FPGA Acceleration of Bioinformatics on the XD1

A Case Study

Jim Maltby, Steve Margerm, Jeff Chow

Cray Inc.

Outline of Talk

- FPGAs and Bioinformatics on the XD1
- Smith-Waterman Algorithm
- Implementation Decisions
- Logic Design
- Performance Predictions
- Conclusions

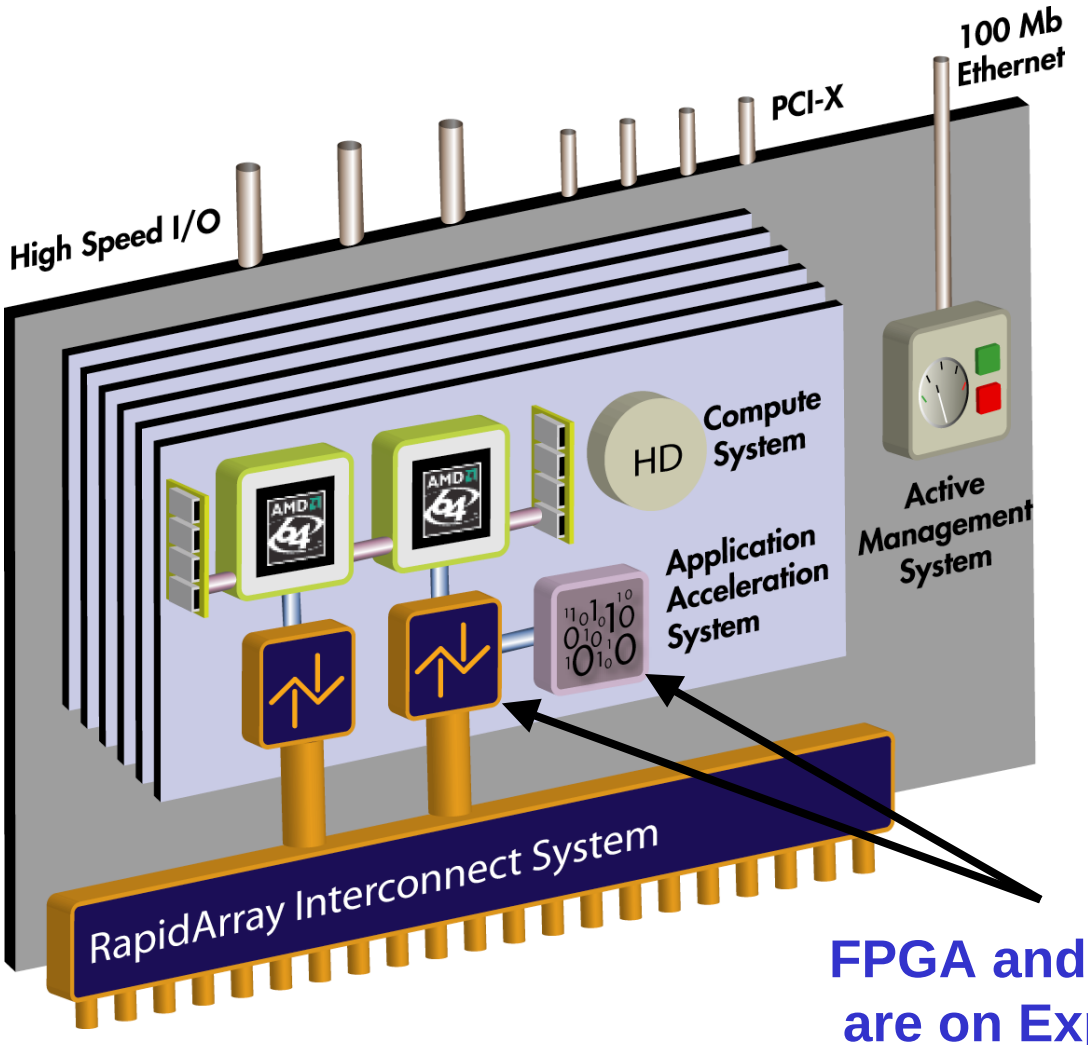
FPGAs and Bioinformatics

- When to use FPGA on Cray XD1?
  - Code has one or a few bottlenecks
    - **They must use a significant fraction of runtime**
    - **They should not be too complex**
  - The bottlenecks are a good fit for FPGA
    - **Lots of inherent parallelism**
    - **Lots of use per data touch and fit in QDRAM**
      - **Like fft butterfly**
    - **Bit fiddling and subword operations are best**
- Bioinformatics is a good fit
