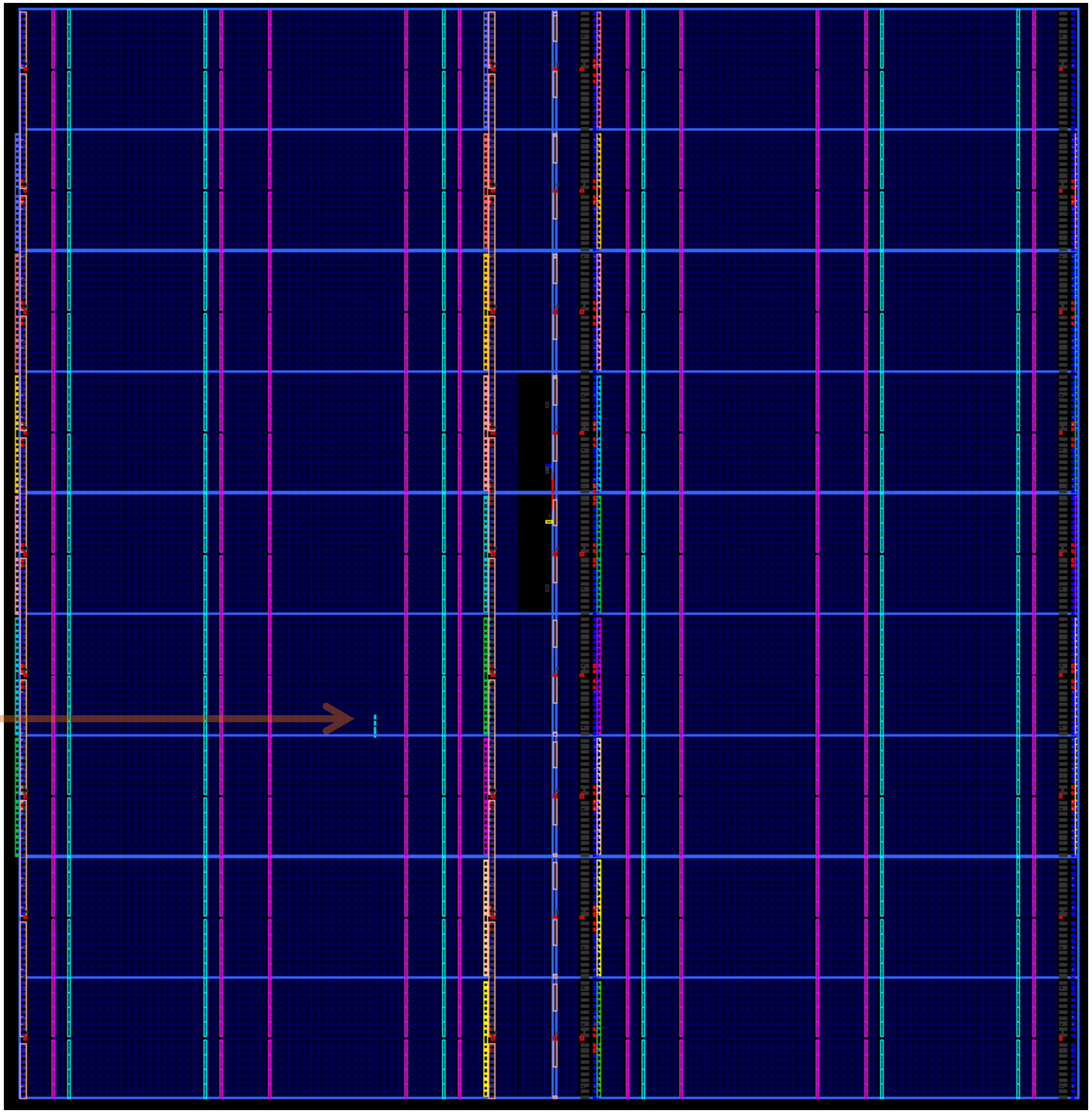
We are excited to announce the immediate availability of Cluster GPU Instances for Amazon EC2, a new instance type designed to deliver the power of GPU processing in the cloud. GPUs are increasingly being used to accelerate the performance of many general purpose computing problems. However, for many organizations, GPU processing has been out of reach due to the unique infrastructural challenges and high cost of the technology. Amazon Cluster GPU Instances remove this barrier by providing developers and businesses immediate access to the highly tuned compute performance of GPUs with no upfront investment or long-term commitment.

Learn more about the new Cluster GPU instances for Amazon EC2 and their use in running HPC applications.


NVIDIA

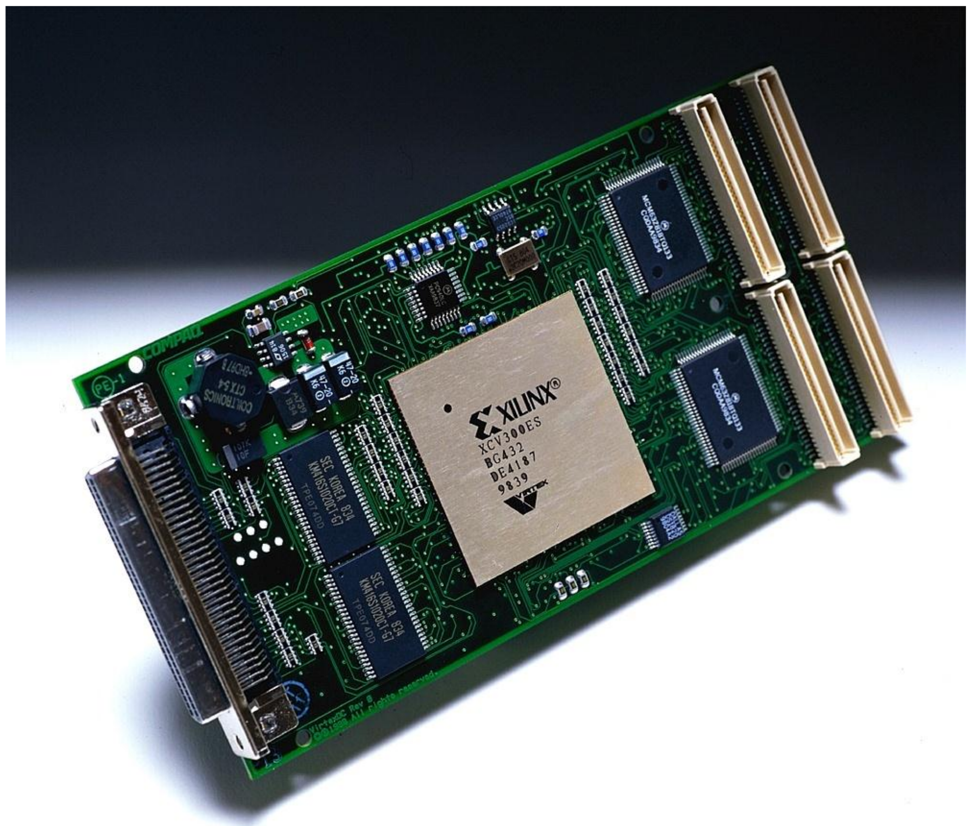
14820 sim-adds
1,037,400,000,000
additions/second

32-bit
integer
Adder
(32/474,240)
>700MHz

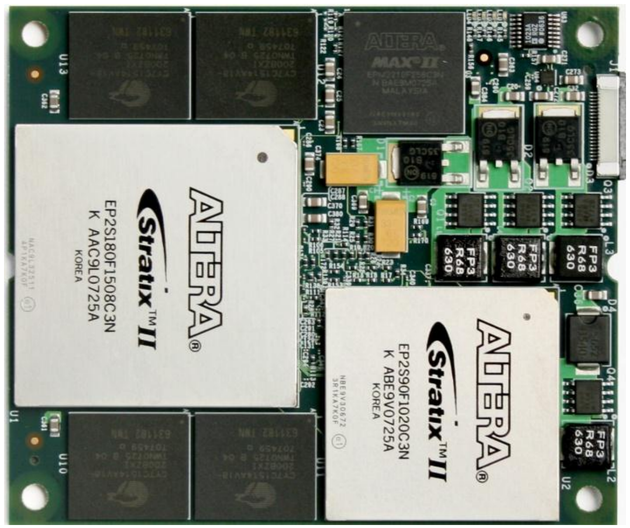


332x1440

XC6VLX760 758,784 logic cells, 864 DSP blocks, 1,440 dual ported 18Kb RAMs

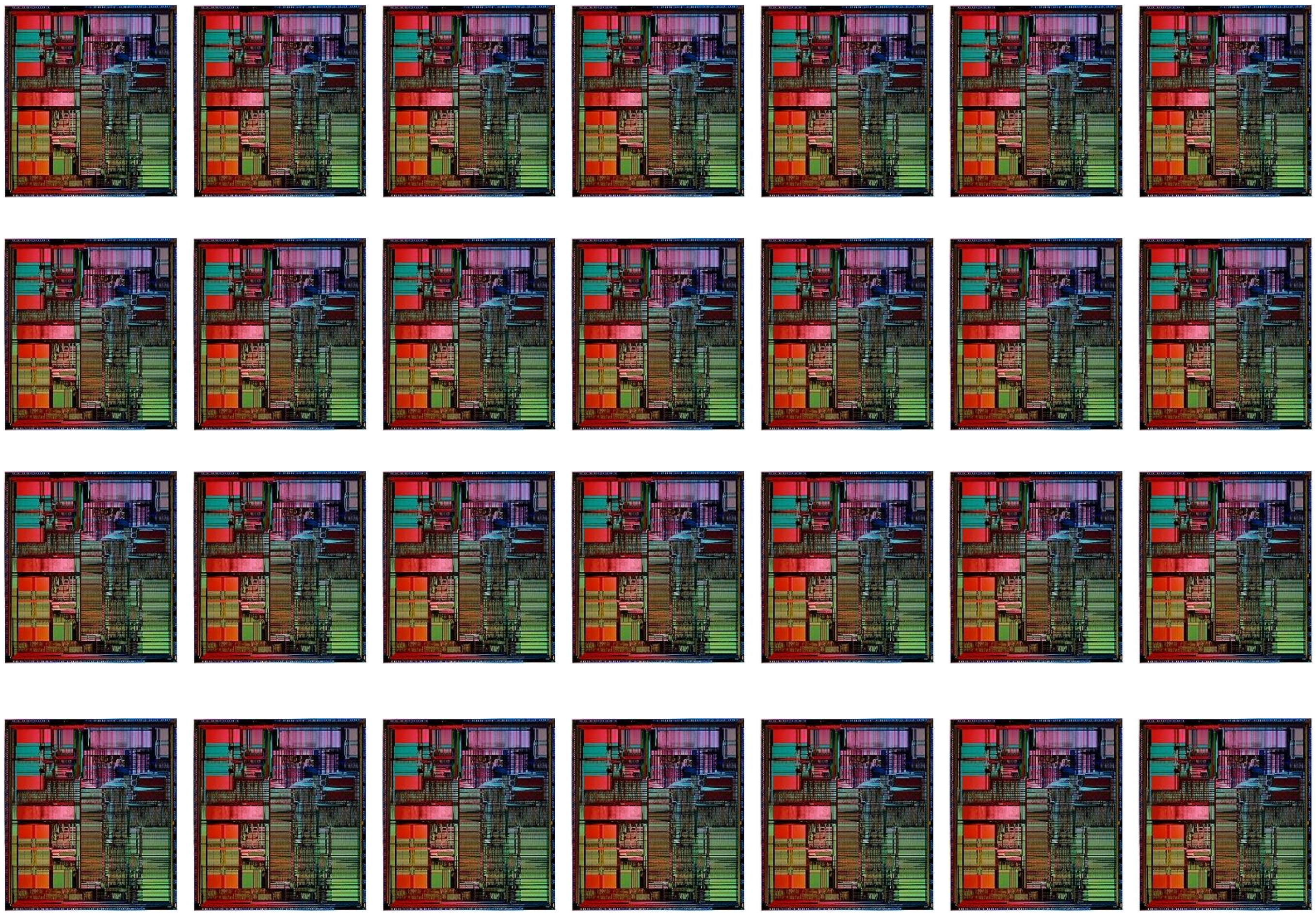

XILINX
XCV300ES
BG432
DE4187
9839

XD2000i FPGA in-socket accelerator for Intel FSB



XD2000F FPGA in-socket accelerator for AMD socket F

XD1000 FPGA co-processor module for socket 940



locks
monitors
condition variables
spin locks
priority inversion

A problem has been detected and Windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

***   gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.

BIOINFO... ATICS **ORIGINAL PAPER** Vol. 23 no. 2 2007, pages 156–161
doi:10.1093/bioinformatics/btl582

Sequence ana...

Striped Sm... -Waterman speeds database searches six times over oth... implementations

Michael Fa...

Received on Ju... on November 13, 2006; accepted on November 14, 2006
Advance Access ... er 16, 2006
Associate Editor...

ABSTRACT

Motivation: Th... uaranteed to find the optimal local alignment ... an. It is also one of the slowest due to the nu... s required for the search. To speed up the ... struction Multiple-Data (SIMD) instructions hav... ze the algorithm at the instruction level.

Results: A faste... e Smith–Waterman algorithm is presented. This ... –8 times performance improvement over oth... h–Waterman implementations. On a 2.0 GHz X... essor, speeds of >3.0 billion cell updates/s were...

Availability: htt... glepages.com/Smith-waterman

Contact: farrar....

search. A disadvantage introduced by processing the values vertically is that conditional branches are placed in the inner loop to compute F. With conditional code the execution time is dependent on the length of the query string and the database, the scoring matrix and gap penalties. A speedup of over six times was reported over an optimized non-SIMD implementation.

This paper presents a new Smith–Waterman implementation where the SIMD registers are parallel to the query sequence, but are accessed in a striped pattern. Like the Rognes implementation, the query profile is calculated once for the database search, but the conditional F calculations are moved outside the inner loop. Calculations speeds of >3.0 GCUPS are achieved. This is a speedup of 2–8 times over the Wozniak and Rognes SIMD implementations.

BMC Bioinformatics

Research

Open Access

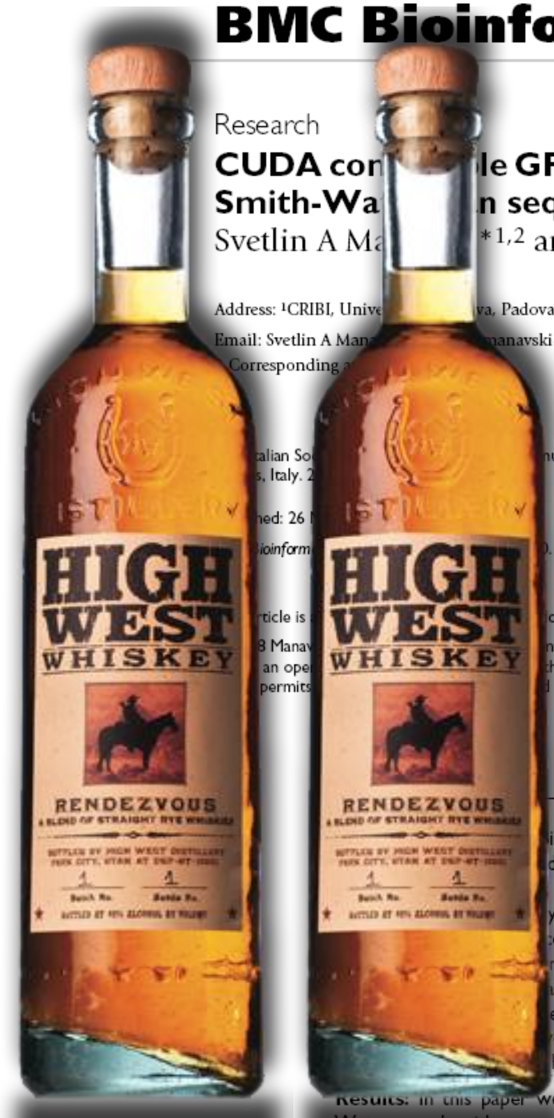
CUDA cor le GPU cards as efficient hardware accelerators for Smith-Wa n sequence alignment

Svetlin A Ma *[1,2] and Giorgio Valle[1]

Address: [1]CRIBI, Unive va, Padova, Italy and [2]Elaide, Srl, Padova, Italy

Email: Svetlin A Man manavski@cribi.unipd.it; Giorgio Valle - giorgio.valle@unipd.it

Corresponding a



alian So ual Meeting 2007
, Italy. 2

hed: 26

ioinform .1186/1471-2105-9-S2-S10

rticle is central.com/1471-2105/9/S2/S10

for similarities in protein and DNA databases has become a routine iology. The Smith-Waterman algorithm has been available for more than dynamic programming approach that explores all the possible alignments as a result it returns the optimal local alignment. Unfortunately, the y high, requiring a number of operations proportional to the product of es. Furthermore, the exponential growth of protein and DNA databases man algorithm unrealistic for searching similarities in large sets of sons heuristic approaches such as those implemented in FASTA and ed, allowing faster execution times at the cost of reduced sensitivity. The ork is to exploit the huge computational power of commonly available high performance solutions for sequence alignment.

Results: In this paper we present what we believe is the fastest solution of the exact Smith-Waterman algorithm running on commodity hardware. It is implemented in the recently released CUDA programming environment by NVidia. CUDA allows direct access to the hardware primitives of the last-generation Graphics Processing Units (GPU) G80. Speeds of more than 3.5 GCUPS (Giga Cell Updates Per Second) are achieved on a workstation running two GeForce 8800 GTX. Exhaustive tests have been done to compare our implementation to SSEARCH and BLAST, running on a 3 GHz Intel Pentium IV processor. Our solution was also compared to a recently published GPU implementation and to a Single Instruction Multiple Data (SIMD) solution. These tests show that our implementation performs from 2 to 30 times faster than any other previous attempt available on commodity hardware.

