Development of the SMART Coprocessor for Fuzzy Matching in Bioinformatics Applications

Harrison B. Smith and
Eric A. Stahlberg
The Ohio Supercomputer Center
Columbus, Ohio

OSC SPRINGFIELD

ASC

OSC Mission

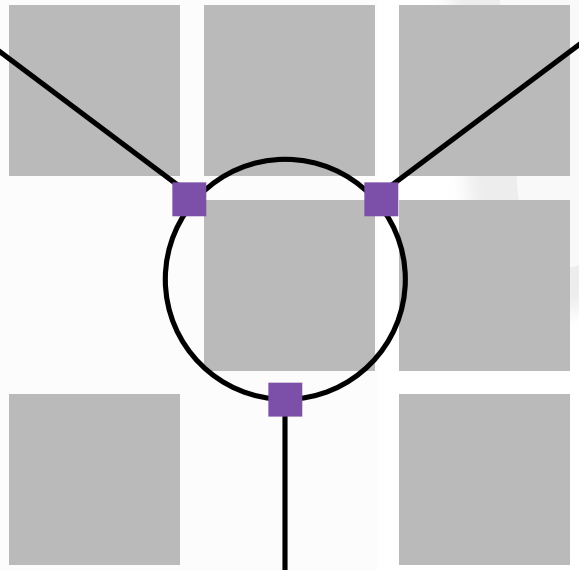
- OSC provides a reliable **high-performance computing and high-performance communications infrastructure** for a diverse, statewide/regional community including education, academic research, industry, and state government

- OSC **promotes and stimulates computational research** in order to act as a **key enabler for the state's aspirations in advanced technology, information systems, and advanced industries**, and,

- OSC acts as a **catalytic partner of Ohio universities and industries** to enable Ohio to compete for international, federal, and state funding, focusing on new research and business opportunities in:

Networking

- Internet
- Internet2
- OARtech
- OSTEER
- Support
- TFN
- Training
- Videoconferencing
- 91 Higher Education Institutions



Computing

- Access Grid
- BALE
- Cluster Ohio
- HPC Services
- ODCE
- Platform Lab
- SUG
- Summer Institute
- Support
- Training
- Young Women's Summer Institute