  - **Genomic data is typically stored in 2 to 8 bit quantities**
  - **Bioinformatics algorithms are often highly parallel**
  - **There is little floating point**
  - **Most ops are additions, masking or comparison**

Why the XD1?

- One FPGA per Blade (two Opterons)
- Close coupling between FPGA and Opterons, direct access to network
  - Many other solutions have a cluster of FPGAs on a remote (PCI-X?) link
- High-speed memory transfers to and from FPGA with high-level API
  - Read/write registers
  - Memory mapping

Cray XD1 System Architecture



Compute
- **12 AMD Opteron 32/64 bit, x86 processors**
- **High Performance Linux**

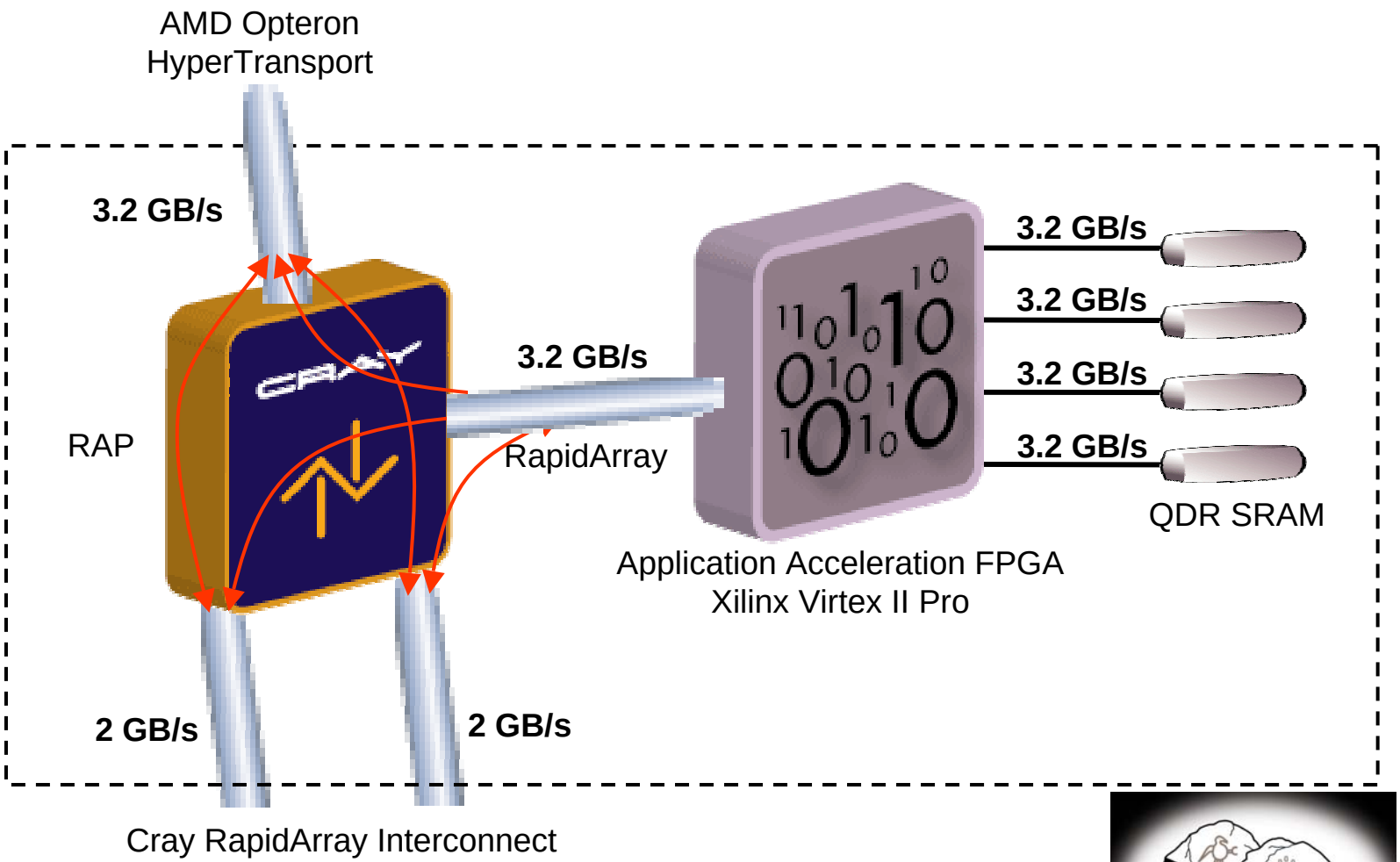
RapidArray Interconnect
- **12 communications processors**
- **1 Tb/s switch fabric**