BMC Bioinformatics

BioMed Central

Methodolo e

Open Access

160-fold eration of t ith-Waterm gorithm using a field pro mable gate (FPGA)

Isaac TS L rren Shum[2] a in Truong*[1,2]

Address: [1]Institut ials and Biomedical En iversity of Toronto, 164 , Toronto, Ontario, M5S 3G9, Canada and [2]Edward S ment of Electrical neering, University g's College Circle, Toronto, Ontario, M5S 3G4, C

Email: Isa .ca; Warre toronto.ca; ong@utoronto.ca

* Corresp

Published: Received: 1
Accepted: 7

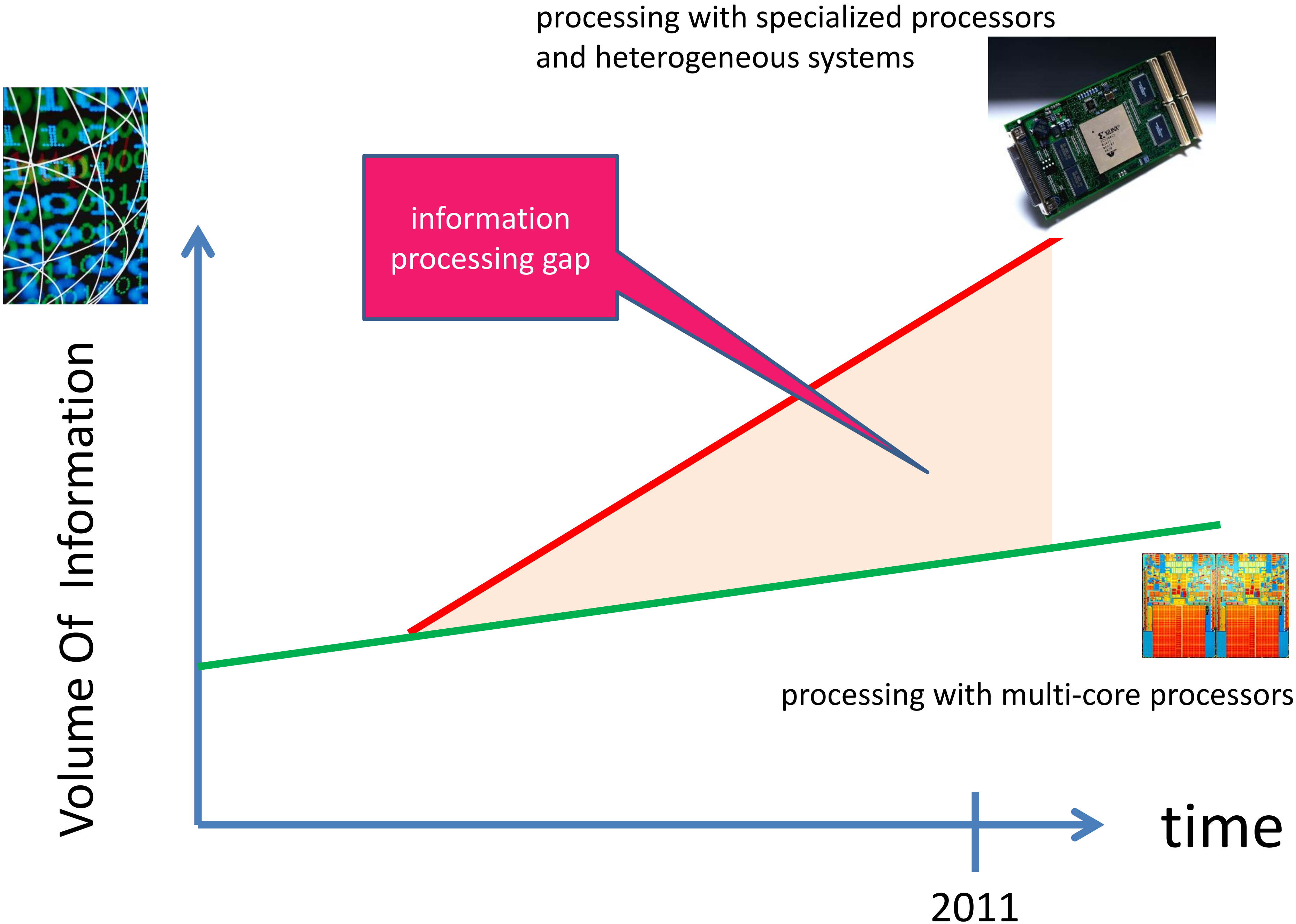
MC Bioinf 1186/147

This articl .biomedc

infer h tly gene terman (SW) o find ent betw hen searching s that millions thm becomes pensive

per, we he Smit using FPGA- at impl omputin ell of the SW a grid SW ma speed of field h the F cations e algorithm's by up to a pure so unning on the same FPGA with an Altera Nios II softprocessor.

Conclusion: This design of FPGA accelerated hardware offers a new promising direction to seeking computation improvement of genomic database searching.


processing with specialized processors
and heterogeneous systems
information
processing gap
Volume Of Information
processing with multi-core processors
time
2011


11
12

OpenMP

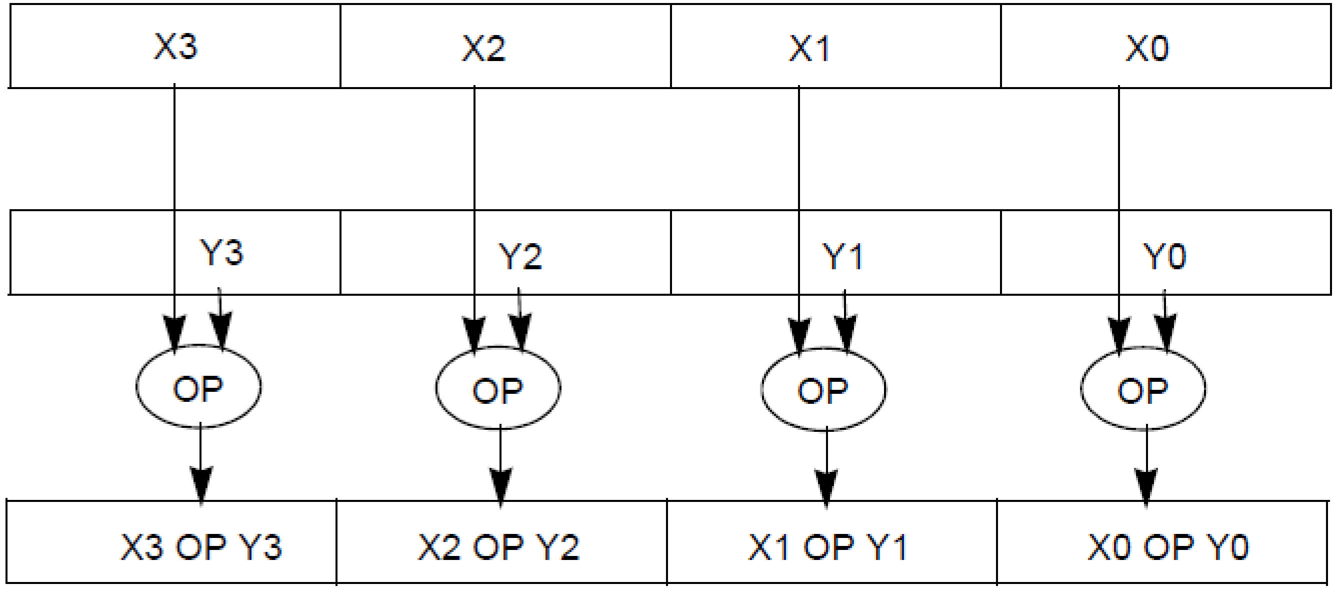
```
#include <stdio.h>
int main(int argc, char* argv[])
{
     const unsigned int n = 5000000 ;
     float *a = new float[n];
     float *b = new float[n];
     float *c = new float[n];
     int i, j ;
     #pragma omp parallel for
     for (i=0; i<n; i++)
        c[i] = a[i] + b[i] ;
     return 0;
}
```

SSE2: ADDPS

__m128 **_mm_add_ps** (__m128 a , __m128 b);

r0 := x0 + y0
r1 := x1 + y1
r2 := x2 + y2
r3 := x3 + y3

128-bits
MMX/

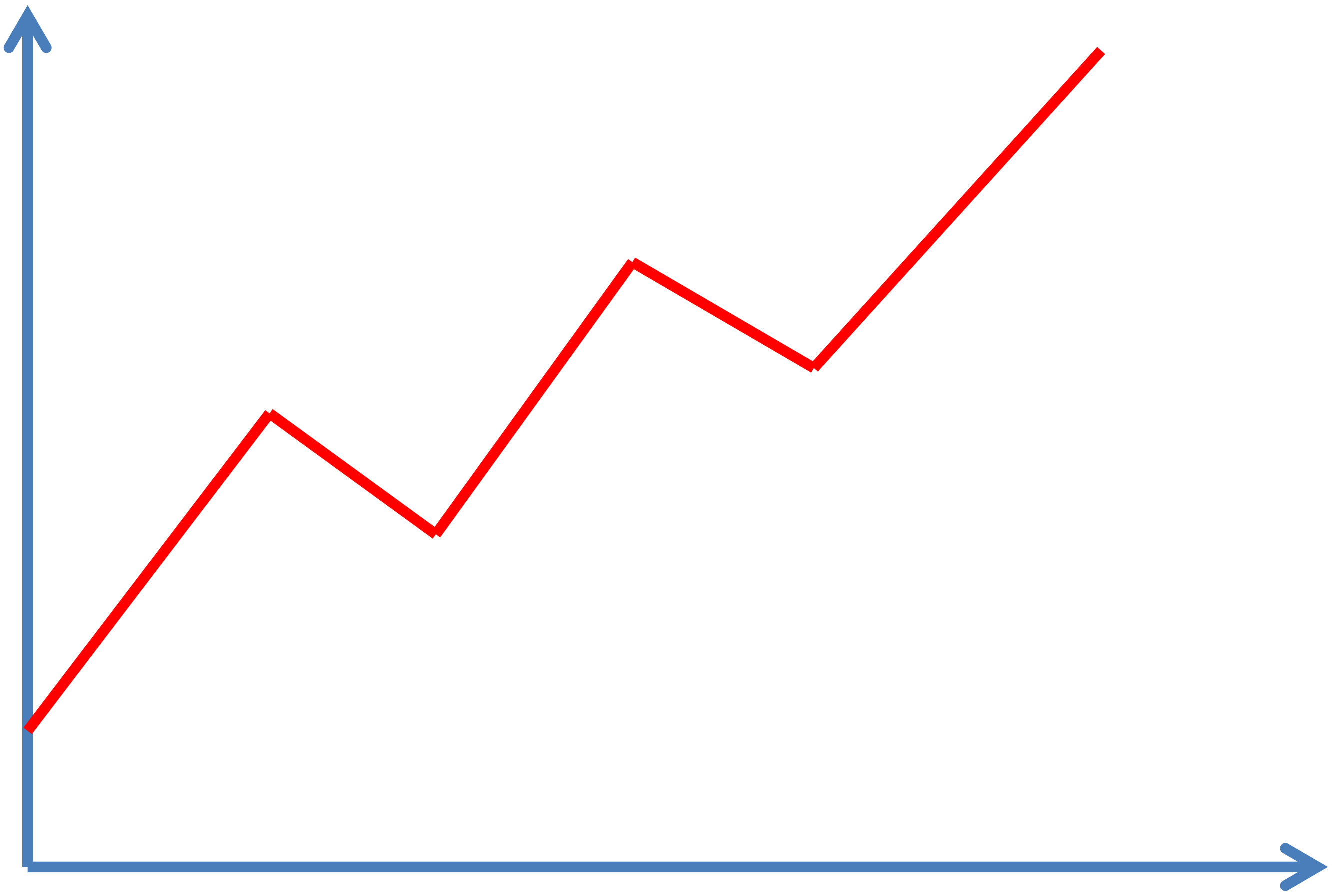


The Accidental Semi-colon

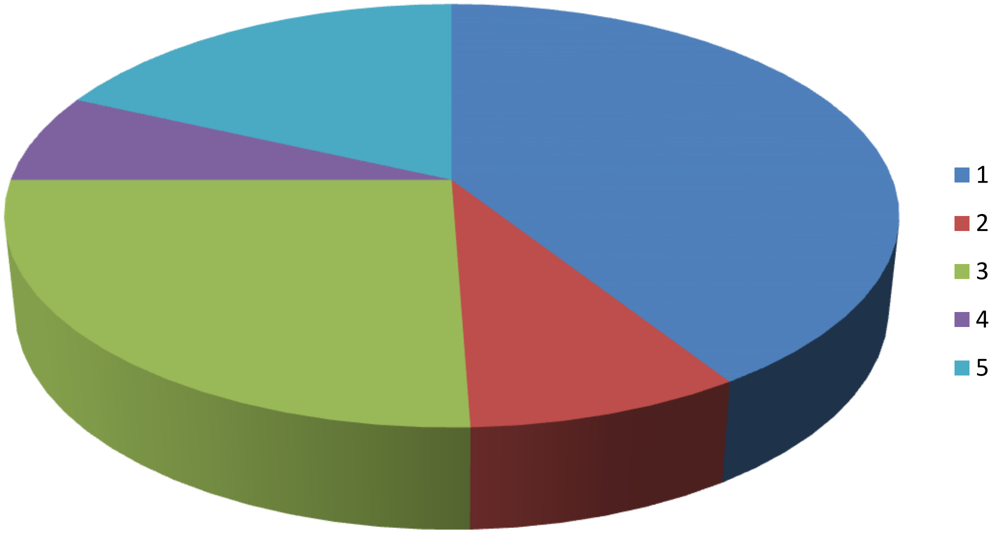
```
public static int[] SequentialFIRFunction(int[] weights, int[] input)
        {
            int[] window = new int[size];
            int[] result = new int[input.Length];
            // Clear to window of x values to all zero.
            for (int w = 0; w < size; w++)
                window[w] = 0;
            // For each sample...
            for (int i = 0; i < input.Length; i++)
            {
                // Shift in the new x value
                for (int j = size - 1; j > 0; j--)
                    window[j] = window[j - 1];
                window[0] = input[i];
                // Compute the result value
                int sum = 0;
                for (int z = 0; z < size; z++)
                    sum += weights[z] * window[z];
                result[i] = sum;
            }
            return result;
        }
```

$$y_t = \sum_{k=0}^{N-1} a_k x_{t-k}$$

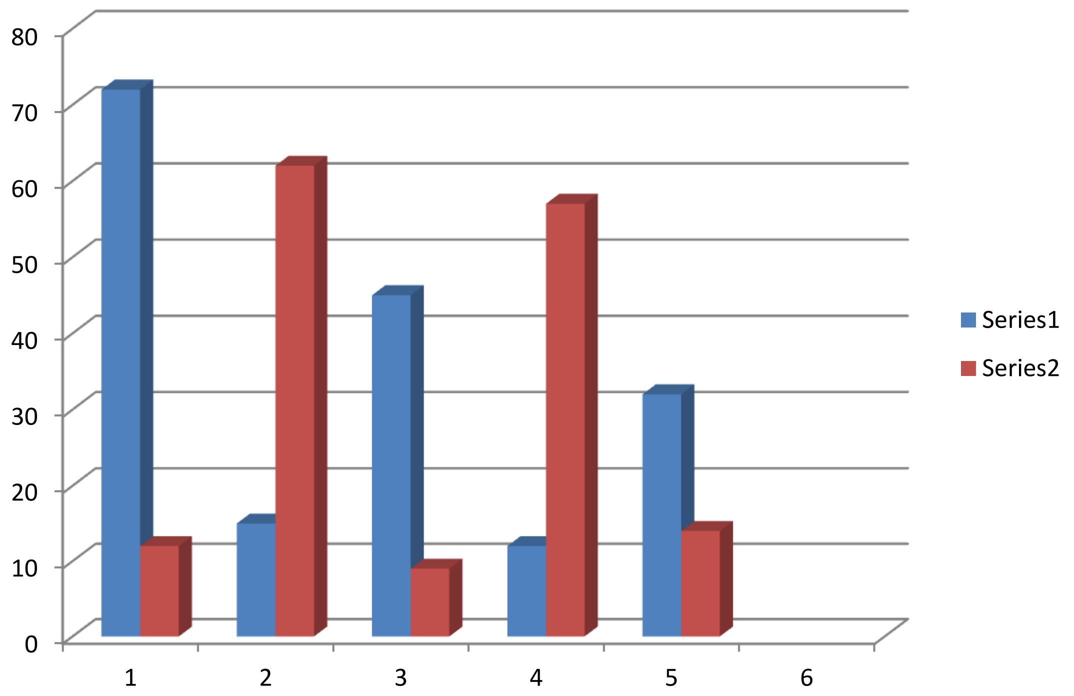
PLDI 1998



PLDI 2003



PLDI 2010



POPL 1998

$$\frac{p \xrightarrow[I \cup \{S\}]{O, k} p' \quad S \in O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O \setminus \{S\}, k} \delta_1^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})}$$

$$\frac{p \xrightarrow[I \setminus \{S\}]{O, k} p' \quad S \notin O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O, k} \delta_1^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})}$$

POPL 2002

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S \in O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O\setminus\{S\},k} \delta_1^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S \notin O}{\texttt{signal } S \texttt{ in } p \texttt{ end} \xrightarrow[I]{O,k} \delta_1^k(\texttt{signal } S \texttt{ in } p' \texttt{ end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S \in O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S \in O^+}{\texttt{signal } S \texttt{ in } p \texttt{ end} \circ\!\xrightarrow[I]{O^+\setminus\{S\},k^+} \delta_1^{k^+}(\texttt{signal } S \texttt{ in } p^+ \texttt{ end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S \notin O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S \notin O^+}{\texttt{signal } S \texttt{ in } p \texttt{ end} \circ\!\xrightarrow[I]{O^-,k^-} \delta_1^{k^-}(\texttt{signal } S \texttt{ in } p^- \texttt{ end})}$$

$$\frac{\texttt{emit S} \circ\!\xrightarrow[\{\texttt{A}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S} \in \{\texttt{S}\} \qquad \texttt{emit S} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S} \in \{\texttt{S}\}}{\texttt{signal S in emit S end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,0} \texttt{nothing}}$$

$$\frac{\texttt{pause} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S} \notin \emptyset \qquad \texttt{pause} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S} \notin \emptyset}{\texttt{signal S in pause end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{signal S in nothing end}}$$

Proof. Structural induction on p. Let us consider the case $p =$ "$\texttt{signal } S \texttt{ in } q \texttt{ end}$". By hypothesis, $p \circ\!\xrightarrow[I]{O_0,k_0} p_0$. As (signal++) or (signal−−) must be used to define this reaction, there exist $O_0^-, k_0^-, q_0^-, O_0^+, k_0^+, q_0^+$ such that:

$$q \circ\!\xrightarrow[I\setminus\{S\}]{O_0^-,k_0^-} q_0^- \quad \text{and} \quad q \circ\!\xrightarrow[I\cup\{S\}]{O_0^+,k_0^+} q_0^+$$

Then, using Lemma 3.1,

- either $S \notin O_0^-, S \notin O_0^+, O_0 = O_0^-, k_0 = k_0^-, p_0 = \delta_1^{k_0^-}(\texttt{signal } S \texttt{ in } q_0^- \texttt{ end})$,
- or $S \in O_0^-, S \in O_0^+, O_0 = O_0^+\setminus\{S\}, k_0 = k_0^+, p_0 = \delta_1^{k_0^+}(\texttt{signal } S \texttt{ in } q_0^+ \texttt{ end})$.

POPL 2010

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S\in O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O\setminus\{S\},k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S\notin O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\in O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\in O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^+\setminus\{S\},k^+} \delta_1^{k^+}(\texttt{signal}\ S\ \texttt{in}\ p^+\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\notin O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\notin O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^-,k^-} \delta_1^{k^-}(\texttt{signal}\ S\ \texttt{in}\ p^-\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S\in O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O\setminus\{S\},k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S\notin O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\in O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\in O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^+\setminus\{S\},k^+} \delta_1^{k^+}(\texttt{signal}\ S\ \texttt{in}\ p^+\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\notin O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\notin O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^-,k^-} \delta_1^{k^-}(\texttt{signal}\ S\ \texttt{in}\ p^-\ \texttt{end})}$$

$$\frac{\texttt{emit S} \circ\!\xrightarrow[\{\texttt{A}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\} \qquad \texttt{emit S} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\}}{\texttt{signal S in emit S end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,0} \texttt{nothing}}$$

$$\frac{\texttt{pause} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset \qquad \texttt{pause} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset}{\texttt{signal S in pause end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{signal S in nothing end}}$$

Proof. Structural induction on p. Let us consider the case $p =$ "$\texttt{signal}\ S\ \texttt{in}\ q\ \texttt{end}$". By hypothesis, $p \circ\!\xrightarrow[I]{O_0,k_0} p_0$. As (signal++) or (signal−−) must be used to define this reaction, there exist $O_0^-, k_0^-, q_0^-, O_0^+, k_0^+, q_0^+$ such that:

$$q \circ\!\xrightarrow[I\setminus\{S\}]{O_0^-,k_0^-} q_0^- \quad \text{and} \quad q \circ\!\xrightarrow[I\cup\{S\}]{O_0^+,k_0^+} q_0^+$$

Then, using Lemma 3.1,

- either $S\notin O_0^-, S\notin O_0^+, O_0 = O_0^-, k_0 = k_0^-, p_0 = \delta_1^{k_0^-}(\texttt{signal}\ S\ \texttt{in}\ q_0^-\ \texttt{end})$,
- or $S\in O_0^-, S\in O_0^+, O_0 = O_0^+\setminus\{S\}, k_0 = k_0^+, p_0 = \delta_1^{k_0^+}(\texttt{signal}\ S\ \texttt{in}\ q_0^+\ \texttt{end})$.

$$\frac{\texttt{emit S} \circ\!\xrightarrow[\{\texttt{A}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\} \qquad \texttt{emit S} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\}}{\texttt{signal S in emit S end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,0} \texttt{nothing}}$$

$$\frac{\texttt{pause} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset \qquad \texttt{pause} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset}{\texttt{signal S in pause end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{signal S in nothing end}}$$

Proof. Structural induction on p. Let us consider the case $p =$ "$\texttt{signal}\ S\ \texttt{in}\ q\ \texttt{end}$". By hypothesis, $p \circ\!\xrightarrow[I]{O_0,k_0} p_0$. As (signal++) or (signal−−) must be used to define this reaction, there exist $O_0^-, k_0^-, q_0^-, O_0^+, k_0^+, q_0^+$ such that:

$$q \circ\!\xrightarrow[I\setminus\{S\}]{O_0^-,k_0^-} q_0^- \quad \text{and} \quad q \circ\!\xrightarrow[I\cup\{S\}]{O_0^+,k_0^+} q_0^+$$

Then, using Lemma 3.1,

- either $S\notin O_0^-, S\notin O_0^+, O_0 = O_0^-, k_0 = k_0^-, p_0 = \delta_1^{k_0^-}(\texttt{signal}\ S\ \texttt{in}\ q_0^-\ \texttt{end})$,
- or $S\in O_0^-, S\in O_0^+, O_0 = O_0^+\setminus\{S\}, k_0 = k_0^+, p_0 = \delta_1^{k_0^+}(\texttt{signal}\ S\ \texttt{in}\ q_0^+\ \texttt{end})$.