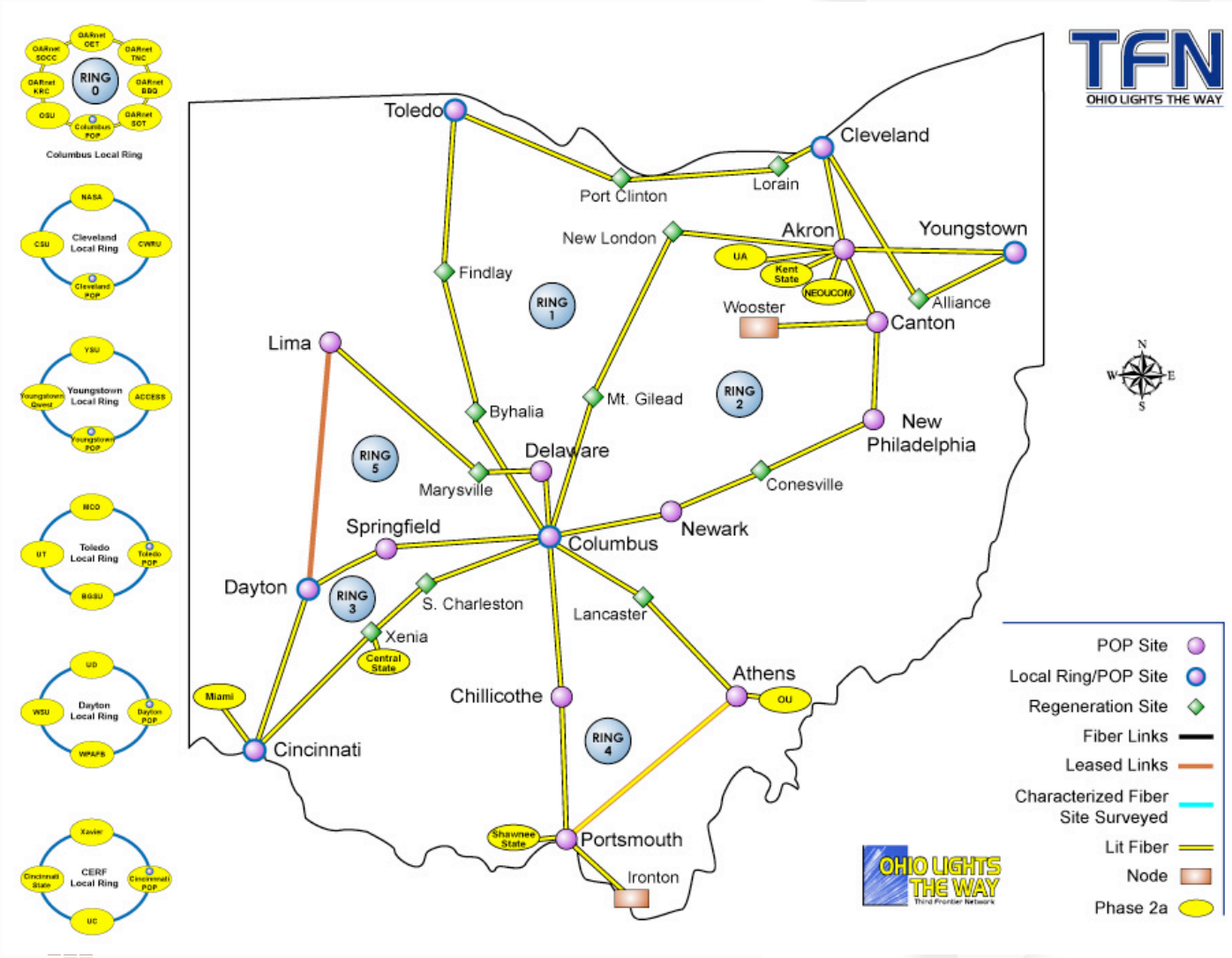
Research

- Bioinformatics
- Biomedical Applications
- Blue Collar Computing
- Data Intensive Computing
- Grid Computing
- HPC for Defense and Homeland Security
- Nanotechnology
- Networking and File Systems

OSC SPRINGFIELD

ASC™

OSC OARnet: TFN Overview



OSC HPC: Systems

- **Columbus**
  - **512p Intel Pentium 4 Xeon Cluster (2.5T)**
  - **540p HP Intel Itanium2 Cluster (2.4T)**
  - **32p SGI Altix (Itanium2) (.17T)**
  - **3 16-way SGI Altix 350s (Itanium2 1.4GHz) (.27T)**
  - **40p Sun SunFire 6800 (.07T)**
  - **100p AMD Athlon Visualization Cluster (.3T)**
  - **IBM FastT900 and FastT600 (.5 PetaBytes of disk)**

- **Springfield**
  - **Cray X1 (.2T)**
  - **Cray XD1 (.32T+FPGA)**
  - **Apple Xserve G5 (.5T)**