Active Management
- **Dedicated processor**

Application Acceleration
- **6 co-processors**

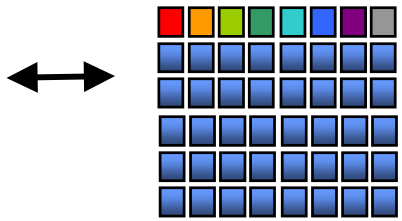
FPGA and 2nd RAP are on Expansion Module

High Bandwidth from FPGA to system



Two processor choices

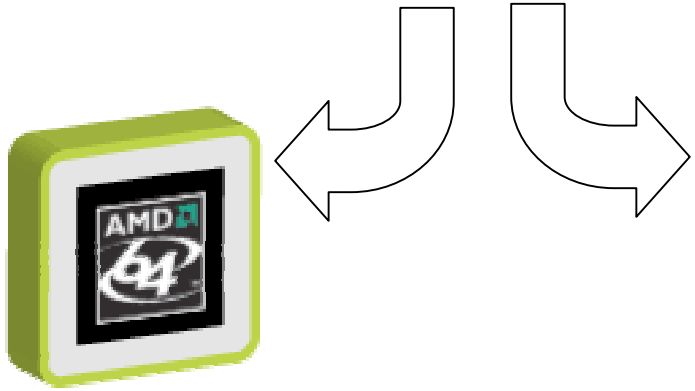
```
...
do for each array element
        .
        .
        .
end
...
```