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S\in O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O\setminus\{S\},k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S\notin O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\in O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\in O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^+\setminus\{S\},k^+} \delta_1^{k^+}(\texttt{signal}\ S\ \texttt{in}\ p^+\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\notin O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\notin O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^-,k^-} \delta_1^{k^-}(\texttt{signal}\ S\ \texttt{in}\ p^-\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\cup\{S\}]{O,k} p' \qquad S\in O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O\setminus\{S\},k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \xrightarrow[I\setminus\{S\}]{O,k} p' \qquad S\notin O}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \xrightarrow[I]{O,k} \delta_1^k(\texttt{signal}\ S\ \texttt{in}\ p'\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\in O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\in O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^+\setminus\{S\},k^+} \delta_1^{k^+}(\texttt{signal}\ S\ \texttt{in}\ p^+\ \texttt{end})}$$

$$\frac{p \circ\!\xrightarrow[I\setminus\{S\}]{O^-,k^-} p^- \qquad S\notin O^- \qquad p \circ\!\xrightarrow[I\cup\{S\}]{O^+,k^+} p^+ \qquad S\notin O^+}{\texttt{signal}\ S\ \texttt{in}\ p\ \texttt{end} \circ\!\xrightarrow[I]{O^-,k^-} \delta_1^{k^-}(\texttt{signal}\ S\ \texttt{in}\ p^-\ \texttt{end})}$$

$$\frac{\texttt{emit S} \circ\!\xrightarrow[\{\texttt{A}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\} \qquad \texttt{emit S} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\}}{\texttt{signal S in emit S end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,0} \texttt{nothing}}$$

$$\frac{\texttt{pause} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset \qquad \texttt{pause} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset}{\texttt{signal S in pause end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{signal S in nothing end}}$$

Proof. Structural induction on p. Let us consider the case $p =$ "$\texttt{signal}\ S\ \texttt{in}\ q\ \texttt{end}$". By hypothesis, $p \circ\!\xrightarrow[I]{O_0,k_0} p_0$. As (signal++) or (signal−−) must be used to define this reaction, there exist $O_0^-, k_0^-, q_0^-, O_0^+, k_0^+, q_0^+$ such that:

$$q \circ\!\xrightarrow[I\setminus\{S\}]{O_0^-,k_0^-} q_0^- \quad \text{and} \quad q \circ\!\xrightarrow[I\cup\{S\}]{O_0^+,k_0^+} q_0^+$$

Then, using Lemma 3.1,

- either $S\notin O_0^-, S\notin O_0^+, O_0 = O_0^-, k_0 = k_0^-, p_0 = \delta_1^{k_0^-}(\texttt{signal}\ S\ \texttt{in}\ q_0^-\ \texttt{end})$,
- or $S\in O_0^-, S\in O_0^+, O_0 = O_0^+\setminus\{S\}, k_0 = k_0^+, p_0 = \delta_1^{k_0^+}(\texttt{signal}\ S\ \texttt{in}\ q_0^+\ \texttt{end})$.

$$\frac{\texttt{emit S} \circ\!\xrightarrow[\{\texttt{A}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\} \qquad \texttt{emit S} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\{\texttt{S}\},0} \texttt{nothing} \qquad \texttt{S}\in\{\texttt{S}\}}{\texttt{signal S in emit S end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,0} \texttt{nothing}}$$

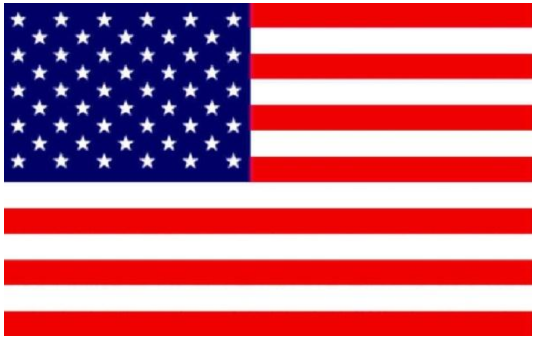
$$\frac{\texttt{pause} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset \qquad \texttt{pause} \circ\!\xrightarrow[\{\texttt{A,S}\}]{\emptyset,1} \texttt{nothing} \qquad \texttt{S}\notin\emptyset}{\texttt{signal S in pause end} \circ\!\xrightarrow[\{\texttt{A}\}]{\emptyset,1} \texttt{signal S in nothing end}}$$

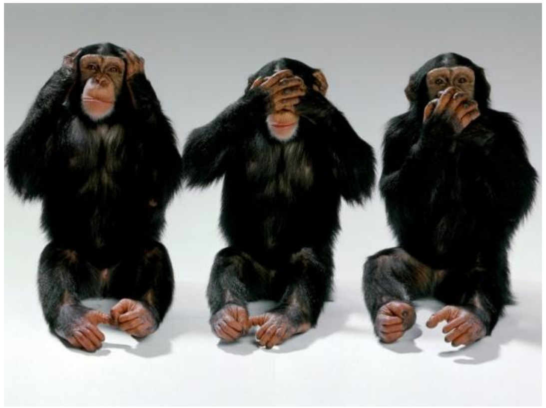
Proof. Structural induction on p. Let us consider the case $p =$ "$\texttt{signal}\ S\ \texttt{in}\ q\ \texttt{end}$". By hypothesis, $p \circ\!\xrightarrow[I]{O_0,k_0} p_0$. As (signal++) or (signal−−) must be used to define this reaction, there exist $O_0^-, k_0^-, q_0^-, O_0^+, k_0^+, q_0^+$ such that:

$$q \circ\!\xrightarrow[I\setminus\{S\}]{O_0^-,k_0^-} q_0^- \quad \text{and} \quad q \circ\!\xrightarrow[I\cup\{S\}]{O_0^+,k_0^+} q_0^+$$

Then, using Lemma 3.1,

- either $S\notin O_0^-, S\notin O_0^+, O_0 = O_0^-, k_0 = k_0^-, p_0 = \delta_1^{k_0^-}(\texttt{signal}\ S\ \texttt{in}\ q_0^-\ \texttt{end})$,
- or $S\in O_0^-, S\in O_0^+, O_0 = O_0^+\setminus\{S\}, k_0 = k_0^+, p_0 = \delta_1^{k_0^+}(\texttt{signal}\ S\ \texttt{in}\ q_0^+\ \texttt{end})$.





$$\frac{}{\Pi; \Sigma; \Theta \vdash n : \texttt{int}}\,(\text{T-INT})$$

$$\frac{}{\Pi; \Sigma; \Theta \vdash !\ell : \Sigma(\ell), \{rd_\ell\}}\,(\text{T-READ})$$

$$\frac{\Pi; \Sigma; \Theta \vdash e : A, \varepsilon_1 \qquad A <: B \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Pi; \Sigma; \Theta \vdash e : B, \varepsilon_2}\,(\text{T-SUB})$$

grand unification theory

polygots

Accelerator

Accelerator is a high-level data parallel library which uses parallel processors such as the GPU or multicore CPU to accelerate execution. Accelerator v1 was released to the MSR Web site in October 2006 and has been periodically updated since then. Accelerator v2 is an MSR incubation project whose goal is to to validate the architecture and API approach with high quality engineering sufficient to gather real-world usage data.

What's in Accelerator v2?

Accelerator v2 builds on Accelerator v1's programming model and adds features that were commonly requested by Accelerator v1 users. New functionality includes:

- Accelerator v2 is written as a native-code C++ library with a managed API wrapper
- Execution on multicore CPUs, both 32 and 64 bit, in addition to DX9 GPUs and CUDA.
- Extensible HW target interface enabling support for execution on new devices
- Ability to execute on multiple devices within a single Accelerator instance
- Asynchronous evaluation of parallel arrays
- Reusable expression graphs: Across different devices and on the same device with different leaf-node data

Download the Accelerator v2 Preview today to try it out. The package includes the Accelerator SDK, extensive documentation and several sample applications to help you get started.

Microsoft Redmond Accelerator Team

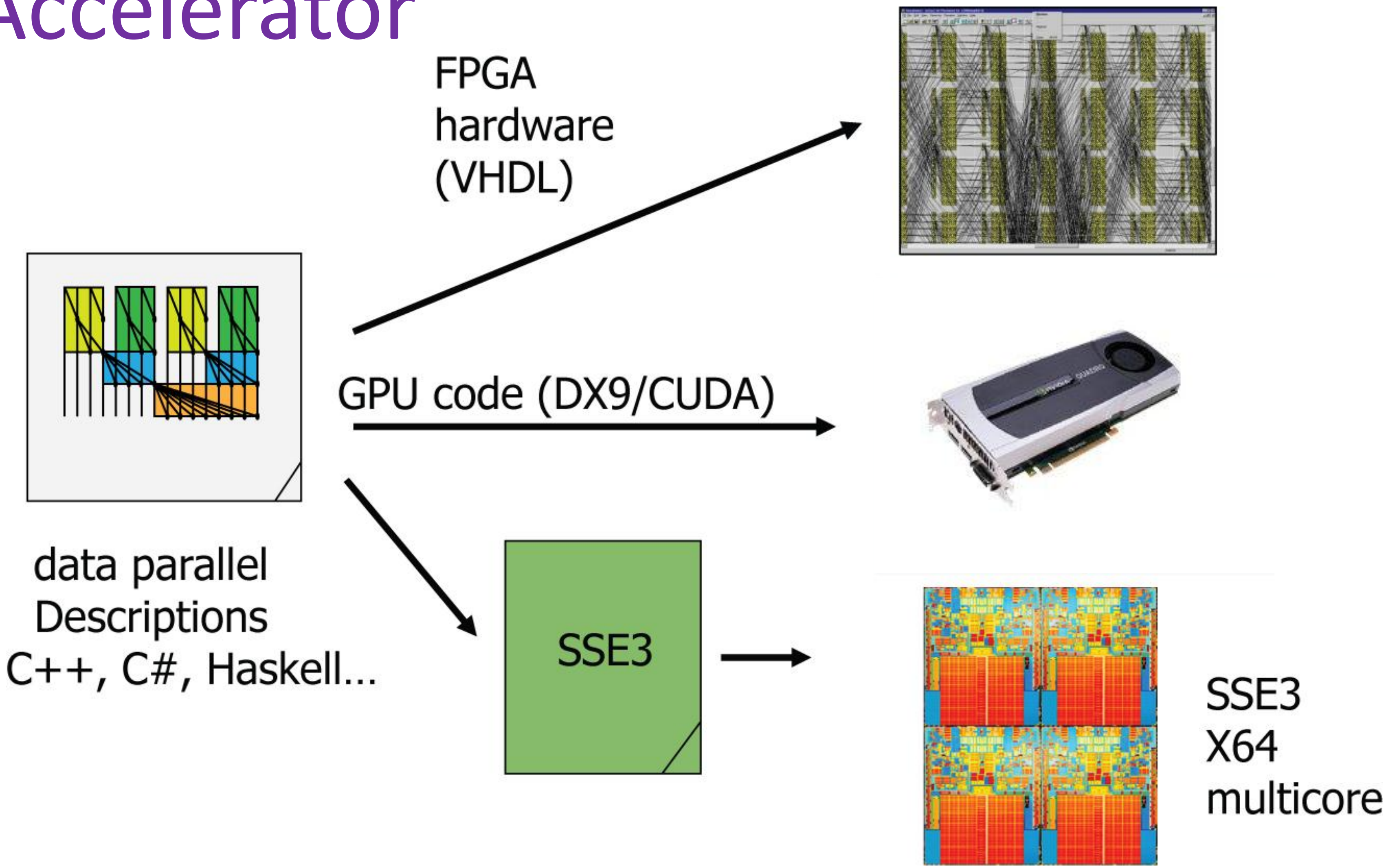
Barry Bond
Kerry Hammil
Lubomir Litchev
<anonymous other person>

Effort vs. Reward (Productivity)

	Thurst Accelerator	CUDAC OpenCL HLSL DirectCompute

low effort	medium effort	high effort
low reward	medium reward	high reward

Accelerator




Kingston
KVR
KVR1066D3S7/4G
1.5V
ASSY IN KOREA
Warranty Void If Removed
GENUINE KINGSTON

DRAM

Addition	$R_{i,j} = A_{i,j} + B_{i,j}$
Multiplication	$R_{i,j} = A_{i,j} \times B_{i,j}$
Scalar Multiplication (k)	$R_{i,j} = kA_{i,j}$
Maximum	$R_{i,j} = \max(A_{i,j}, B_{i,j})$
Sine	$R_{i,j} = \sin A_{i,j}$
Square root	$R_{i,j} = \sqrt{A_{i,j}}$
And	$R_{i,j} = A_{i,j} \wedge B_{i,j}$
Equality test	$R_{i,j} = \begin{cases} \text{true} & \text{if } A_{i,j} = B_{i,j} \\ \text{false} & \text{otherwise} \end{cases}$
Greater than test	$R_{i,j} = A_{i,j} > B_{i,j}$
Select	$R_{i,j} = \begin{cases} B_{i,j} & \text{if } A_{i,j} = \text{true} \\ C_{i,j} & \text{otherwise} \end{cases}$

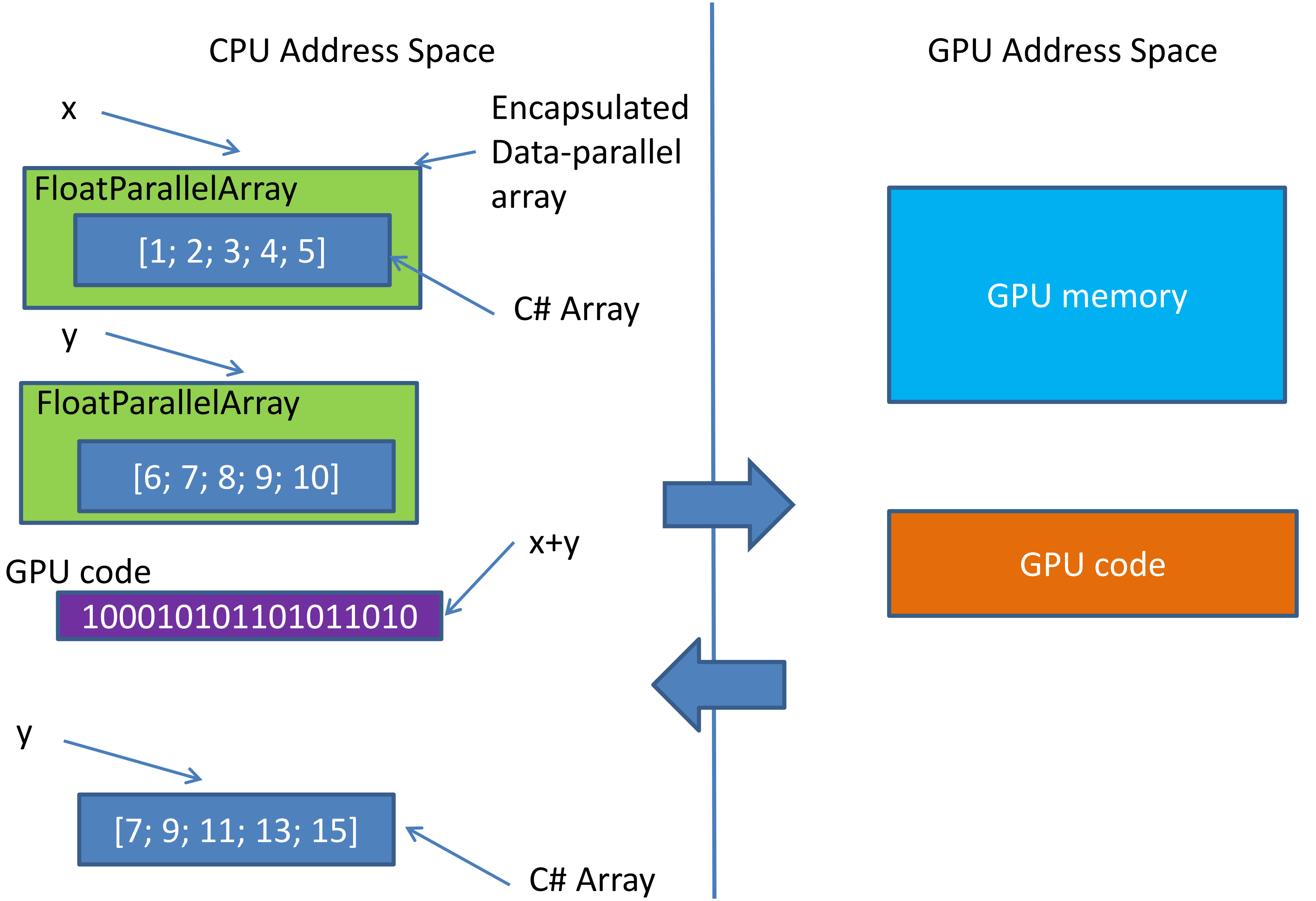
Sum(0)	$R_i = \sum_i A_{i,j}$
Sum(1)	$R_i = \sum_j A_{i,j}$
Maximum value (1)	$R_i = \max_j A_{i,j}$

Section $(b_i, c_i, s_i, b_j, c_j, s_j)$	$R_{i,j} = A_{b_i} + s_i \times i, b_j + s_j \times j$
Shift (m, n)	$R_{i,j} = A_{i-m,j-n}$
Rotate (m, n)	$R_{i,j} = A_{(i-m)\text{mod}M,(j-n)\text{mod}N}$
Replicate (m, n)	$R_{i,j} = A_{i \text{ mod } m,\ j \text{ mod } n}$
Expand (b_i, a_i, b_j, a_j)	$R_{i,j} = A_{i-b_i\text{mod}M,(j-b_j)\text{mod}N}$
Pad (m, a_i, m, a_j, c)	$R_{i,j} = \begin{cases} A_{i-m,j-n} & \text{if in bounds} \\ c & \text{otherwise} \end{cases}$
Transpose(1,0)	$R_{i,j} = A_{j,i}$

Drop Dimension (0)	$R_j = A_{0,j}$
Drop Dimension (1)	$R_i = A_{i,0}$
Add Dimension (1)	$R_{i,j,k} = A_{i,k}$

```csharp
using System;
using Microsoft.ParallelArrays;

namespace AddArraysPointwise
{
    class AddArraysPointwiseDX9
    {
        static void Main(string[] args)
        {
            var x = new FloatParallelArray (new[] {1.0F, 2, 3, 4, 5});
            var y = new FloatParallelArray (new[] {6.0F, 7, 8, 9, 10});
            var dx9Target = new DX9Target();
            var z = x + y;
            foreach (var i in dx9Target.ToArray1D (z))
                Console.Write(i + " ");
            Console.WriteLine();
        }
    }
}
```

```
ps_3_0
dcl_2d  s0
dcl_texcoord0 v0.xy
dcl_2d  s1
texld   r0, v0, s0
texld   r1, v0, s1
add     r1,   r0,   r1
mov     oC0,   r1
```