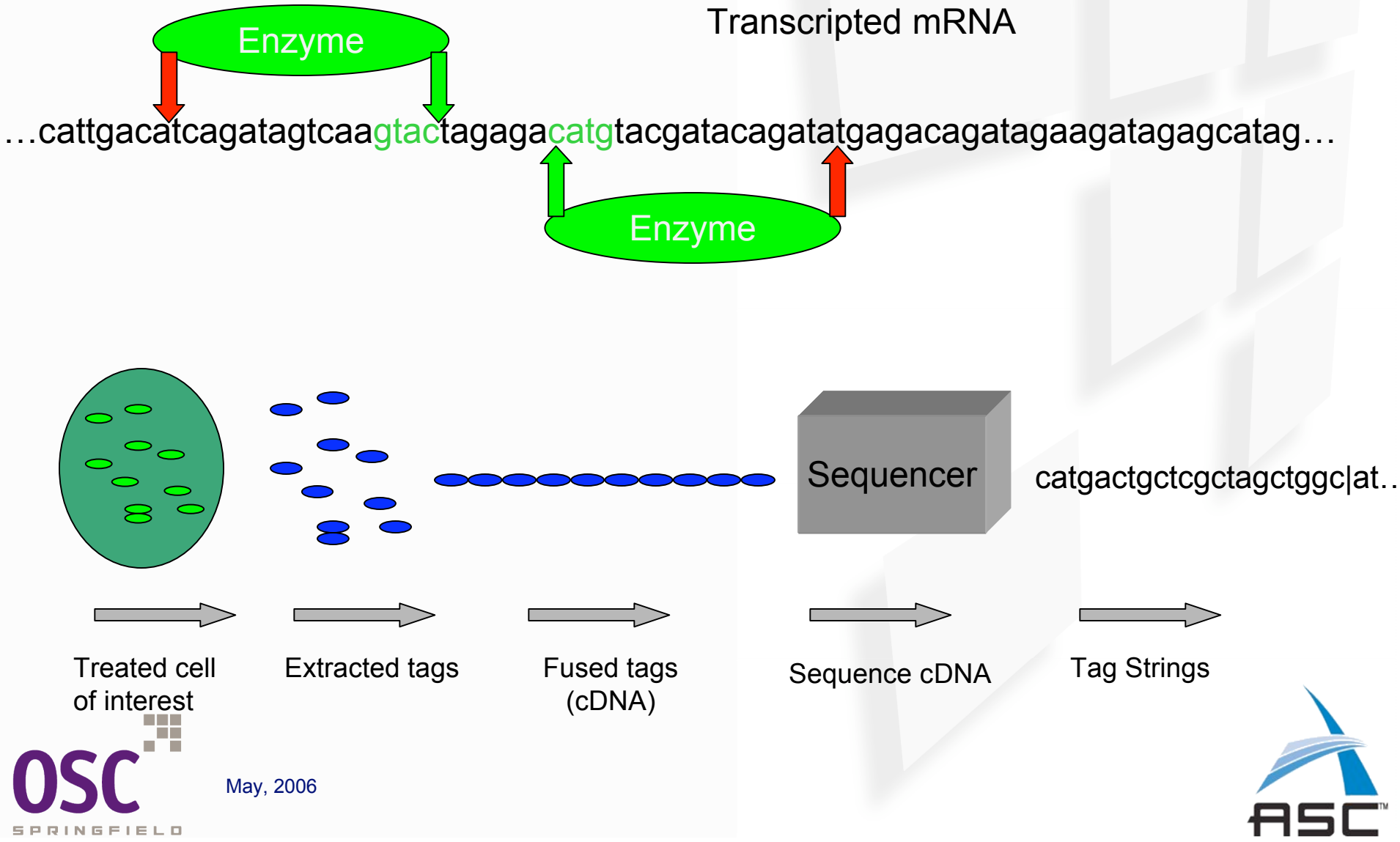
OSC and FPGAs

- Began using FPGAs in 2001
  - TimeLogic production system for bioinformatics
  - Users were pleased with performance
  - Wanted to program, but environment was closed
- Acquired XD1 in 2004
  - Beta site for XD1
  - Part of Data Intensive Computing Initiative
  - FPGAs became available to program for applications
  - Fortunate to have VHDL experience available
  - Tools were in infancy, but potential was getting sorted out
- Demonstrated potential
  - Papers at CUG 2005, FCCM 2006, etc.

What is SAGE?

- SAGE = Serial Analysis of Gene Expression
- Originally developed in 1995 to study cancer
- Select for short unique segments that can be readily extracted, characterized and located
- Primary use
  - Transcriptome analysis – creating a physical map of an organism's genome
  - Quantitative measurement of an organisms response to events
  - Alternative to gene expression micro-array analysis

SAGE Schematic

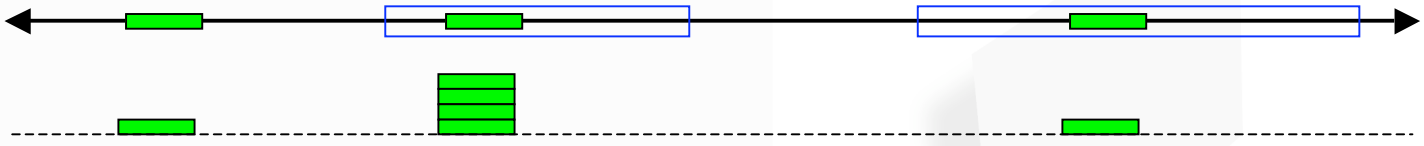


Map Back to Known Genome Regions

SAGE tags from experiment
O(100k)