200 MHz

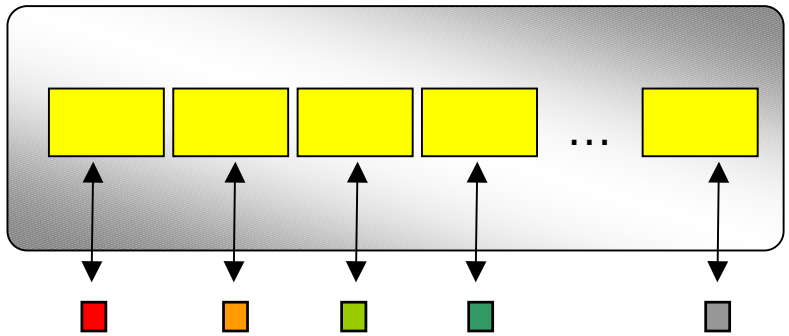
16 MB

parallel



Compute Processor

Application Acceleration FPGA



2.4 GHz

4-8 GB

serial

Map your algorithms to the appropriate processor

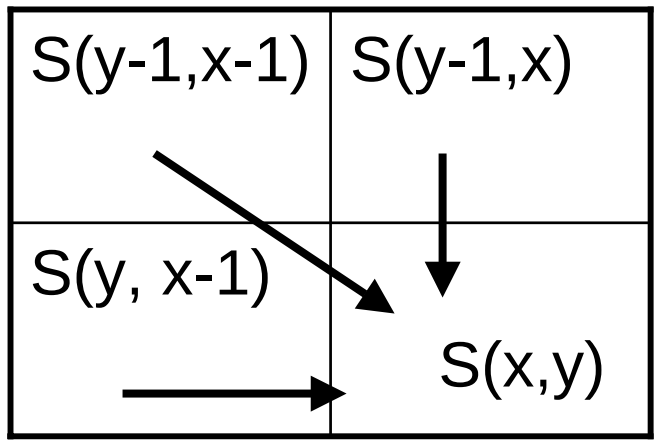
The Smith-Waterman Algorithm

- Genomic comparison and alignment algorithm
  - Similar to BLAST, but 10x slower
  - Provably optimum- the “gold standard” for alignment algorithms
  - Based on Dynamic Programming
- Two-step process
  - Create scoring matrix and find maximum score
    - “forward pass”
  - Work back to determine alignment
    - “traceback”

Smith-Waterman formulation

- Create a “scoring matrix” with one string along horizontal axis and one along vertical axis
- Calculate each cell according to the values of its neighbors above and to the left, according to the formula below:

$$S(y,x) = \max\begin{Bmatrix} 0, \\ S(y-1,x-1) + Sub(cy,cx), \\ S(y,x-1) - eog, \\ S(y-1,x) - eog, \\ H(x-1) - e, \\ V(y-1) - e \end{Bmatrix} \quad for\, 1 \le x \le s1, 1 \le y \le s2$$



where:

$$H(y,x) = \max\begin{Bmatrix} H(y,x-1) - e \\ S(y,x-1) - eog \end{Bmatrix} \quad for\ a\ given\ y$$

$$V(y,x) = \max\begin{Bmatrix} V(y-1,x) - e \\ S(y-1,x) - eog \end{Bmatrix} \quad for\ a\ given\ x$$

and:

$$S(0,x) = S(y,0) = 0$$

$$H(0) = V(0) = -eog$$

S-W Example

- Alignment of ACGAACCCTTGC and ACGTATGC
- Maximum score is 8 (lower right corner of matrix)
- Trace back along the path that led to the optimum score (traceback information not shown)
- Final alignment is shown below:

Scoring matrix

	0	A	C	G	T	A	T	G	C
0	0	0	0	0	0	0	0	0	0
A	0	2	0	0	0	2	0	0	0
C	0	0	4	2	1	0	1	0	2
G	0	0	2	6	4	3	2	3	1
A	0	2	1	4	5	6	4	3	2
A	0	2	1	3	3	7	5	4	3
C	0	2	4	2	2	5	6	4	6
C	0	0	2	3	1	4	4	5	6
C	0	0	2	1	2	3	3	3	7
T	0	0	0	1	3	2	5	3	5
T	0	0	0	0	3	2	4	4	4
G	0	0	0	2	1	2	2	6	4
C	0	0	2	0	1	0	1	4	8

Final Alignment

A	C	G	A	A	C	C	C	T	T	G	C
|	|	|		|					|	|	|
A	C	G	T	A	-	-	-	-	T	G	C

Implementation Decisions

- How to partition algorithm between Opteron and FPGA
  - Identify code regions which are highly parallel, but “small” enough to fit many copies in FPGA
  - Remember FPGA has small but very fast memory
  - Identify bandwidth limitations between Opteron and FPGA
- Determine external and internal FPGA data sizes
- Decide features to support in each core