CPU Address Space
GPU Address Space
x
Encapsulated
Data-parallel
array
FloatParallelArray
[1; 2; 3; 4; 5]
C# Array
GPU memory
y
FloatParallelArray
[6; 7; 8; 9; 10]
x+y
GPU code
GPU code
100010101101011010
y
[7; 9; 11; 13; 15]
C# Array

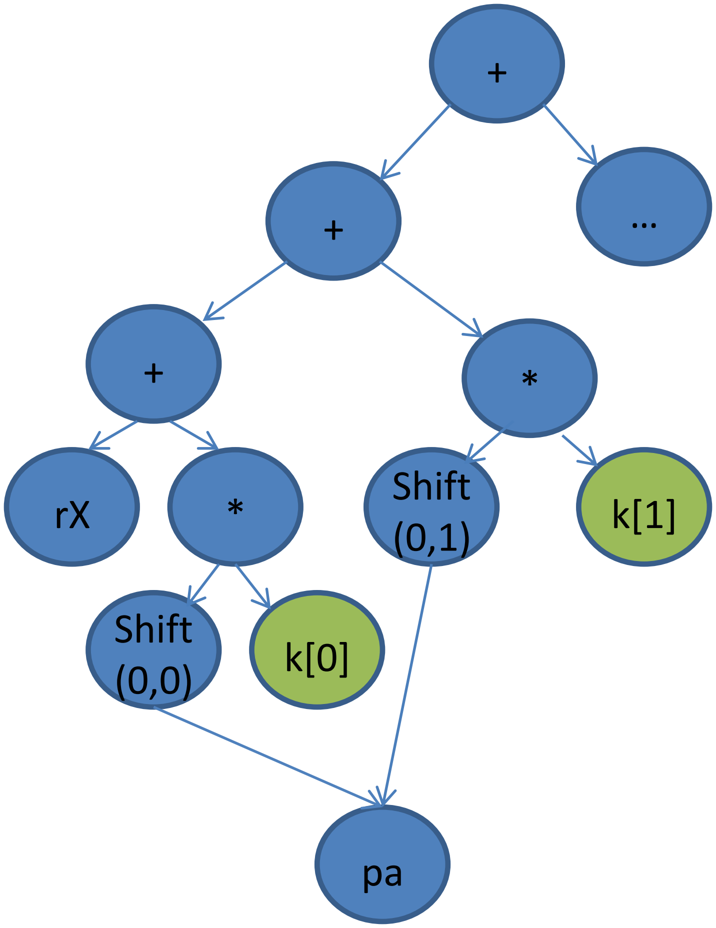
```
using System;
using Microsoft.ParallelArrays;

namespace AddArraysPointwiseMulticore
{
    class AddArraysPointwiseMulticore
    {
        static void Main(string[] args)
        {
            var x = new FloatParallelArray (new[] {1.0F, 2, 3, 4, 5});
            var y = new FloatParallelArray (new[] {6.0F, 7, 8, 9, 10});
            var multicoreTarget = new X64MulticoreTarget();
            var z = x + y;
            foreach (var i in multicoreTarget.ToArray1D (z))
                Console.Write(i + " ");
            Console.WriteLine();
        }
    }
}
```

```csharp
using System;
using Microsoft.ParallelArrays;

namespace AddArraysPointwiseFPGA
{
    class AddArraysPointwiseMulticore
    {
        static void Main(string[] args)
        {
            var x = new FloatParallelArray (new[] {1.0F, 2, 3, 4, 5});
            var y = new FloatParallelArray (new[] {6.0F, 7, 8, 9, 10});
            var fpgaTarget = new FPGATarget();
            var z = x + y;
            fpgaTarget.ToArray1D (z) ;
        }
    }
}
```

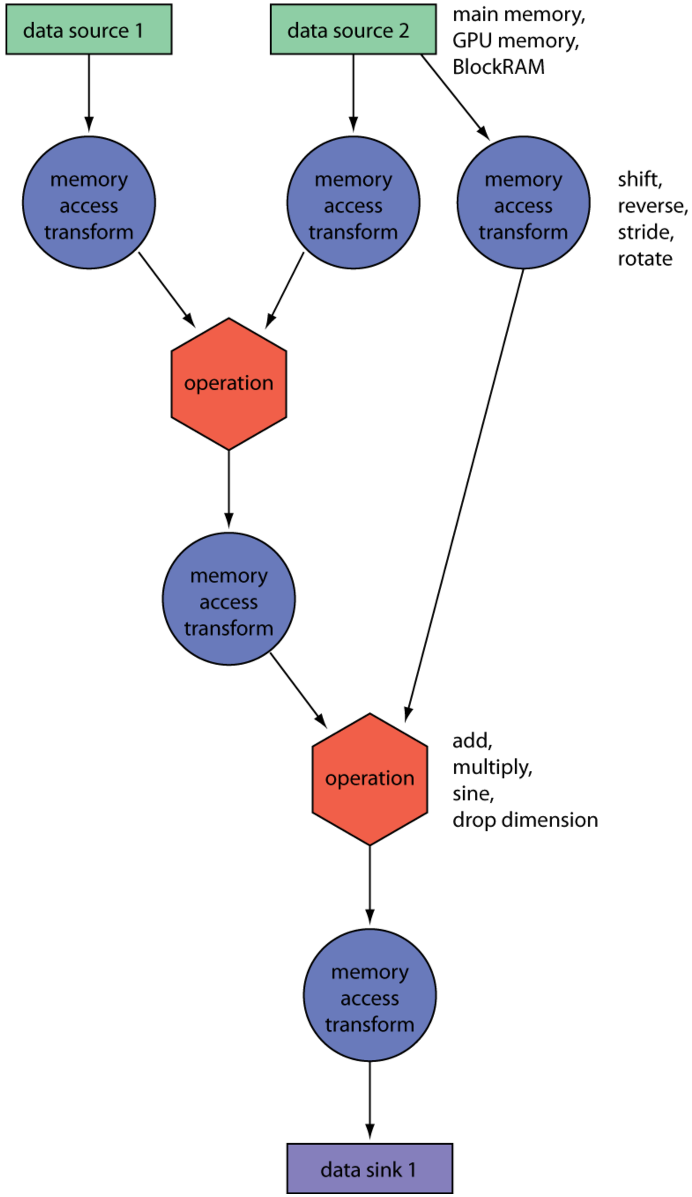
```vhdl
library ieee ;
use ieee.std_logic_1164.all ;
use work.addarrays_package.all ;
entity addarrays is
  port (signal clk, en, rst : in std_logic ;
        signal result : out float) ;
end entity addarrays ;

library ieee ;
use ieee.std_logic_unsigned.all ;
architecture accelerator of addarrays is
  attribute rom_style: string ;
  attribute ram_style: string ;
  attribute keep : string ;
  type net_1_array_type is array (0 to 4) of float ;
  signal net_1_array : net_1_array_type ; -- result  (*)
  attribute ram_style of net_1_array : signal is "block";
  signal net_1 : float ; -- Array input signal
  attribute keep of net_1 : signal is "true" ;
  type ext_1_array_type is array (0 to 4) of float ;
  signal ext_1_array : ext_1_array_type := (X"3f800000", X"40000000", X"40400000", X"40800000", X"40a00000") ;
  attribute rom_style of ext_1_array : signal is "block";
  signal ext_1_row_major : float := (others => '0') ; -- 1D array array output signal
  attribute keep of ext_1_row_major : signal is "true" ;
  signal net_2 : float ; -- Reference to array with external ID 1
  type ext_2_array_type is array (0 to 4) of float ;
  signal ext_2_array : ext_2_array_type := (X"40c00000", X"40e00000", X"41000000", X"41100000", X"41200000") ;
  attribute rom_style of ext_2_array : signal is "block";
  signal ext_2_row_major : float := (others => '0') ; -- 1D array array output signal
  attribute keep of ext_2_row_major : signal is "true" ;
  signal net_3 : float ; -- Reference to array with external ID 2
  signal net_4 : float := (others => '0') ;
  signal float_4_a : float := (others => '0') ;
  signal float_4_b : float := (others => '0') ;
  type ext_1_delayed_type is array (0 downto 0, 0 downto 0) of float ;
  signal ext_1_delayed : ext_1_delayed_type := (others => (others => (others => '0'))) ;
  type ext_2_delayed_type is array (0 downto 0, 0 downto 0) of float ;
  signal ext_2_delayed : ext_2_delayed_type := (others => (others => (others => '0'))) ;
```

```
    -- dimensions = (5) rank = 1
    variable col : integer := 0 ;
    variable col_shifted : integer ;
  begin
    wait until clk'event and clk='1' and en='1' ;
    if rst = '1' then
      col := 0 ;
    else
      col_shifted := col ;
      if col_shifted < 0 then
        col_shifted := 0 ;
      elsif col_shifted > 4 then
        col_shifted := 4 ;
      end if ;
      ext_2_delayed(0, 0) <= ext_2_array(col_shifted) ;
      if col < 4 then -- Advance along col
        col := col + 1 ;
      end if ; -- 1D array case
    end if ;
  end process gen_addr_ext_net_2_row_0 ;

  net_3 <= ext_2_delayed(0, 0) ;

  net_1_expr : process
  begin
    wait until clk'event and clk='1' and en='1' ;
    float_4_a <= net_2 ;
    float_4_b <= net_3 ;
  end process net_1_expr ;
  net_1 <= net_4 ;
  -- Section delay: 1 cycles
  result <= net_1 ;
  float_add_4 : floating_point_ieee_single_add port map (clk => clk, a => float_4_a, b => float_4_b, result => net_4) \
;

end architecture accelerator ;
```

```
open System
open Microsoft.ParallelArrays
let main(args) =
    let x = new FloatParallelArray (Array.map float32 [|1; 2; 3; 4;  5 |])
    let y = new FloatParallelArray (Array.map float32 [|6; 7; 8; 9; 10 |])
    let z = x + y
    use dx9Target = new DX9Target()
    let zv = dx9Target.ToArray1D(z)
    printf "%A\n" zv
    0
```




```
let rec convolve (shifts : int -> int [])
                 (kernel : float32 []) i
                 (a : FloatParallelArray)
  = let e = kernel.[i] * ParallelArrays.Shift(a, shifts i)
    if i = 0 then
      e
    else
      e + convolve shifts kernel (i-1) a
```


data source 1
data source 2
main memory,
GPU memory,
BlockRAM
memory access transform
memory access transform
memory access transform
shift,
reverse,
stride,
rotate
operation
memory access transform
operation
add,
multiply,
sine,
drop dimension
memory access transform
data sink 1

```
namespace AddArrays1D
{
    class AddArrays1D
    {
        static void Main(string[] args)
        {
            FloatParallelArray a = new FloatParallelArray(new float[] {1.0f, 2.0f, 3.0f, 4.0f});
            FloatParallelArray b = new FloatParallelArray(new float[] {5.0f, 6.0f, 7.0f, 8.0f });
            FloatParallelArray c = a + b;
            Target gpuTarget = new DX9Target();
            float[] result = gpuTarget.ToArray1D(c);
            foreach (float f in result)
                Console.Write(f + " ");
            Console.WriteLine();
        }
    }
}
```

```
ps_3_0
dcl_2d s0
dcl_texcoord0 v0.xy
dcl_2d s1
texld   r0, v0, s0
texld   r1, v0, s1
add    r1,   r0,   r1
mov oC0,   r1
```

```
FPA operator+(FPA a1, FPA a2);
```

```csharp
using System;
using Microsoft.ParallelArrays;
using FPA = Microsoft.ParallelArrays.FloatParallelArray;
namespace MultiplyAdd1D
{
    class MultiplyAdd1D
    {
        static void Main(string[] args)
        {
            FPA a = new FPA(new float[] { 1.0f, 2.0f, 3.0f, 4.0f });
            FPA b = new FPA(new float[] { 5.0f, 6.0f, 7.0f, 8.0f });
            FPA c = new FPA(new float[] { 9.0f, 10.0f, 11.0f, 12.0f });
            FPA d = ParallelArrays.MultiplyAdd(a, b, c);
            Target gpuTarget = new DX9Target();
            float[] result = gpuTarget.ToArray1D(d);
            foreach (float f in result)
                Console.Write(f + " ");
            Console.WriteLine();
        }
    }
}
```

```
ps_3_0
dcl_2d s0
dcl_texcoord0 v0.xy
dcl_2d s1
dcl_2d s2
texld   r0, v0, s0
texld   r1, v0, s1
texld   r2, v0, s2
mad    r2,   r0,   r1,   r2
mov oC0,   r2
```

```
static void Main(string[] args)
{
    Random random = new Random(42);
    FPA a = MakeRandomArray(3, 4, random);
    FPA b = MakeRandomArray(3, 4, random);
    FPA c = a + b;
    Target gpuTarget = new DX9Target();
    float[,] result = gpuTarget.ToArray2D(c);
    WriteArray(result);
}
```

```
ps_3_0
dcl_2d s0
dcl_texcoord0 v0.xy
dcl_2d s1
texld   r0, v0, s0
texld   r1, v0, s1
add    r1,   r0,   r1
mov oC0,   r1
```

```
int main()
{
     // Create a GPGPU computing resource
     DX9Target *tgtDX9 = CreateDX9Target() ;

     // Declare some sample input arrays
     float xvalues[5] = {1, 2, 3, 4,  5} ;
     float yvalues[5] = {6, 7, 8, 9, 10} ;

     // Create data-parallel versions of inputs
     FPA x = FPA(xvalues, 5) ;
     FPA y = FPA(yvalues, 5) ;

     // Specify data-parallel computation
     FPA z = x + y ; // Computation does not occur here...

     // Allocate space for the result array
     float* zvalues = (float*) malloc (5 * sizeof(float)) ;

     // Execute the data-parallel computation on the GPU
     tgtDX9->ToArray(z, zvalues, 5) ; // z = x + y is now evaluated

     // Write out the result
     for (int i = 0; i < 5; i++)
              cout << zvalues[i] << " " ;
     cout << endl ;
}
```

$$
\begin{aligned}
cnd(d) &= 1 - x \quad \text{if} \quad x < 0 \\
&= x \quad \text{otherwise} \\
x &= 1/\sqrt{2\pi} e^{-d^2/2} poly \\
poly &= horner(a, k) \\
horner(a, k) &= k(a_1 + k(a_2 + k(a_3 + k(a + 4 + ka_5)))) \\
k &= 1/(1 + 0.2316419\,|d|) \\
a &= [0.31938153, -0.356563782, 1.781477937, \\
&\quad -1.821255978, 1.330274429]
\end{aligned}
$$

$$
\begin{aligned}
V_{call} &= S \cdot cnd(d_1) - X \cdot e^{-rT} \cdot cnd(d2) \\
V_{put} &= X \cdot e^{-rT} \cdot cnd(-d_2) - S \cdot cnd(-d_1)
\end{aligned}
$$

```
// Horner approximaiton for the software version

let horner coe k
  = Array.foldBack (fun a b -> b * k + a) coe 0.0f
```

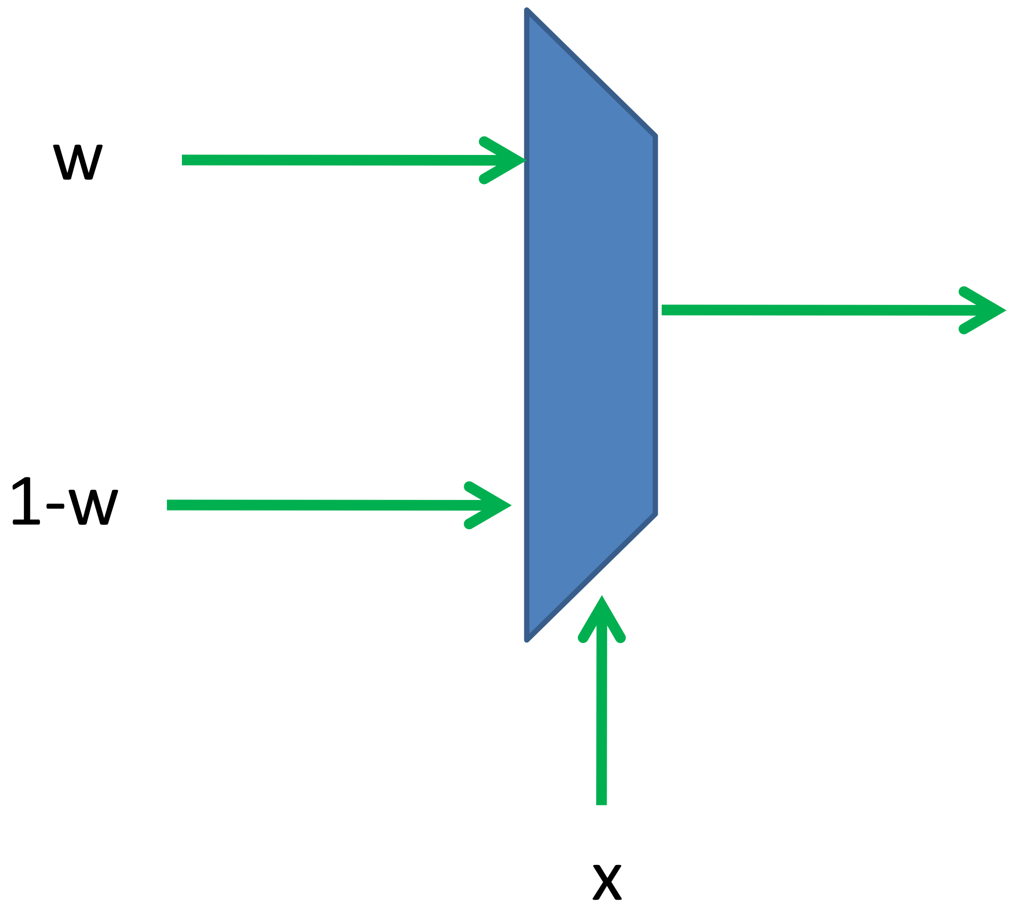
```
// Horner approximation for the Accelerator version

let horner2 coe (k : FPA)
  = let zero = new FPA (0.0f, k.Shape)
    Array.foldBack (fun a b -> b * k + a) coe zero
```

```
let cnd (x:float32)
  = let coe = [| 0.0f; 0.31938153f; -0.356563782f; 1.781477937f; -1.821255978f; 1.330274429f |]
    let l = abs x
    let k = 1.0f/(1.0f + 0.2316419f*l)
    let poly k = horner coe k
    let w = 1.0f - 1.0f/sqrt(2.0f * float32 Math.PI) *
            exp (-l*l/2.0f) * poly k
    if x < 0.0f then
      1.0f - w
    else
      w
```

```
let cndAccel (x : FPA) e
  = let coe = [| 0.0f; 0.31938153f; -0.356563782f; 1.781477937f; -1.821255978f; 1.330274429f |]
    let l = PA.Abs(x)
    let k = 1.0f / (1.0f + 0.2316419f * l)
    let poly = horner2 coe k
    let w = 1.0f - 1.0f / float32 (Math.Sqrt(2.0 * Math.PI)) *
                PA.Pow(e, -l * l / 2.0f) * poly
    PA.Select(x, w, 1.0f - w)
```

```
if x < 0.0f then
    1.0f - w
  else
    w
```

```
PA.Select(x, w, 1.0f - w)
```