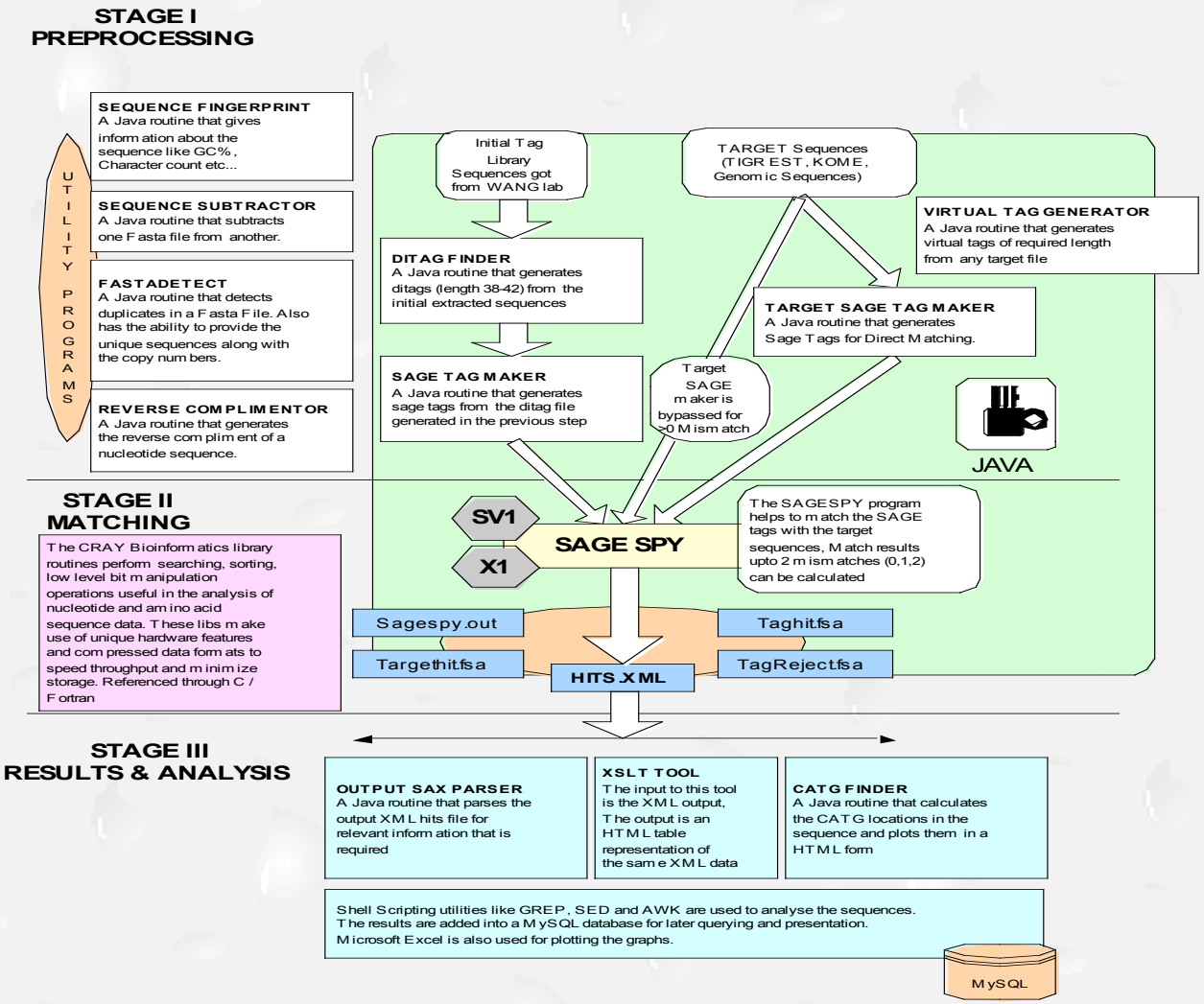
catgactgctcgctagctggc
catgattcgctggcatgcagc
catgatttacgtagaccgata
…

Full Genome Sequence Sets
O(1M)