Parallelization

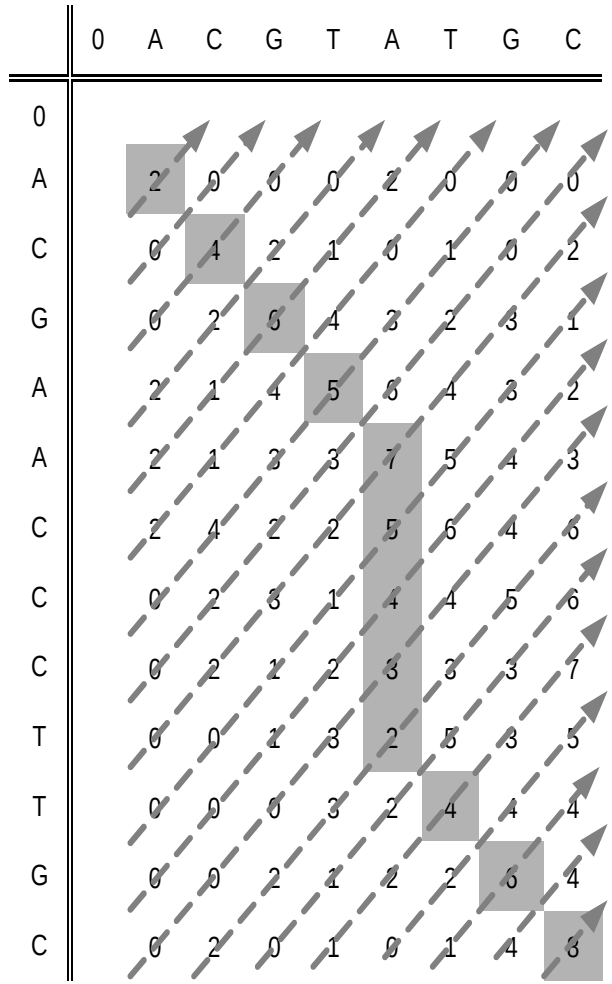
Filling scoring matrix is parallel along antidiagonal, while traceback is serial…

SO…

Put forward pass on FPGA, traceback and alignment on Opteron.

BUT…

Scoring matrix would take longer to send back than it did to calculate!



Two solutions:

1. Send back traceback information only (2 bits per cell)
2. Use FPGA for scoring only and regenerate matrix on Opteron for top scores

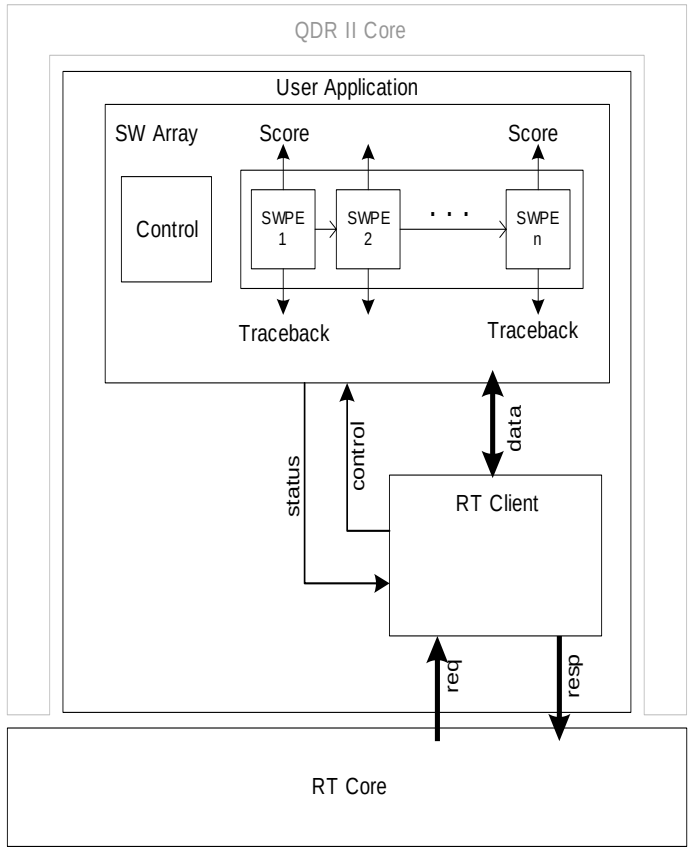
Currently, we use (2.)