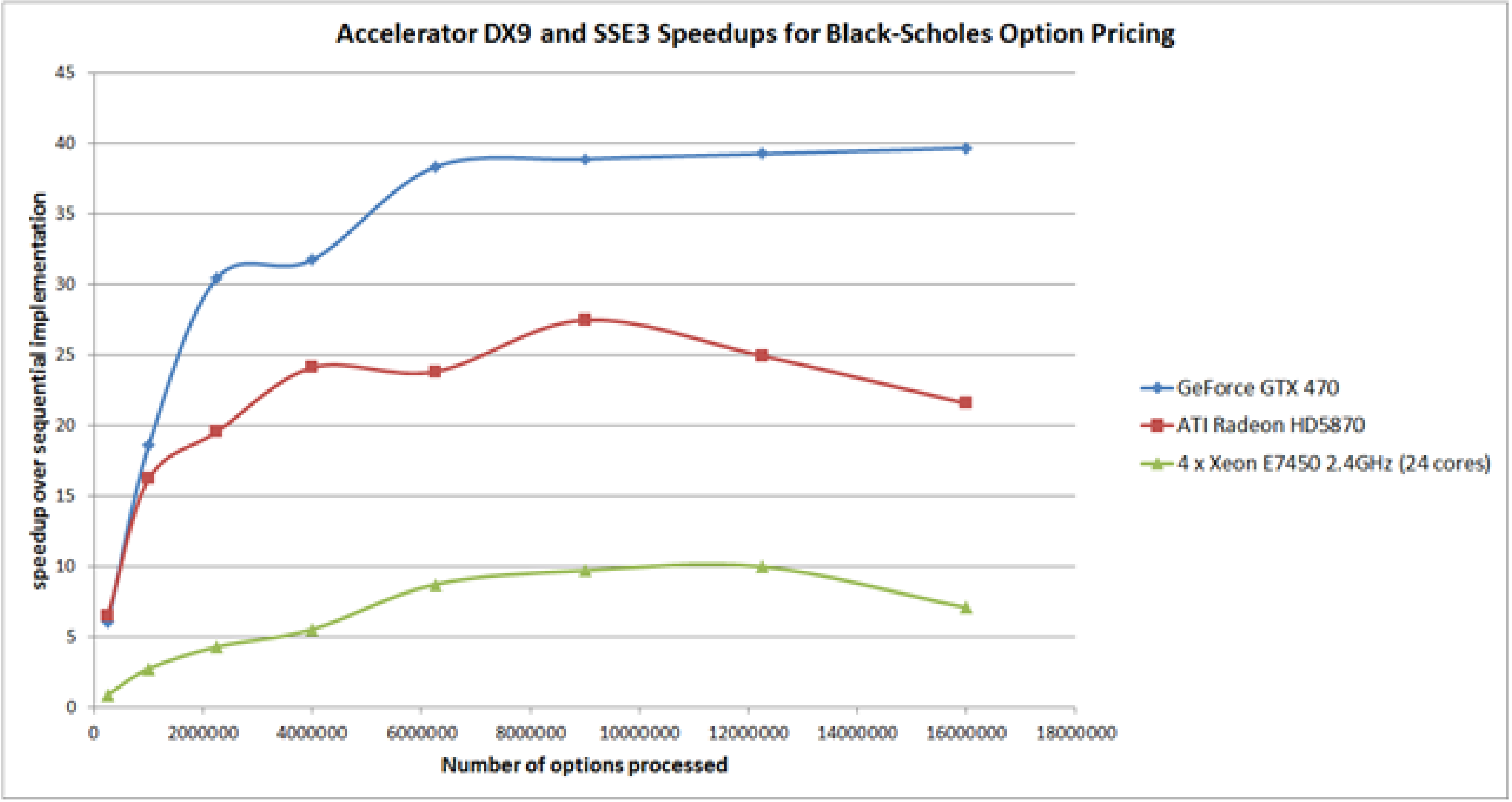

w
1-w
x

```
// Compute just the put option on the CPU
let optionPut s x t r v
  = let d1 = (log (s / x) + (r + v * v / 2.0f) * t) /
             (v * sqrt t)
    let d2 = d1 - v * sqrt t
    let expRT = exp (-r * t)
    x * expRT * cnd (-d2) - s * cnd (-d1)
```

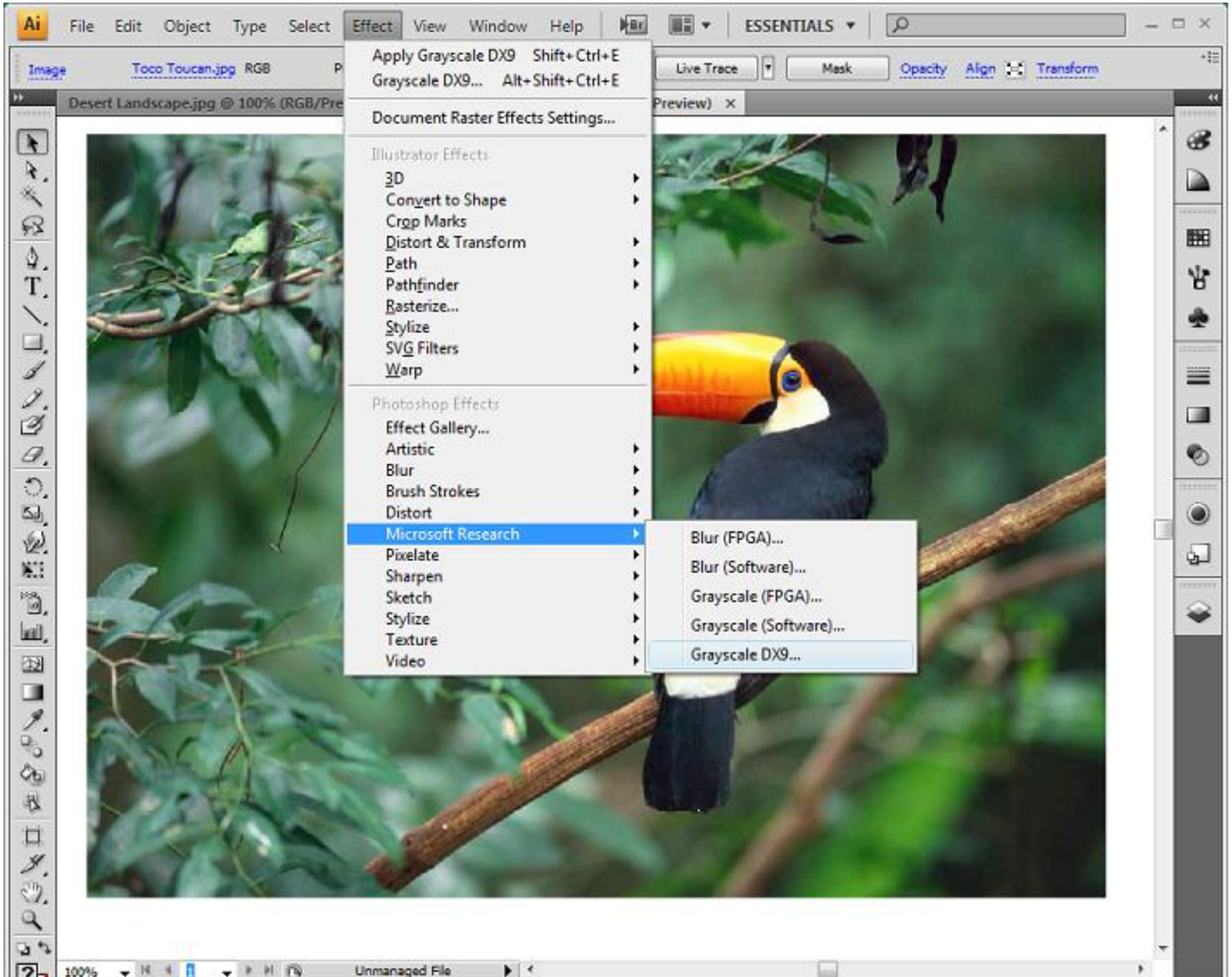
```
// Accelerator computation of the put option
let optionPutAccel (ss : FPA) (xs : FPA) (ts : FPA) r (v : float32)
  = let e = new FPA (float32 Math.E, ss.Shape)
    let d1 = loge (ss / xs) + ((r + v * v / 2.0f) * ts) /
             (v * PA.Sqrt(ts))
    let d2 = d1 - v * PA.Sqrt(ts)
    let expRT = PA.Pow(e, -r * ts)
    xs * expRT * cndAccel (-d2) e - ss * cndAccel (-d1) e
```

```
// A software version for only the call option for 2D input
let software2DPutOnly (ss : float32[,]) (xs : float32[,])
                      (ts : float32[,]) r v
  = Array2D.init (ss.GetLength(0)) (ss.GetLength(1))
      (fun i j ->  optionPut ss.[i,j] xs.[i,j] ts.[i,j] r v)
```

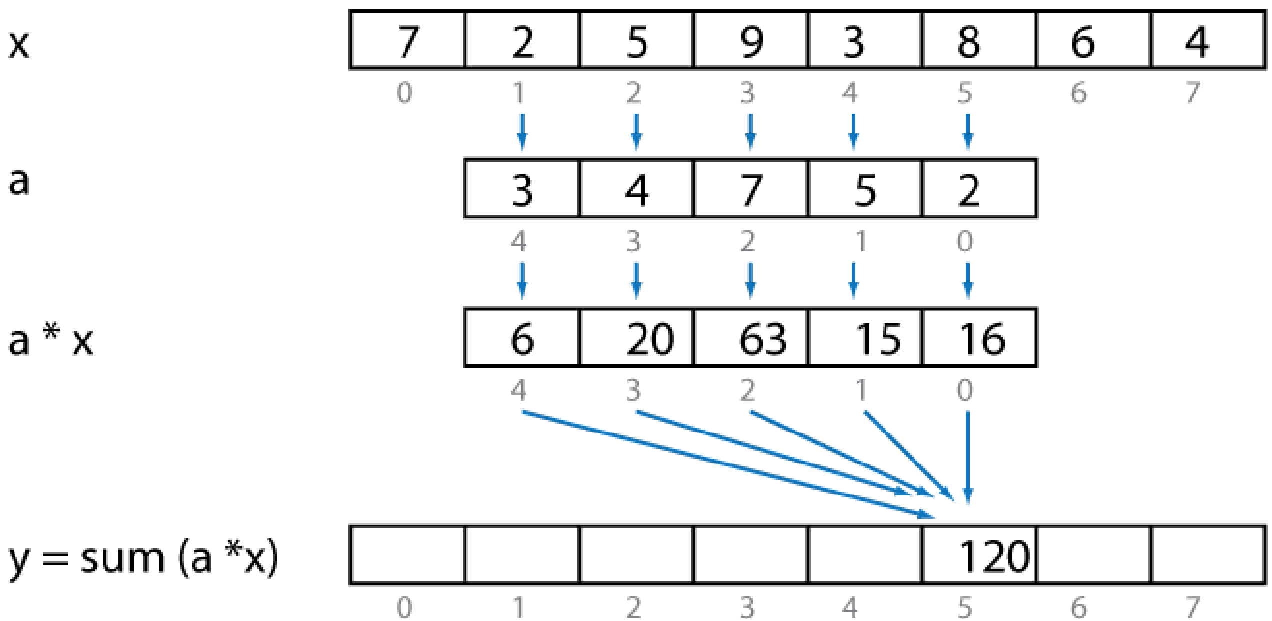
```
// Accelerator version for only the call option for 2D input
optionPutAccel ssA xsA tsA r v
```



Accelerator DX9 and SSE3 Speedups for Black-Scholes Option Pricing
speedup over sequential implementation
0
5
10
15
20
25
30
35
40
45
0
2000000
4000000
6000000
8000000
10000000
12000000
14000000
16000000
18000000
Number of options processed
GeForce GTX 470
ATI Radeon HD5870
4 x Xeon E7450 2.4GHz (24 cores)


File Edit Object Type Select Effect View Window Help
ESSENTIALS
Image
Toco Toucan.jpg RGB
Live Trace
Mask
Opacity
Align
Transform
Desert Landscape.jpg @ 100% (RGB/Pre
Preview)
Apply Grayscale DX9 Shift+Ctrl+E
Grayscale DX9... Alt+Shift+Ctrl+E
Document Raster Effects Settings...
Illustrator Effects
3D
Convert to Shape
Crop Marks
Distort & Transform
Path
Pathfinder
Rasterize...
Stylize
SVG Filters
Warp
Photoshop Effects
Effect Gallery...
Artistic
Blur
Brush Strokes
Distort
Microsoft Research
Pixelate
Sharpen
Sketch
Stylize
Texture
Video
Blur (FPGA)...
Blur (Software)...
Grayscale (FPGA)...
Grayscale (Software)...
Grayscale DX9...
100%
Unmanaged File

$$y_t = \sum_{k=0}^{N-1} a_k x_{t-k}$$



```
public static int[] SequentialFIRFunction(int[] weights, int[] input)
        {
            int[] window = new int[size];
            int[] result = new int[input.Length];
            // Clear to window of x values to all zero.
            for (int w = 0; w < size; w++)
                window[w] = 0;
            // For each sample...
            for (int i = 0; i < input.Length; i++)
            {
                // Shift in the new x value
                for (int j = size - 1; j > 0; j--)
                    window[j] = window[j - 1];
                window[0] = input[i];
                // Compute the result value
                int sum = 0;
                for (int z = 0; z < size; z++)
                    sum += weights[z] * window[z];
                result[i] = sum;
            }
            return result;
        }
```

$$y_t = \sum_{k=0}^{N-1} a_k x_{t-k}$$

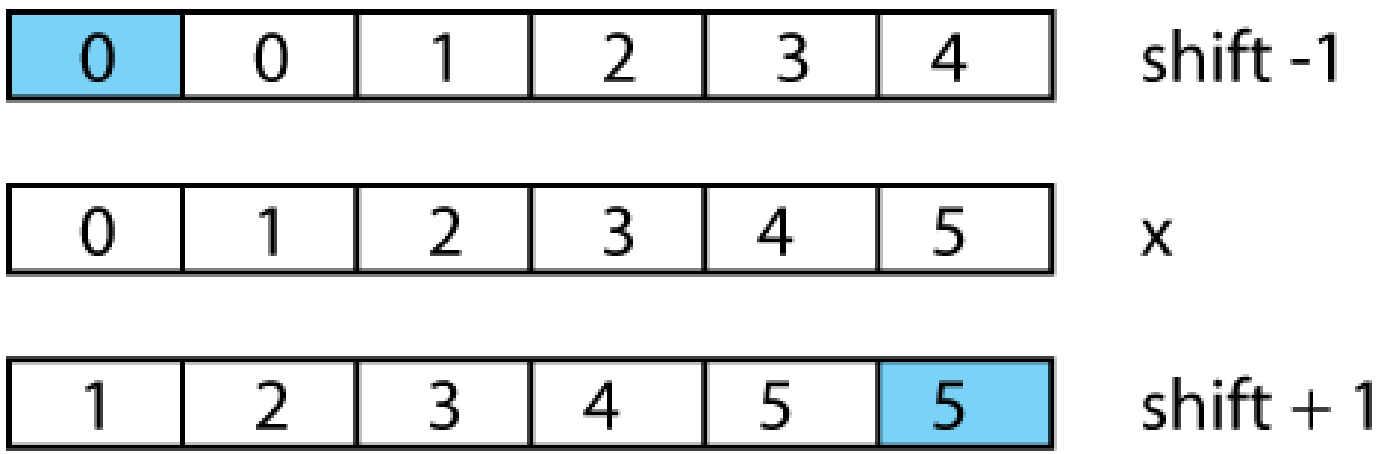
The Accidental Semi-colon

$y = [y[0], y[1], y[2], y[3], y[4], y[5], y[6], y[7]]$

$$
\begin{aligned}
y[0] &= a[0]x[0] + \mathbf{a[1]x[-1]} + a[2]x[-2] + a[3]x[-3] + a[4]x[-4] \\
y[1] &= a[0]x[1] + \mathbf{a[1]x[0]} + a[2]x[-1] + a[3]x[-2] + a[4]x[-3] \\
y[2] &= a[0]x[2] + \mathbf{a[1]x[1]} + a[2]x[0] + a[3]x[-1] + a[4]x[-2] \\
y[3] &= a[0]x[3] + \mathbf{a[1]x[2]} + a[2]x[1] + a[3]x[0] + a[4]x[-1] \\
y[4] &= a[0]x[4] + \mathbf{a[1]x[3]} + a[2]x[2] + a[3]x[1] + a[4]x[0] \\
y[5] &= a[0]x[5] + \mathbf{a[1]x[4]} + a[2]x[3] + a[3]x[2] + a[4]x[1] \\
y[6] &= a[0]x[6] + \mathbf{a[1]x[5]} + a[2]x[4] + a[3]x[3] + a[4]x[2] \\
y[7] &= a[0]x[7] + \mathbf{a[1]x[6]} + a[2]x[5] + a[3]x[4] + a[4]x[3]
\end{aligned}
$$

$$
\begin{aligned}
y &= [y[0], y[1], y[2], y[3], y[4], y[5], y[6], y[7]] \\
&= \mathrm{a[0]} * [\mathrm{x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7]}] + \\
&\quad \mathrm{a[1]} * [\mathrm{x[-1], x[0], x[1], x[2], x[3], x[4], x[5], x[6]}] + \\
&\quad \mathrm{a[2]} * [\mathrm{x[-2], x[-1], x[0], x[1], x[2], x[3], x[4], x[5]}] + \\
&\quad \mathrm{a[3]} * [\mathrm{x[-3], x[-2], x[-1], x[0], x[1], x[2], x[3], x[4]}] + \\
&\quad \mathrm{a[4]} * [\mathrm{x[-4], x[-3], x[-2], x[-1], x[0], x[1], x[2], x[3]}]
\end{aligned}
$$

shift (x, 0) = [7, 2, 5, 9, 3, 8, 6, 4] = x
shift (x, -1) = [7, 7, 2, 5, 9, 3, 8, 6]
shift (x, -2) = [7, 7, 7, 2, 5, 9, 3, 8]



$$
\begin{aligned}
y \quad &= [y[0], y[1], y[2], y[3], y[4], y[5], y[6], y[7]] \\
&= a[0] * [x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7]] + \\
&\quad a[1] * [x[-1], x[0], x[1], x[2], x[3], x[4], x[5], x[6]] + \\
&\quad a[2] * [x[-2], x[-1], x[0], x[1], x[2], x[3], x[4], x[5]] + \\
&\quad a[3] * [x[-3], x[-2], x[-1], x[0], x[1], x[2], x[3], x[4]] + \\
&\quad a[4] * [x[-4], x[-3], x[-2], x[-1], x[0], x[1], x[2], x[3]]
\end{aligned}
$$

$$
\begin{aligned}
y = \quad & a[0] * \text{shift}\ (x, 0) + \\
& a[1] * \text{shift}\ (x, -1) + \\
& a[2] * \text{shift}\ (x, -2) + \\
& a[3] * \text{shift}\ (x, -3) + \\
& a[4] * \text{shift}\ (x, -4)
\end{aligned}
$$

shift (x, 0)	7	2	5	9	3	8	6	4
shift (x, -1)	7	7	2	5	9	3	8	6
shift (x, -2)	7	7	7	2	5	9	3	8
shift (x, -3)	7	7	7	7	2	5	9	3
shift (x, -4)	7	7	7	7	7	2	5	9

a[0] * shift (x, 0)	14	4	10	18	6	16	12	8
a[1] * shift (x, -1)	35	35	10	25	45	15	40	30
a[2] * shift (x, -2)	49	49	49	14	35	63	21	56
a[3] * shift (x, -3)	28	28	28	28	8	20	36	12
a[4] * shift (x, -4)	21	21	21	21	21	6	15	27
	+	+	+	+	+	+	+	+
y =	147	137	118	106	115	120	124	133

```
using Microsoft.ParallelArrays;
using A = Microsoft.ParallelArrays.ParallelArrays;
namespace AcceleratorSamples
{
    public class Convolver
    {
```

```
for (int i = 0; i < a.Length; i++)
  ypar += a[i] * A.Shift(xpar, -i);
```

```
            var n = x.Length;
            var ypar = new FloatParallelArray(0.0f, new [] { n });
            for (int i = 0; i < a.Length; i++)
                ypar += a[i] * A.Shift(xpar, -i);
            float[] result = computeTarget.ToArray1D(ypar);
            return result;
        }
    }
}
```

shift (x, 0, 0)

7	2	5	9	3	8	6	4
2	8	7	4	8	9	3	5

→ a[0] * shift (x, 0, 0)