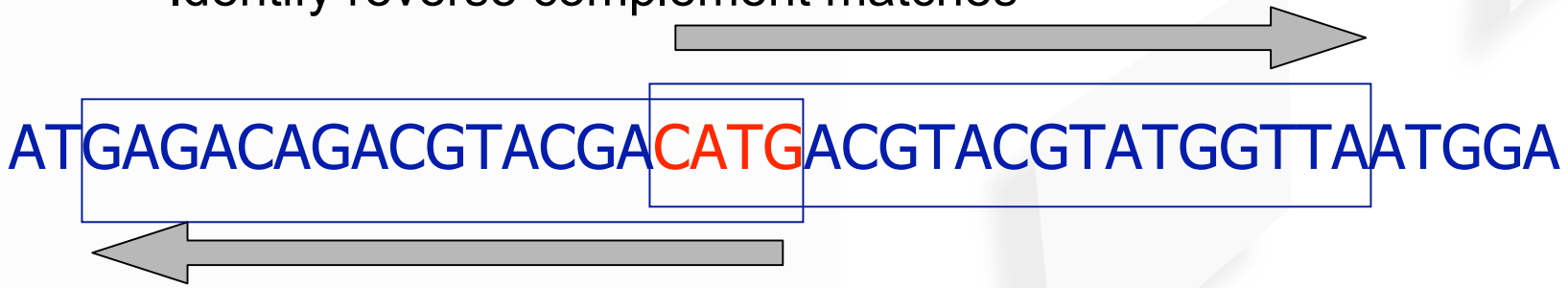
Genome location and frequency amplitude

SAGE ANALYSIS (OSC and Dept. of PLANT PATHOLOGY)



The Challenge

- To create a highly efficient means to compare very large quantities of short DNA sequences
  - All short tags (limited by chemistry of enzyme)
  - Theoretical limit for RL-SAGE: 4^{17} unique tags
  - Reality: millions matched against hundreds of thousands
- Scanning entire genomes for SAGE tag patterns
  - Identify matching single and double mismatches (compensate for experimental error)
  - Identify reverse complement matches



SAGE Background

- Originally designed for lengths of 9-10 nucleotides
- Various extensions to improve discriminating power when mapped back to original DNA locations
- Extended to 21 nucleotides in RL-SAGE
- Current capabilities demand
  - Files containing 3 million or more tags
  - Searching against entire genomes
  - Hit rates low in terms of absolutes

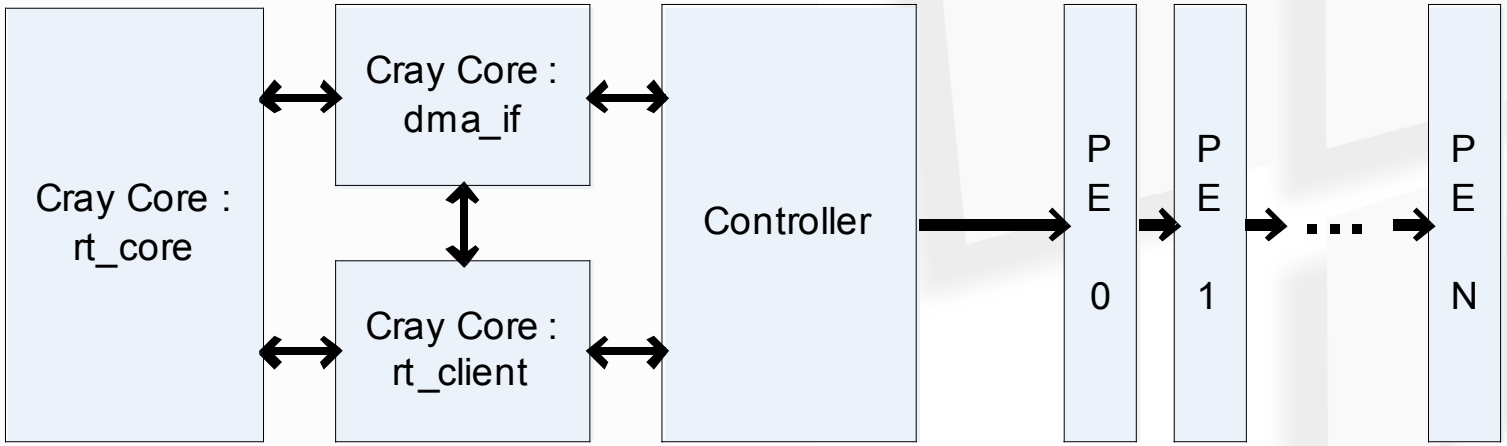
Motivations for Current Work

- Speed analysis of SAGE data
- Move analysis closer to experimental benchtop
- Lower costs to compare while maintaining turnaround time
- To determine if FPGA implementation can be faster than fastest other implementations