Data sizes and features

- Input data can be 4-bit (Nucleotides) or 8-bit (Amino acids)
  - We are generating two separate FPGA cores
- Keep other data as small as possible
  - Scoring matrix, max score – 32 bit
  - Traceback – 2 bits
  - Gap penalties and substitution matrix – 8 bits
- Support only single affine gap model

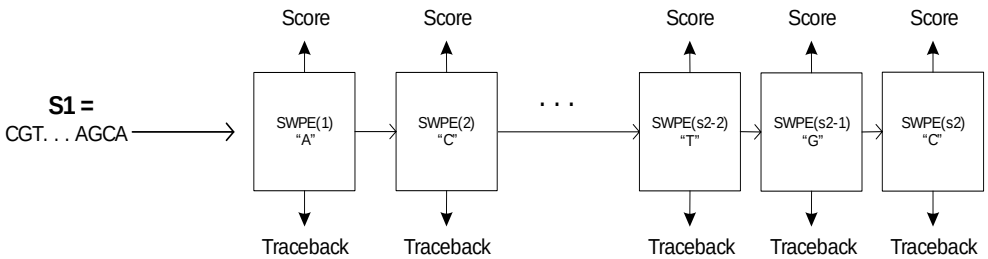
Logic Design

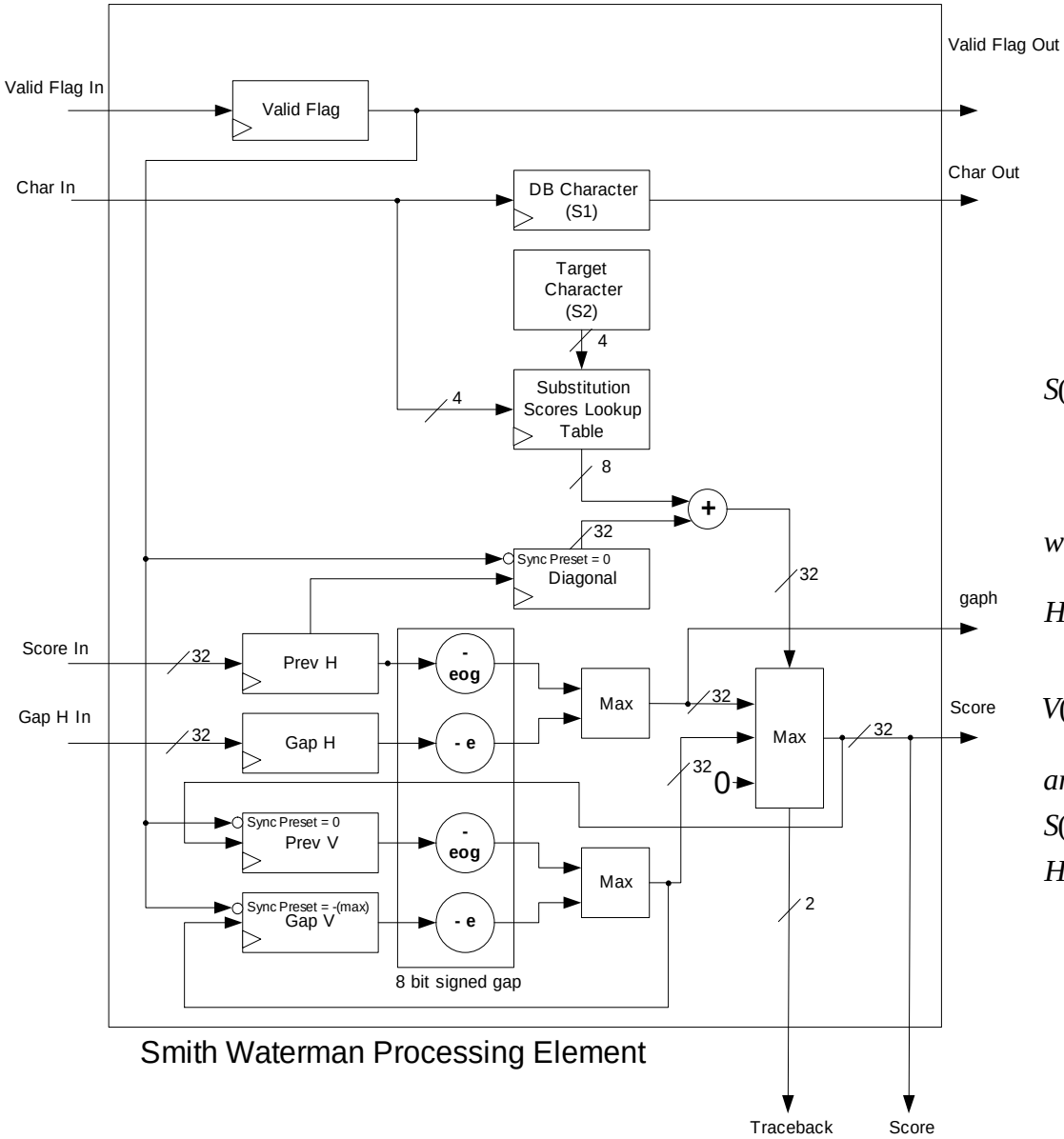
- Systolic array of SWPEs (Smith Waterman Processing Elements)
- Each calculates one cell, for one letter of the query string
- The database string is streamed into the array, as shown below:



Overall Architecture

Systolic array of SWPEs





Smith Waterman Processing Element

- Single SWPE Logic Design

$$S(y,x)=\max\left\{\begin{array}{c}0,\\ S(y-1,x-1)+Sub(cy,cx),\\ S(y,x-1)-eog\\ S(y-1,x)-eog\\ H(x-1)-e,\\ V(y-1)-e\end{array}\right\}\ for 1\le x\le s1, 1\le y\le s2$$

where

$$H(y,x)=\max\left\{\begin{array}{c}H(y,x-1)-e\\ S(y,x-1)-eog\end{array}\right\}\ for a given y$$

$$V(y,x)=\max\left\{\begin{array}{c}V(y-1,x)-e\\ S(y-1,x)-eog\end{array}\right\}\ for a given x$$

and:

$$S(0,x)=S(y,0)=0$$

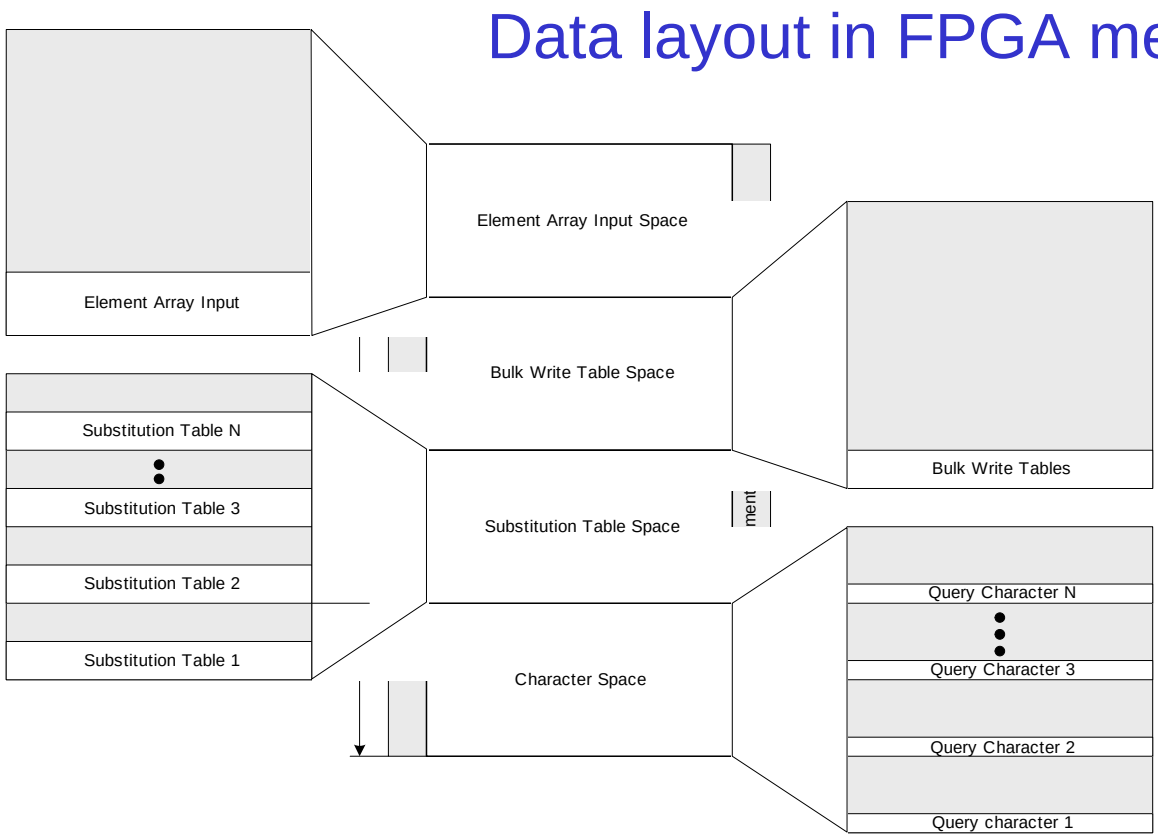
$$H(0)=V(0)=-eog$$

FPGA to Program Interface

- For individual or small transfers, use FPGA registers
  - *fpga_wrt_appif_value()* and *fpga_read_appif value()*
  - Gap penalties, data sizes, flags and max score
- For large data transfers, write directly into FPGA memory map with *fpga_memmap()*
  - Substitution table, database and query strings

SWA Memory Map

- Use *fpga_memmap()* to get pointer, then *memcpy()* or similar to transfer data



Performance Predictions

- Rate = FPGA freq. X clocks/cell X # SWPEs
  - Current unoptimized (working!) design:
    - 80 MHz X 1 X 32 = 2.6 Billion Cell Updates Per Second (GCUPS) – 60% of chip used
  - With optimization:
    - 100 MHZ x 1 x 50 = 5.0 GCUPS
  - With future Virtex 4 FPGA
    - 100 MHZ x 1 x 150 = 15 GCUPS
- SSEARCH34 on Opteron delivers about 100M CUPS
  - 25x speedup now, more in future

Future work

- Nucleotide core is working, but needs to be optimized for longer query strings
- Generate amino acid core (very similar)
- Interface SWA cores to libraries and applications
  - Cray Bioinformatics Library
  - SSEARCH
  - EMBOSS

Conclusions

- The FPGAs on the XD1 can offer a real speedup for bioinformatics algorithms
- Careful partitioning of your algorithm is necessary for best use of the FPGA feature
- Pay attention to parallelism, complexity and memory usage
- Bandwidth, bandwidth bandwidth!

(But if you're here at CUG you already knew that…)

Thank you!

jmaltby@cray.com
(206) 701-2107