14	4	10	18	6	16	12	8
4	16	14	8	16	18	6	10

shift (x, 0, -1)

7	7	2	5	9	3	8	6
2	2	8	7	4	8	9	3

→ a[1] * shift (x, 0, -1)

35	35	10	25	45	15	40	30
10	10	40	35	20	40	45	15

shift (x, 0, -2)

7	7	7	2	5	9	3	8
2	2	2	8	7	4	8	9

→ a[2] * shift (x, 0, -2)

49	49	49	14	35	63	21	56
14	14	14	56	49	28	56	63

shift (x, 0, -3)

7	7	7	7	2	5	9	3
2	2	2	2	8	7	4	8

→ a[3] * shift (x, 0, -3)

28	28	28	28	8	20	36	12
8	8	8	8	32	28	16	32

shift (x, 0, -4)

7	7	7	7	7	2	5	9
2	2	2	2	2	8	7	4

→ a[4] * shift (x, 0, -4)

21	21	21	21	21	6	15	27
6	6	6	6	6	24	21	12

↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓ +

y[0] =	147	137	118	106	115	120	124	133
y[1] =	42	54	82	113	123	138	144	132

```
using Microsoft.ParallelArrays;
using A = Microsoft.ParallelArrays.ParallelArrays;
namespace AcceleratorSamples
{
    public class Convolver
    {
        public static float[,] Convolver1D_2DInput
         (Target computeTarget, float[] a, float[,] x)
```

```
var shiftBy = new [] {0, 0} ;
for (var i = 0; i < a.Length; i++)
 {
   shiftBy[1] = -i;
   ypar += a[i] * A.Shift(xpar, shiftBy);
}
```

```
            }
            var result = computeTarget.ToArray2D(ypar);
            return result;
        }
    }
}
```

```
ps_3_0
dcl_2d s0
dcl_texcoord0 v0.xy
dcl_texcoord1 v1.xy
dcl_texcoord2 v2.xy
dcl_texcoord3 v3.xy
dcl_texcoord4 v4.xy
def c0, 0.054489, 0.054489, 0.054489, 0.054489
def c1, 0.000000, 0.000000, 0.000000, 0.000000
def c2, 0.244201, 0.244201, 0.244201, 0.244201
def c3, 0.402620, 0.402620, 0.402620, 0.402620
texld   r0, v0, s0
mul       r0,   r0,   c0
add       r0,   c1,   r0
texld   r1, v1, s0
mul       r1,   r1,   c2
add       r1,   r0,   r1
texld   r2, v2, s0
mul       r2,   r2,   c3
add       r2,   r1,   r2
texld   r3, v3, s0
mul       r3,   r3,   c2
add       r3,   r2,   r3
texld   r4, v4, s0
mul       r4,   r4,   c0
add       r4,   r3,   r4
mov oC0,   r4
```

shift (x, 0, 0)	7	2	5	9	3	8	6	4
	2	8	7	4	8	9	3	5
shift (x, -1, 0)	7	2	5	9	3	8	6	4
	7	2	5	9	3	8	6	4
shift (x, -2, 0)	7	2	5	9	3	8	6	4
	7	2	5	9	3	8	6	4
shift (x, -3, 0)	7	2	5	9	3	8	6	4
	7	2	5	9	3	8	6	4
shift (x, -4, 0)	7	2	5	9	3	8	6	4
	7	2	5	9	3	8	6	4

→ a[0] * shift (x, 0, 0)	14	4	10	18	6	16	12	8
2	4	16	14	8	16	18	6	10
→ a[1] * shift (x, -1, 0)	35	10	25	45	15	40	30	20
5	35	10	25	45	15	40	30	20
→ a[2] * shift (x, -2, 0)	49	14	35	63	21	56	42	28
7	49	14	35	63	21	56	42	28
→ a[3] * shift (x, -3, 0)	28	8	20	36	12	32	24	16
4	28	8	20	36	12	32	24	16
→ a[4] * shift (x, -4, 0)	21	6	15	27	9	24	18	12
3	21	6	15	27	9	24	18	12
	+	+	+	+	+	+	+	+
y[0] =	147	42	105	189	63	168	126	84
y[1] =	137	54	109	179	73	170	120	86

```
using System;
using Microsoft.ParallelArrays;
namespace AcceleratorSamples
{
    public class Convolver2D
    {
        static FloatParallelArray convolve(Func<int, int[]> shifts, float[] kernel,
                                           int i, FloatParallelArray a)
        {
```

```
static FloatParallelArray convolveXY(float[] kernel,
                                     FloatParallelArray input)
{
  FloatParallelArray convolveX
    =    convolve(i => new [] { -i, 0 }, kernel,
                                          kernel.Length - 1, input);
  return convolve(i => new [] { 0, -i }, kernel,
                                          kernel.Length - 1, convolveX);
}
```

```
    return e + convolve(shifts, kernel, i - 1, a);
}
```

```
            var inputArray = new FloatParallelArray(inputData);
            var result = dx9Target.ToArray2D(convolveXY (testKernel, inputArray));
            for (var row = 0; row < inputSize; row++)
            {
                for (var col = 0; col < inputSize; col++)
                    Console.Write("{0} ", result[row, col]);
                Console.WriteLine();
            }
        }
    }
}
```

```
using System;
using System.Linq;
using Microsoft.ParallelArrays;
namespace AcceleratorSamples
{
```

```
static FloatParallelArray convolve(this FloatParallelArray a,
                                   Func<int, int[]> shifts,
                                   float[] kernel)
{ return kernel
        .Select((k, i) => k * ParallelArrays.Shift(a, shifts(i)))
        .Aggregate((a1, a2) => a1 + a2);
}

static FloatParallelArray convolveXY(this FloatParallelArray input,
                                     float[] kernel)
{ return input
        .convolve(i => new[] { -i, 0 }, kernel)
        .convolve(i => new[] { 0, -i }, kernel);
}
```

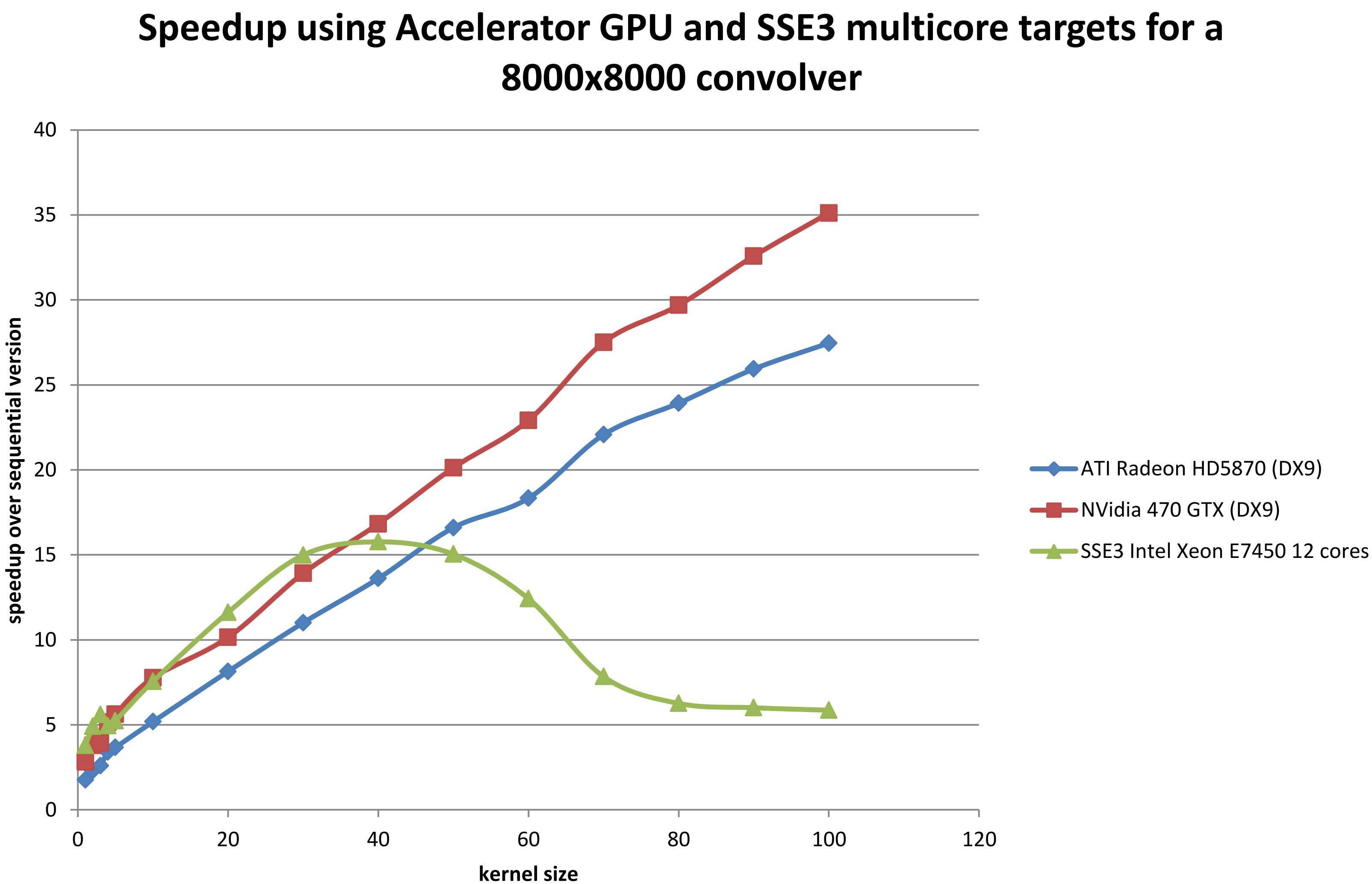
```
                for (int col = 0; col < inputSize; col++)
                    Console.Write("{0} ", result[row, col]);
                Console.WriteLine();
            }
        }
    }
}
```

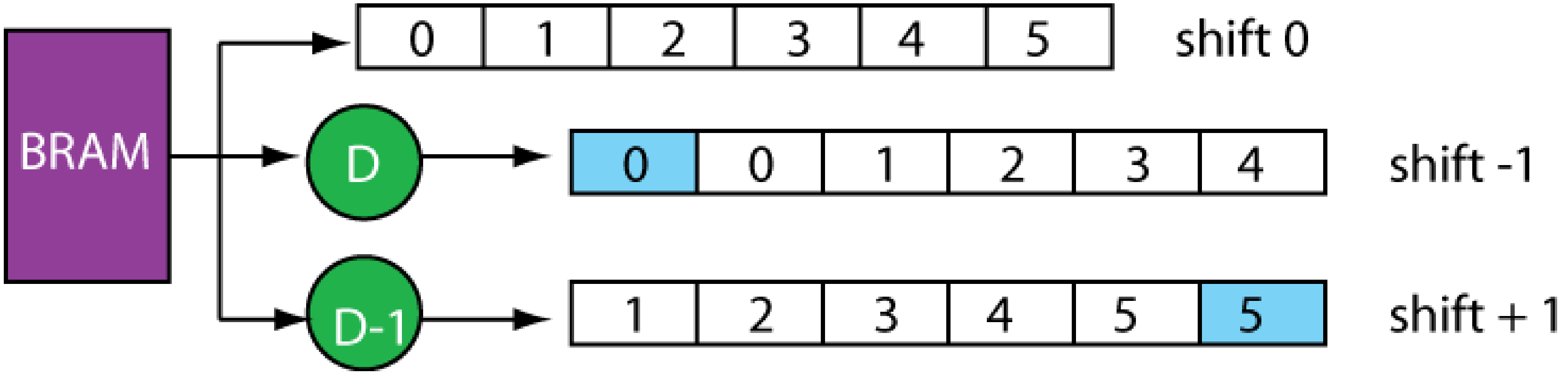
```
FPA ConvolveXY(Target &tgt, int height, int width, int filterSize, float filter[],
               FPA input, float *resultArray)
{
    // Convolve in X (row) direction.
    size_t dims[] = {height,width};
    FPA smoothX = FPA(0,dims, 2);
    intptr_t counts[] = {0,0};
    int filterHalf = filterSize/2;
    float scale;
    for (int i = -filterHalf; i <= filterHalf; i++)
    {
        counts[0] = i;
        scale = filter[i + filterHalf];
        smoothX += Shift(input, counts, 2) * scale;
    }

    // Convolve in Y (col) direction.
    counts[0] = 0;
    FPA result = FPA(0,dims, 2);
    for (int i = -filterHalf; i <= filterHalf; i++)
    {
        counts[1] = i;
        scale = filter[filterHalf + i];
        result += Shift(smoothX, counts, 2) * scale;
    }
    tgt.ToArray(result, resultArray, height, width, width * sizeof(float));
          return smoothX ;
};
```

```
open System
open Microsoft.ParallelArrays
[<EntryPoint>]
let main(args) =
    // Declare a filter kernel for the convolution
    let testKernel = Array.map float32 [| 2; 5; 7; 4; 3 |]
    // Specify the size of each dimension of the input array
    let inputSize = 10
    // Create a pseudo-random number generator
```

```
let convolveXY kernel input
  = // First convolve in the X direction and then in Y
    let convolveX = convolve (fun i -> [| -i; 0 |]) kernel
                                      (kernel.Length - 1) input
    let convolveY = convolve (fun i -> [| 0; -i |]) kernel
                                   (kernel.Length - 1) convolveX
    convolveY
```

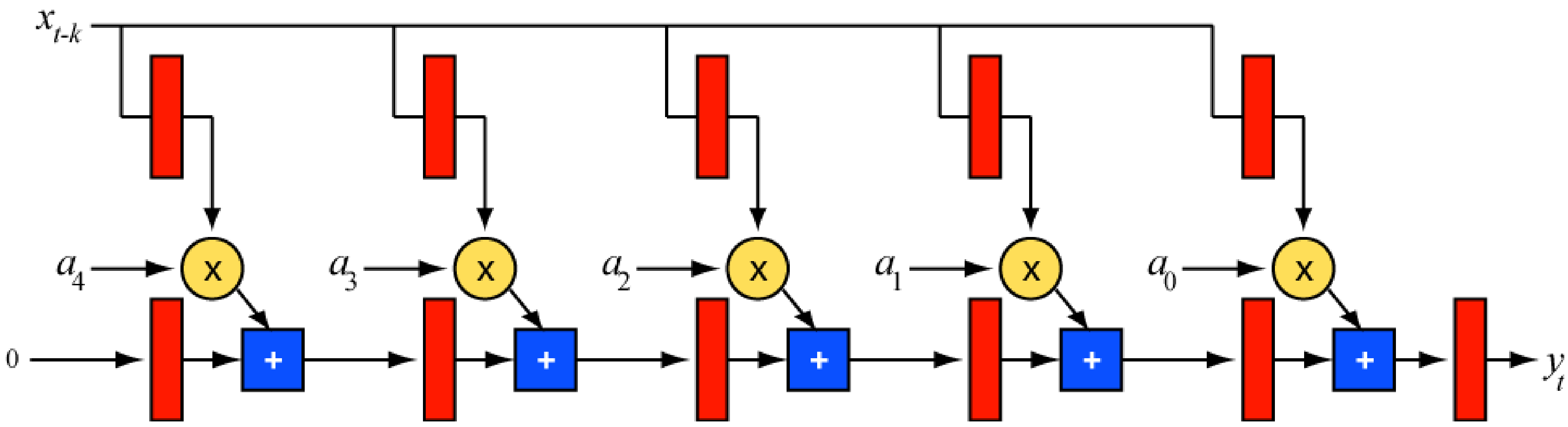
```
            e + convolve shifts kernel (i-1) a
    // Declare a 2D convolver
    let convolveXY kernel input
       = // First convolve in the X direction and then in the Y direction
         let convolveX = convolve (fun i -> [| -i; 0 |]) kernel (kernel.Length - 1) input
         let convolveY = convolve (fun i -> [| 0; -i |]) kernel (kernel.Length - 1) convolveX
         convolveY
    // Create a DX9 target and use it to convolve the test input
    use dx9Target = new DX9Target()
    let convolveDX9 = dx9Target.ToArray2D (convolveXY testKernel testArray)
    printfn "DX9: -> \r\n%A" convolveDX9
    0
```


Speedup using Accelerator GPU and SSE3 multicore targets for a 8000x8000 convolver
speedup over sequential version
kernel size
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
ATI Radeon HD5870 (DX9)
NVidia 470 GTX (DX9)
SSE3 Intel Xeon E7450 12 cores