SMART Solution

- SMART
  - Sage Method Analysis with Reconfigurable Technology
- Guiding philosophy for the solution
  - Fuzzy matching likely to be more efficient on FPGA
  - Keep it simple (maximize parallel operations)
  - Data flow choice
    - Flow target more rapidly than query (each PE has a unique pattern to capture)
    - Rather than flowing query more rapidly than target
  - Design for portability

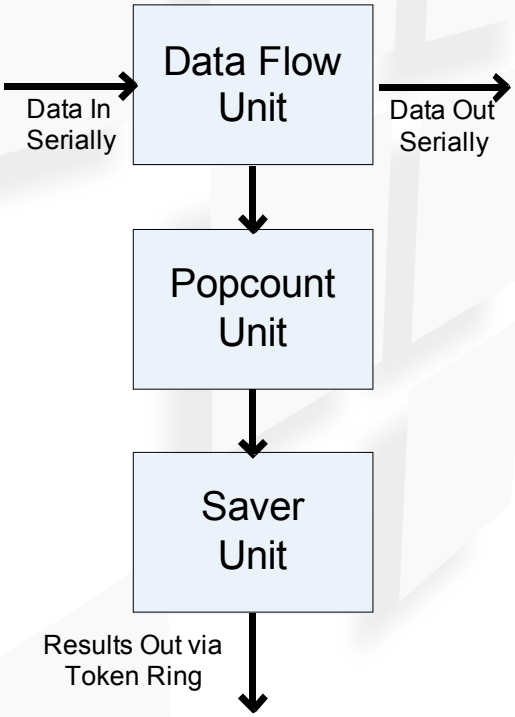
SMART Solution



- Major architectural features
  - Processing Element (PE) – easily replicated to achieve parallelism on larger FPGAs
  - Controller to isolate machine dependence and manage data flow

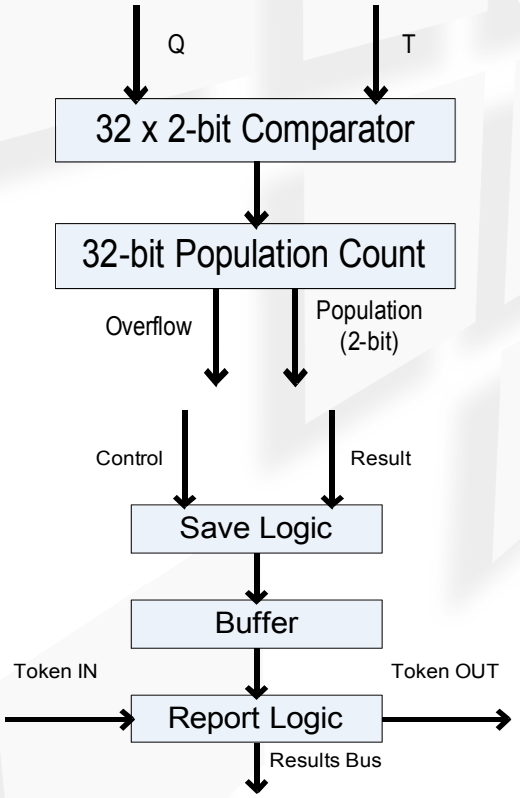
SMART Processing Element

- Overall Design – Linear Array Element
  - All data/control signals passed to next PE
  - Allows for design expandability
  - Three main subcomponents
- Component 1 - Dataflow Unit
  - Handles the passing and buffering of all data needed by the processor
  - Simple but large unit to achieve high buffering efficiency



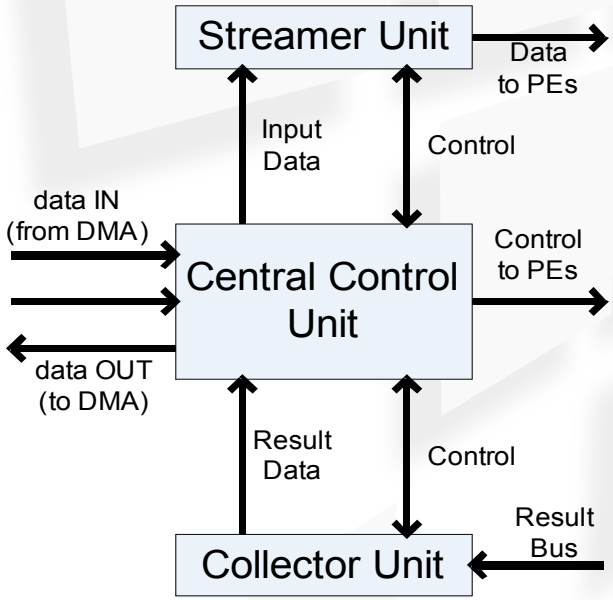
SMART Processing Element

- Component 2 - Pop Count Unit
  - Six level pipelined unit
  - Actual “comparisons” done here
  - Optimization possible by collapsing pipeline stages
- Component 3 - Saver Unit
  - Analyzes popcount output and buffers hits
  - Handles reporting results via token ring bus



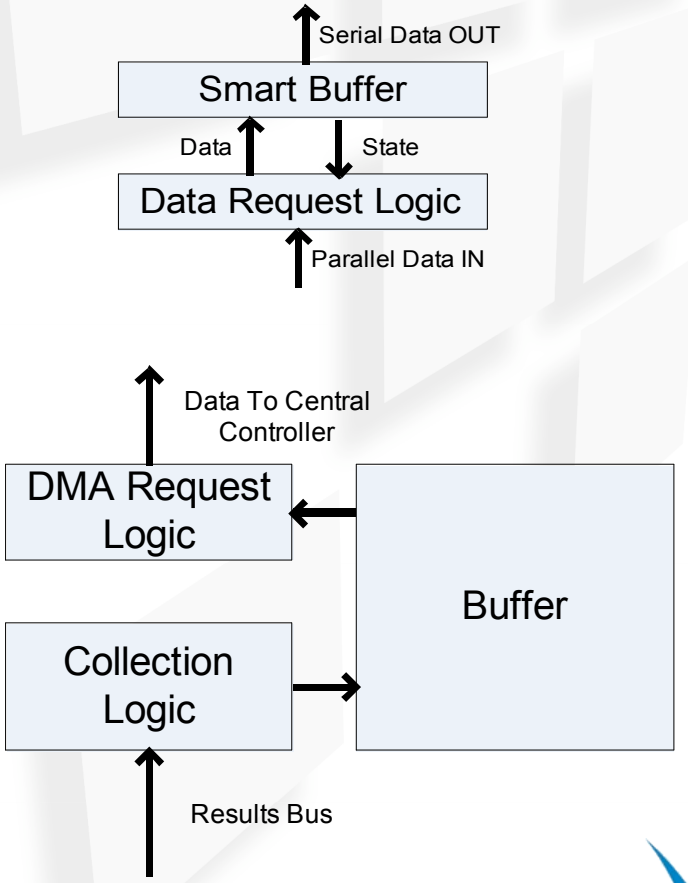
SMART Controller

- Single Controller Terminates Chain of Processors
  - Handles all communication minimizing impact of changes required for new environments
  - Contains all state associated with the system (control FSM)
  - Whenever possible, functionality moved from PEs to Controller
  - Also three major subcomponents



SMART Controller

- The Data Streamer
  - Requests DMA reads from the Central Control Unit
  - Passes characters to the PE chain serially
- The Data Collector
  - Monitors the reporting token ring for results
  - Requests DMA writes from the Central Control Unit



SMART Controller

- Central Control Unit
  - Contains FSM that runs the whole show
  - Provides monitored access to the Cray provided DMA read/write core
  - Passes state information to the PEs via control lines
  - Contains all configuration information