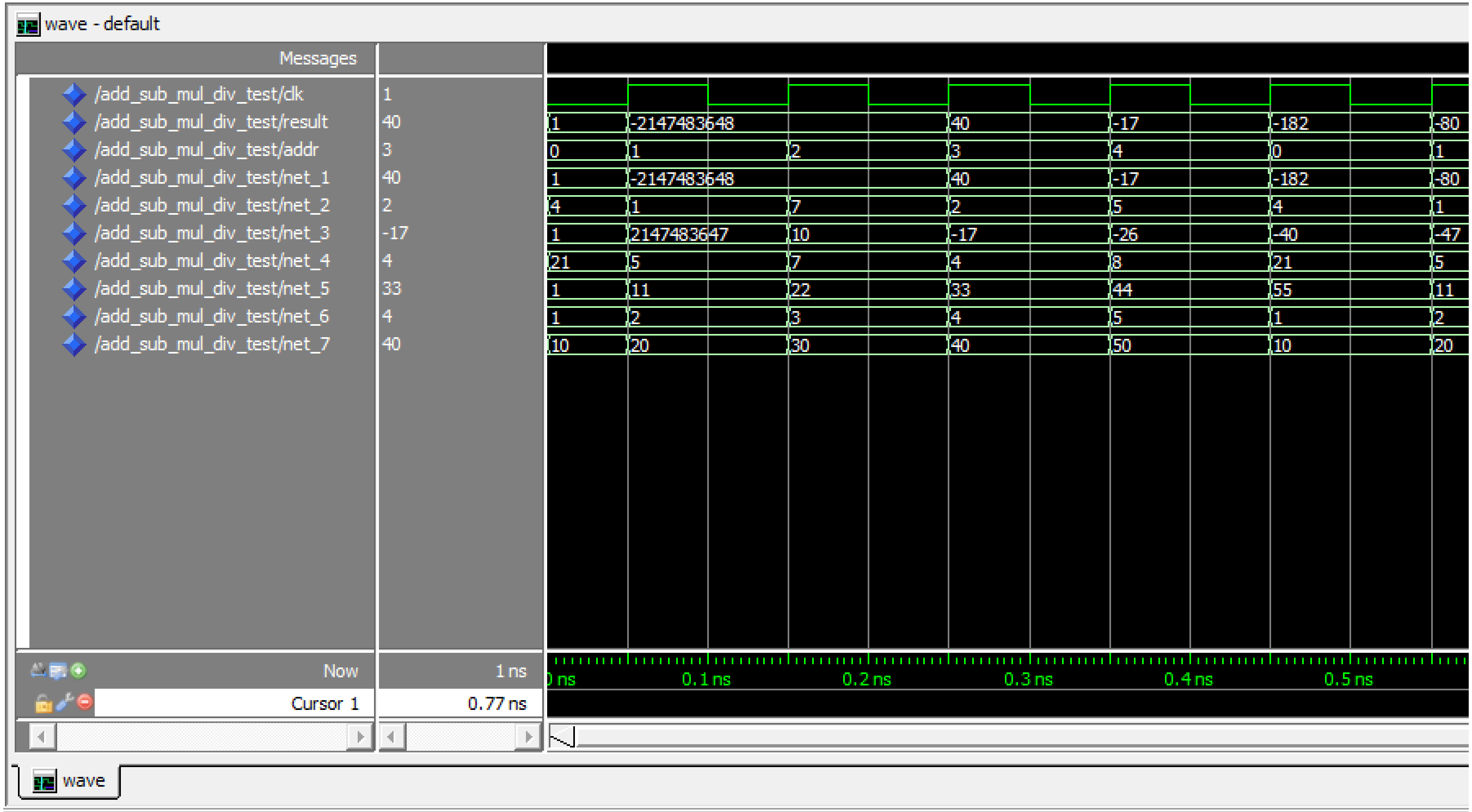


BRAM
D
D-1
0
1
2
3
4
5
shift 0
0
0
1
2
3
4
shift -1
1
2
3
4
5
5
shift + 1

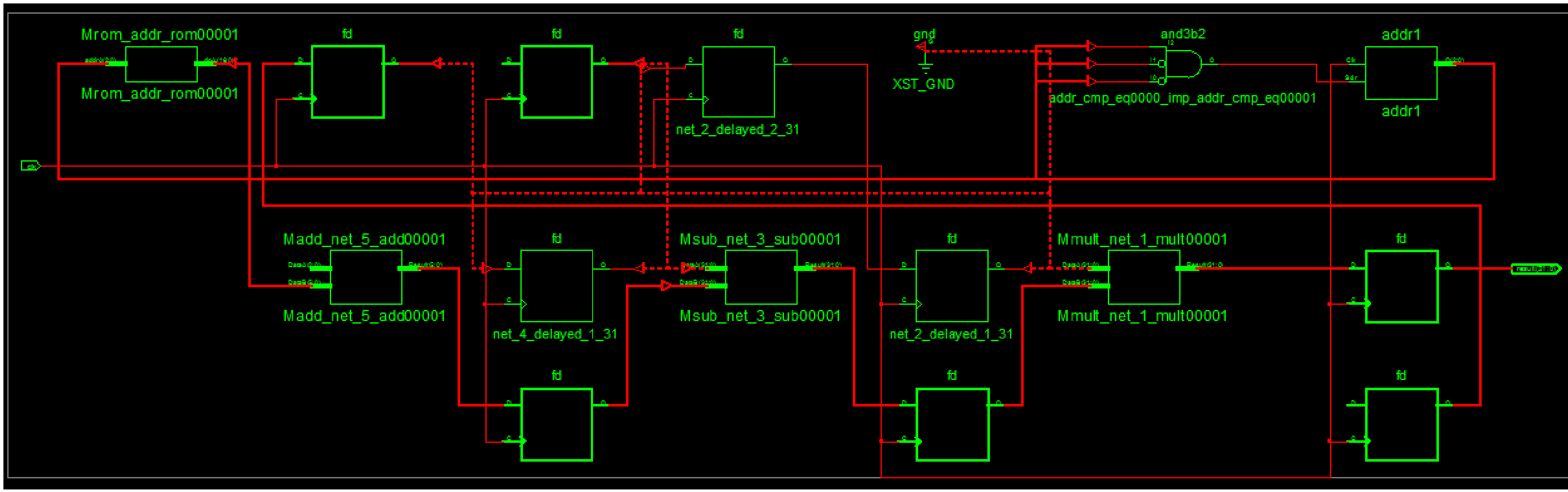
Convolver



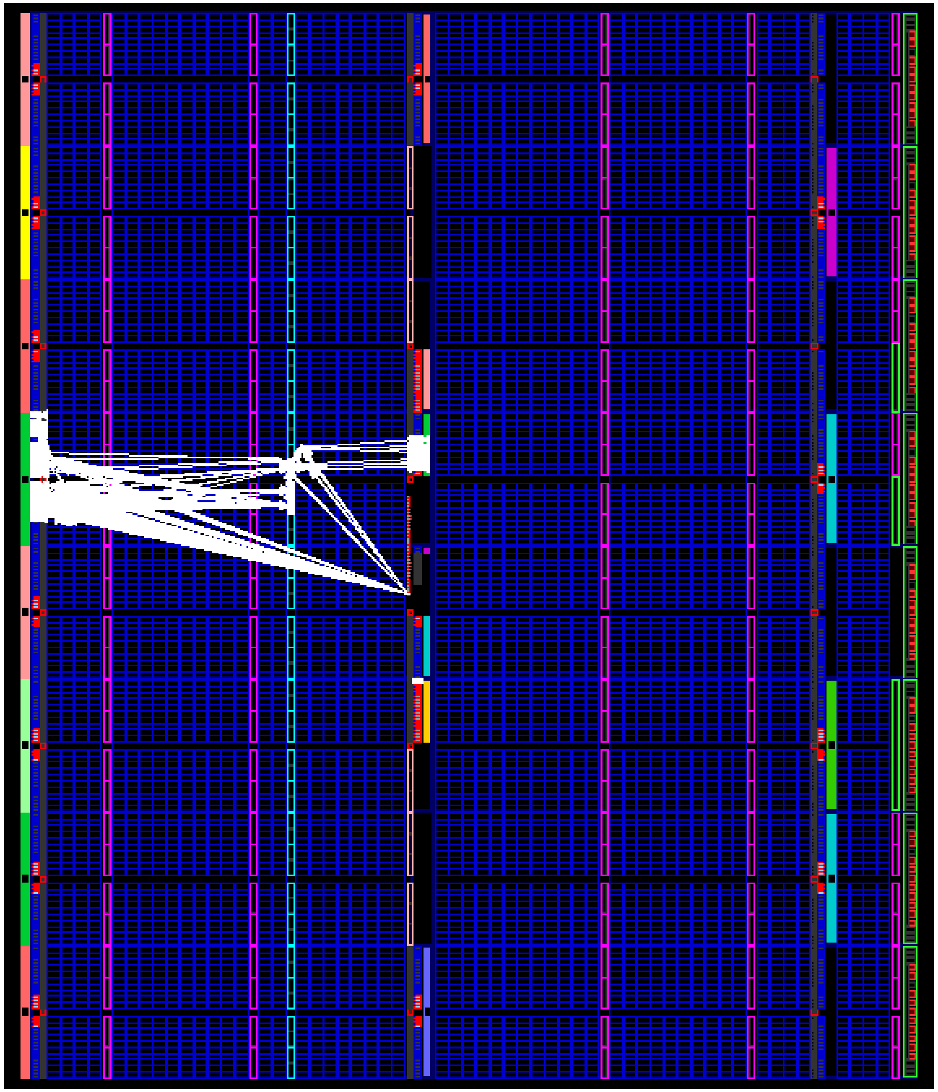
$$y_t = \sum_{k=0}^{N-1} a_k \, x_{t-k}$$

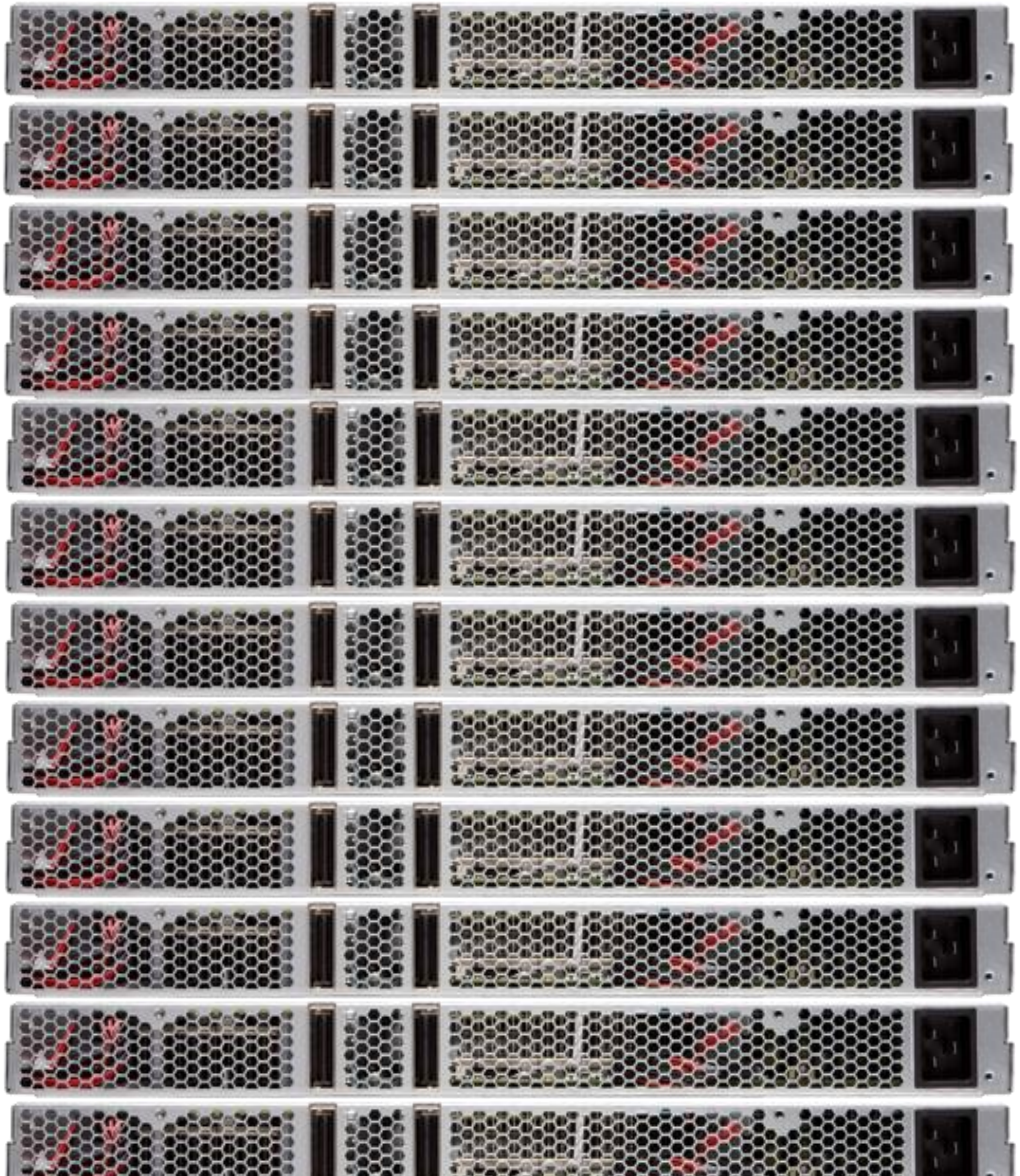


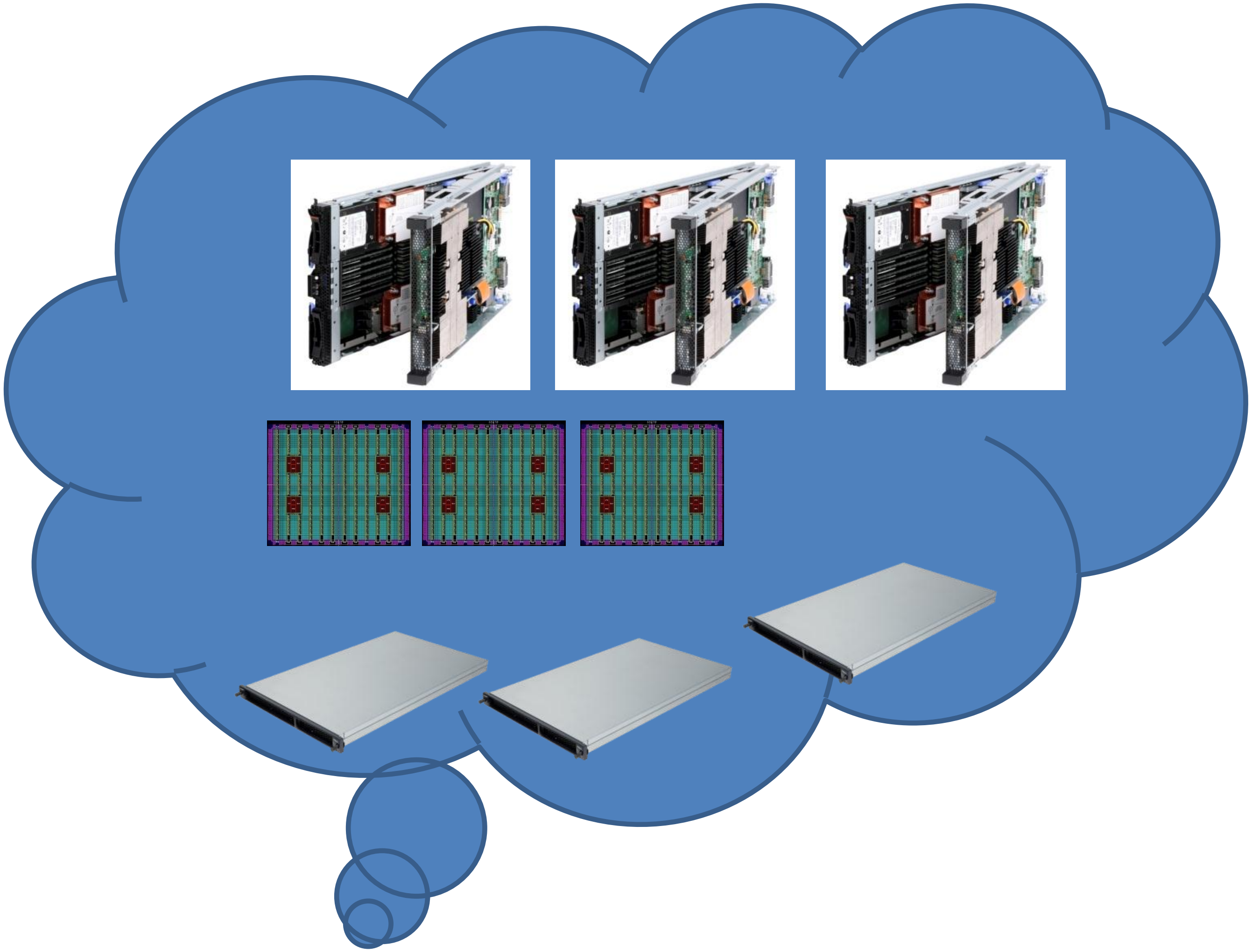
wave - default
Messages
/add_sub_mul_div_test/clk 1
/add_sub_mul_div_test/result 40
/add_sub_mul_div_test/addr 3
/add_sub_mul_div_test/net_1 40
/add_sub_mul_div_test/net_2 2
/add_sub_mul_div_test/net_3 -17
/add_sub_mul_div_test/net_4 4
/add_sub_mul_div_test/net_5 33
/add_sub_mul_div_test/net_6 4
/add_sub_mul_div_test/net_7 40
1 -2147483648 40 -17 -182 -80
0 1 2 3 4 0 1
1 -2147483648 40 -17 -182 -80
4 1 7 2 5 4 1
1 2147483647 10 -17 -26 -40 -47
21 5 7 4 8 21 5
1 11 22 33 44 55 11
1 2 3 4 5 1 2
10 20 30 40 50 10 20
Now 1 ns
Cursor 1 0.77 ns
0 ns 0.1 ns 0.2 ns 0.3 ns 0.4 ns 0.5 ns
wave

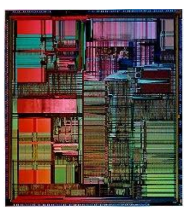


8.249ns max delay
3 x DSP48Es
63 slice registers
24 slice LUTs

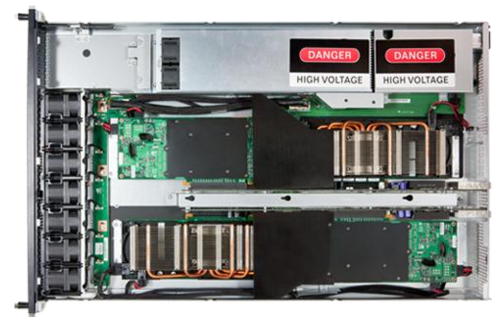


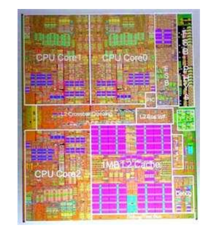


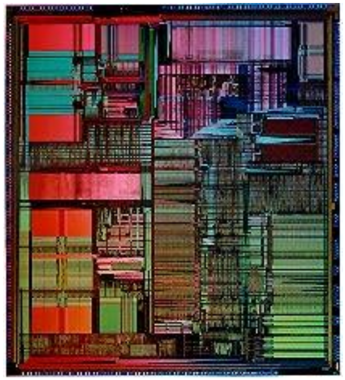




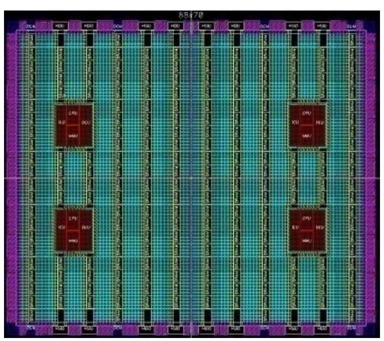
relocation via virtualization???





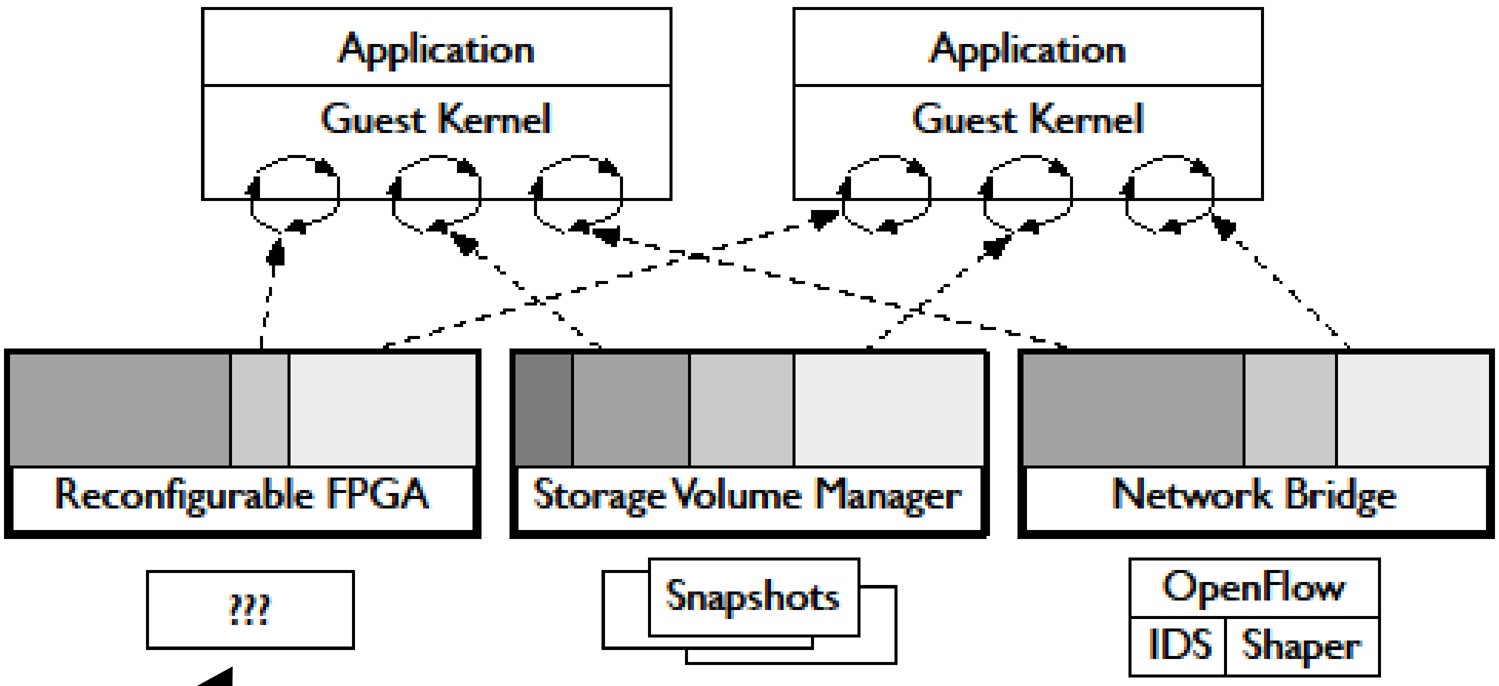


+ encryption + virtualization + data-processing



no standard ABI
no FPGA-kernel-userspace model
The cloud is just an extension of existing OS paradigms… FPGAs get left behind… they lack abstraction boundaries

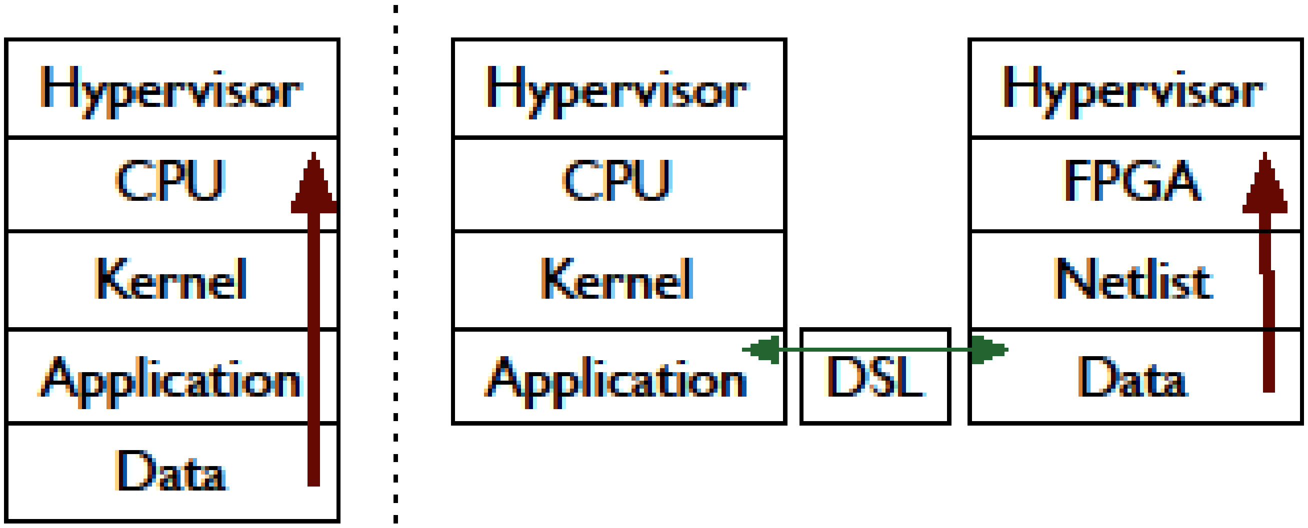
Split Trust



management domain?