SMART Solution Design Elements

- Design uses preexisting Cray provided IP cores for data transfer
- DMA used for bulk of sequential data transfer
- Solution utilizes duplex nature of interconnect
- Configuration and input data contained in same memory space

Design Elements Continued

- Token ring / results bus combination provides efficient data reporting
- Linear array allows for scaling, efficient usage of resources
- PEs can be expanded to accommodate additional functionality easily

Development Overview

- Development Stages
  1. System Design
  2. System Implementation
  3. System Simulation
  4. System Synthesis
- Software Interface
  - Lightweight – reads file, compresses data, sets up DMA regions
  - In development

SMART Design and Implementation

- Initial Design Strategy
  - Space scalability was important
  - Influenced by other OSC/XD1 work
  - Objective: exceed performance of Opteron and X1 version of SAGESpy
- The Implementation/Redesign Cycle
  - Standard practice

SMART Solution Status

- Unit Testing [done]
  - Testing functionality of individual components
- Data Path Testing [done]
  - Testing functionality of strings of components
- Full Controller/Full PE Testing [done]
  - Testing of our complete unit
- Full Unit Testing [simulation in process]
  - Testing of the full hardware

SMART Synthesis

Synthesis results

- Single Virtex2Pro FPGA will hold up to 60 PEs running at 180MHz at least
- Virtex4 could possibly hold 300 PEs running at an estimated 350MHz

Unit	Slices	MHz
Controller – Full Unit	299	240
dma_if (Cray) – Full Unit	727	190
rt_client (Cray) – Data Collector	468	270
Processing Element - Dataflow	119	344
Processing Element - Popcount	65	307
Processing Element - Saver Unit	102	280

SMART Software

- Software development and specification
  - All efforts were made to keep software simple
  - Target use within SAGESpy application and OBL
  - Cray BioLib used in FASTA file read and compression
  - Program arranges data in DMA memory space, then signals FPGA and waits
  - Program outputs results in simple ASCII.
    - XML, other formats planned

SMART Results

- SMART Raw number projections:
  - Will be able to perform 10.8B comparisons per second (Virtex2Pro)
    - 60 processors
    - 180 MHz clock rate
    - One compare per clock
  - Not as compelling as hoped for

Performance Comparison

	Number of target bps	Computation time in seconds				Number of Hits
		SV1	X1	XD1 w V2P (estimated)	XD1 w V4 (theoretical)	
Chrom1	13333992	1.35	0.36	0.066	0.0165	29
Chrom4	9923256	1.00	0.27	0.049	0.0124	34
Chrom10	5964420	0.61	0.16	0.030	0.0074	8
TIGR(EST)	9889404	1.00	0.26	0.049	0.0124	333

Table 1. Performance of SAGESpy XOR comparison approach relative to estimates for SMART co-processor. Speed was measured for one hundred query tags with a two mismatch threshold.

- Assumptions:
  - 200 MHz speed of FPGA (Virtex2Pro)
  - 100 processing elements per FPGA (Virtex2Pro)
  - 20 billion comparisons/second/FPGA

SMART Future

- SMART not yet finished – expecting to:
  - Enhance functionality
    - Simultaneous reverse string comparison
    - Variable sizes for queries/tags
    - Decode output into standard C types
  - Improve performance
    - Use increasing memory on FPGA more effectively
    - Internal buffer overflow protection
    - Consolidate compression with reading of original data
    - Multiple FPGA implementation

What We Learned Along the Way

- Writing hardware is a more dynamic experience than writing software
- Don’t grow attached to your initial ideas. If something isn’t working, find another way!
- Reduce communication protocols between functional units as much as possible
- Naming conventions are very, very important
- Keep detailed development logs
- Unit testing will save you time and frustration

Conclusion

- Reconfigurable computing application development still hard with VHDL
- Virtex2Pro is good, but isn't quite compelling
- Virtex4 becomes compelling
- I/O is still a major factor
- Anxious to compare with C-based solutions to FPGA algorithm specification

Acknowledgements

- US DOE for funding support
- OSC FPGA team for initial guidance
- Wang Lab (OSU) for motivation and sources of data

Thank you.

Contact information
Ben - bsmith@osc.edu
Eric - eas@osc.edu