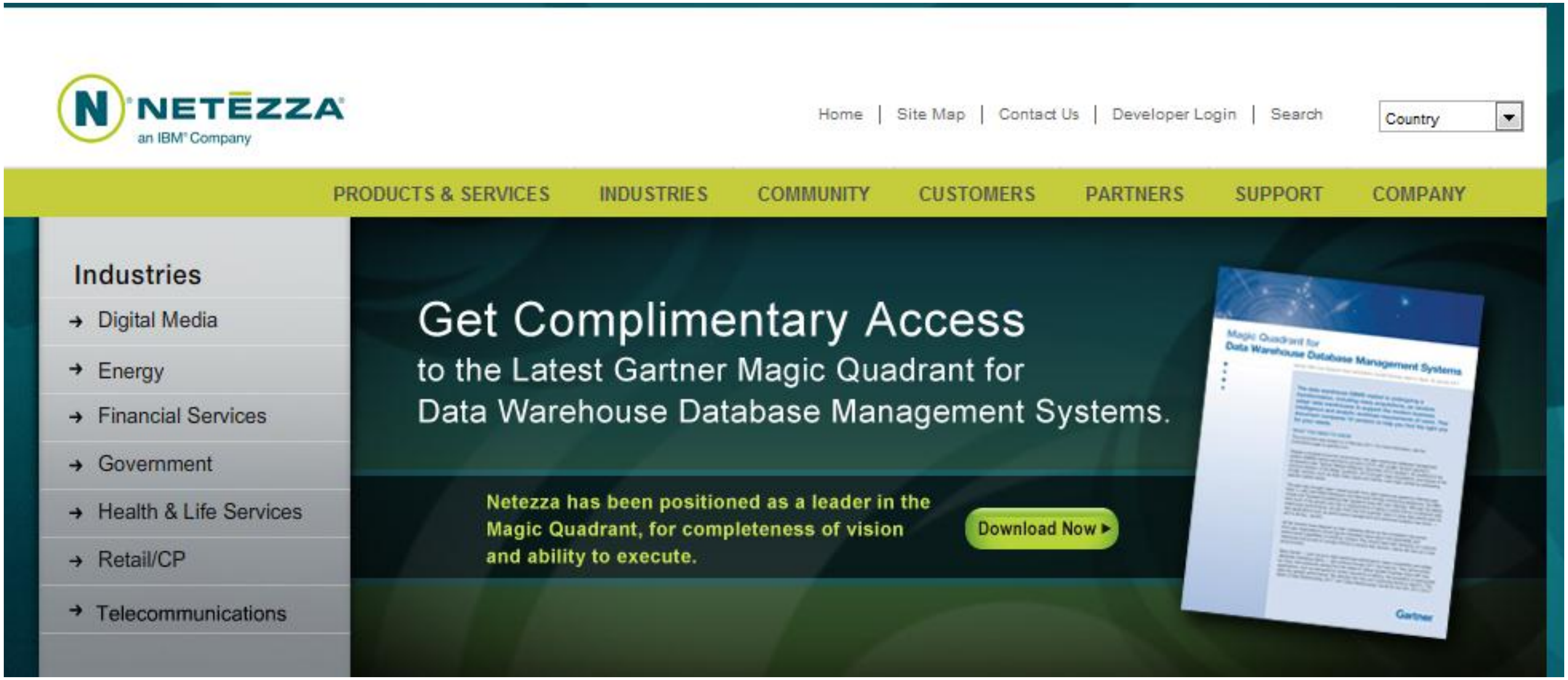
managing physical device
vs.
using a physical device

FPGAs Improve Cloud Security



Barrelfish Heterogeneous Operating System

IBM
United States [change]
Search
Home
Solutions
Services
Products
Support & downloads
My IBM
Welcome [IBM Sign in] [Register]
IBM Software > WebSphere >
WebSphere DataPower SOA Appliances
WebSphere software
WebSphere DataPower SOA Appliances
Library
Success stories
News
Events
Training and certification
Services
How to buy
Support
WebSphere DataPower SOA Appliance product video
→ Get video
WebSphere DataPower Architectural Design Patterns
→ Learn more
We're here to help
Easy ways to get the answers you need.
Call me


NETEZZA
an IBM Company
Home | Site Map | Contact Us | Developer Login | Search
Country
PRODUCTS & SERVICES
INDUSTRIES
COMMUNITY
CUSTOMERS
PARTNERS
SUPPORT
COMPANY
Industries
Digital Media
Energy
Financial Services
Government
Health & Life Services
Retail/CP
Telecommunications
Get Complimentary Access
to the Latest Gartner Magic Quadrant for
Data Warehouse Database Management Systems.
Netezza has been positioned as a leader in the
Magic Quadrant, for completeness of vision
and ability to execute.
Download Now

Sun Quick Links

ORACLE

(Sign In/Register for Account | Help) United States Communities I am a... I want to... Secure Search

Products and Services | Downloads | Store | Support | Education | Partners | About | Oracle Technology Network

Products and Services > Server and Storage Systems > Sun Servers

- Server and Storage Systems
  - Sun Servers
    - Blades
    - SPARC Servers
    - x86 Systems
    - Netra Carrier-Grade Systems
    - Rack Cabinets
    - Cooling Door Systems
    - Previous Products

Sun Servers

The best systems for enterprise environments

Sun Servers

Oracle 1-800-633-0738

- Have Oracle call you
- Global contacts
- Sales Chat Live

Expand All | Close All

Downloads

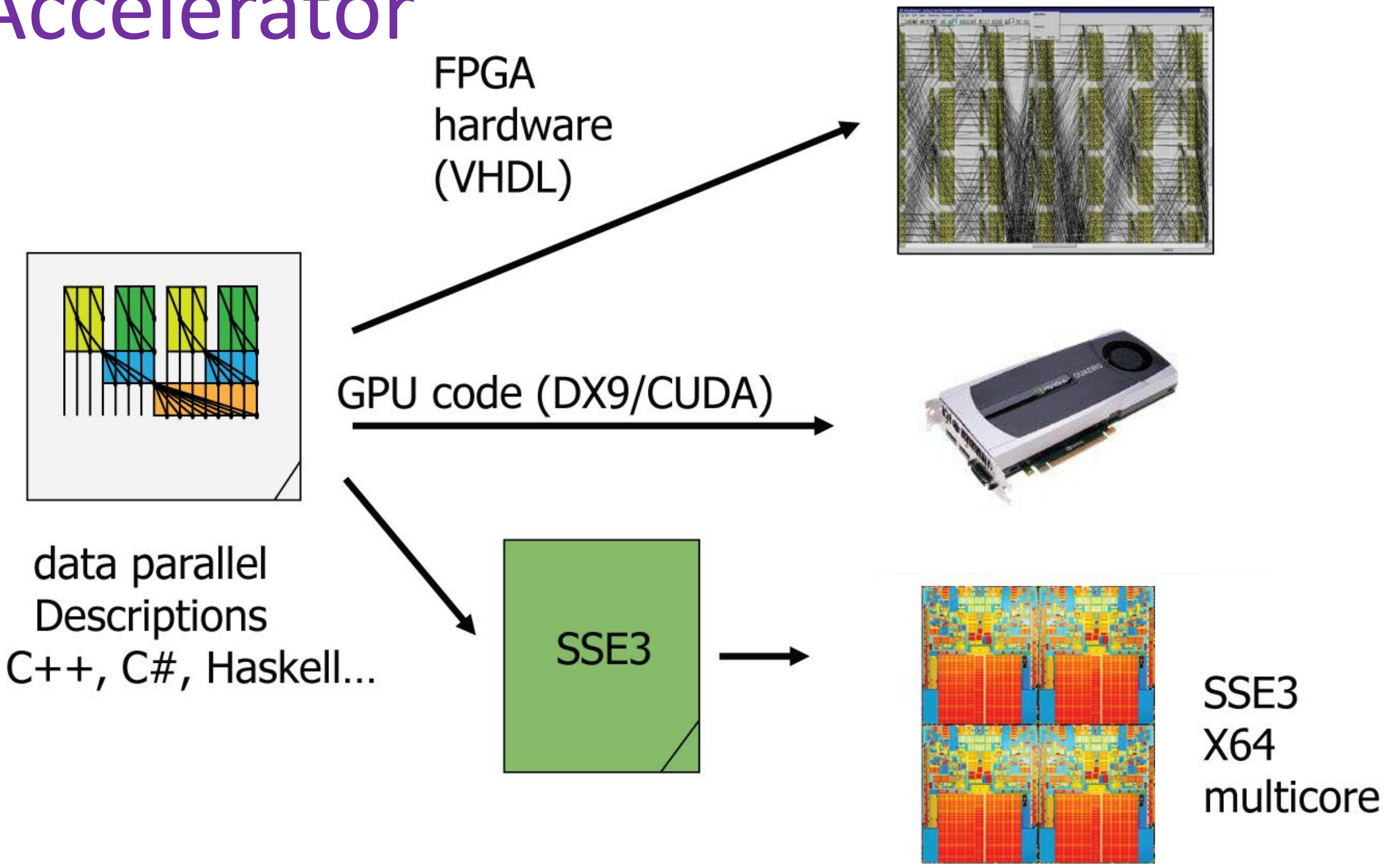
- Drivers and Firmware
- All Downloads

Close

News

Events

Accelerator



DOI:10.1145/1978542.1978558

Article development led by acmqueue
queue.acm.org

Heterogeneous systems allow us to target our programming to the appropriate environment.

BY SATNAM SINGH

Computing Without Processors

FROM THE PROGRAMMER'S perspective the distinction between hardware and software is being blurred. As programmers struggle to meet the performance requirements of today's systems they will face an ever increasing need to exploit alternative computing elements such as graphics processing units (GPUs), which are graphics cards subverted for data-parallel computing,[11] and field-programmable gate arrays (FPGAs), or soft hardware.

The current iteration of mainstream computing architectures is based on cache-coherent multicore processors. Variations on this theme include Intel's experimental Single-Chip Cloud Computer, which contains 48 cores that are not cache coherent. This path, however, is dictated by the end of frequency scaling rather than being driven by requirements about how programmers wish to write software.[4] ... systems are largely based on abstractions developed for writing operating systems (for example, locks and monitors). However, these are not the correct abstractions to use for writing parallel applications.

There are better ways to bake all that sand. Rather than composing many elements that look like regular CPUs, a better approach, from a latency and energy-consumption perspective, is to use a diverse collection of processing elements working in concert and tuned to perform different types of computation and communication. Large coarse-grain tasks are suitable for implementation on multicore pro-

Site Feeds
Twitter feed

BROWSE TOPICS
Programming Languages
Development
Databases
Privacy and Rights
Quality Assurance
System Evolution
Full List of Topics

COLUMNS
The Bikeshed
Kode Vicious
Curmudgeon
Opinion

DISCUSSIONS
CTO Roundtables
Case Studies
Interviews

LATEST COMMENTS
- "How pleasurable to see that good code parallels good design. Clr..."
Coding Guidelines

- "Firstly. Whatever style is used why not pick an IDE that automa ..."
Coding Guidelines

- "So what database or databases does Amazon

Computing without Processors

view issue

by Satnam Singh | June 27, 2011

Topic: Computer Architecture

View Comments Print +1 7
SHARE

Heterogeneous systems allow us to target our programming to the appropriate environment.

SATNAM SINGH, MICROSOFT RESEARCH CAMBRIDGE, UK

From the programmer's perspective the distinction between hardware and software is being blurred. As programmers struggle to meet the performance requirements of today's systems, they will face an ever increasing need to exploit alternative computing elements such as GPUs (graphics processing units), which are graphics cards subverted for data-parallel computing,[11] and FPGAs (field-programmable gate arrays), or soft hardware.

The current iteration of mainstream computing architectures is based on cache-coherent multicore processors. Variations on this theme include Intel's experimental Single-Chip Cloud Computer, which contains 48 cores that are not cache coherent. This path, however, is dictated by the end of frequency scaling rather than being driven by requirements about how programmers wish to write software.[4] The conventional weapons available for writing concurrent and parallel software for such multicore systems are largely based on abstractions developed for writing operating systems (e.g., locks and monitors). However, these are not the right abstractions to use for writing parallel applications.

There are better ways to bake all that sand. Rather than composing many elements that look like regular CPUs, a better approach, from a latency and energy-consumption perspective, is to use a diverse collection of processing elements working in concert and tuned to perform different types of computation and communication. Large coarse-grain tasks are suitable for implementation on multicore processors. Thousands of fine-grain data-parallel computations are

NEW QUEUE CONTENT ON SLASHDOT

- OCaml for the Masses
Yaron Minsky

- The Software Industry IS the Problem
Poul-Henning Kamp

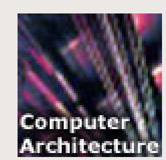
- Any Competing Whois Registry Model is Doomed
Paul Vixie

RELATED CONTENT

ON PLUG-INS AND EXTENSIBLE ARCHITECTURES
Extensible application architectures such as Eclipse offer many advantages, but one must be careful to avoid "plug-in hell."
Dorian Birsan

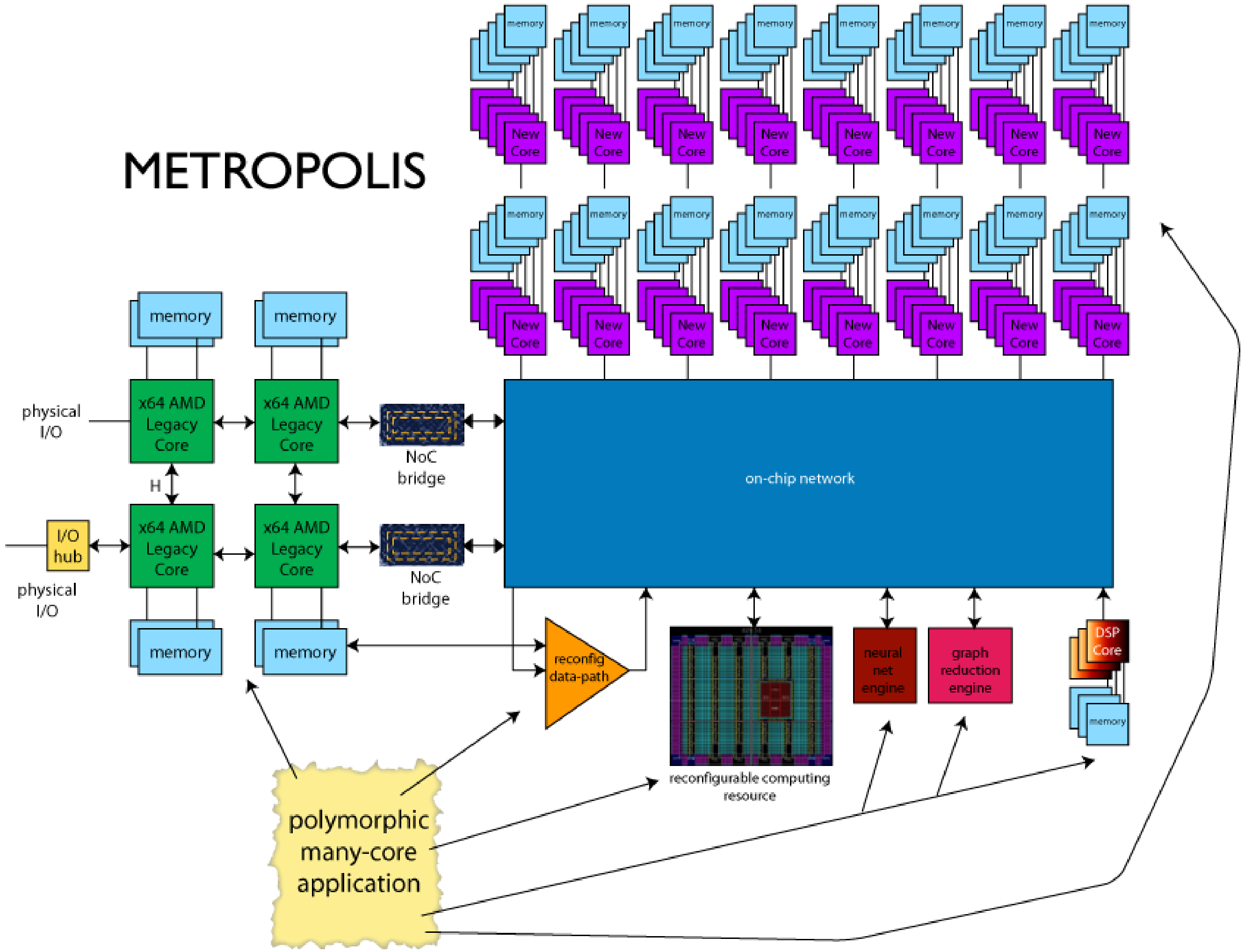
BROWSE THIS TOPIC:

COMPUTER ARCHITECTURE




+Satnam Gmail Calendar Documents Photos Sites Web More
Satnam Singh
Google+
Search Google+
Satnam Singh
Misanthrope programmer.
Edit Profile
Posts About Photos Videos +1's
View profile as...
Send a message
Send an email
In Satnam's circles (225)
Satnam Singh - 14 Nov 2011 (edited) - Public
Several people have complained that I am not answering my Microsoft email. My Microsoft email got cut off over a month ago and email sent to satnams@microsoft.com disappears silently and the sender does not get a message delivery failure email. This is very frustrating and annoying behaviour by Microsoft and there is nothing I can do about it. Please send email to satnam@raintown.org instead (or s.singh@acm.org or s.singh@ieee.org).
+1 - Comment - Share
Suhaib Fahmy - It's a feature of Exchange ;)
Yesterday 09:22
Add a comment...

METROPOLIS



memory
New Core
memory
x64 AMD Legacy Core
physical I/O
H
I/O hub
NoC bridge
on-chip network
reconfig data-path
reconfigurable computing resource
neural net engine
graph reduction engine
DSP Core
polymorphic many-core application