

Streams-C

Maya Gokhale

Los Alamos National Laboratory

September, 1999

LANL/SLAAC Activities

- Challenge Problem Formulation
- Integration of SLAAC Technology
 - Wideband RF
 - Hyperspectral Imaging
- Design Tool Validation
- Streams-C

Background

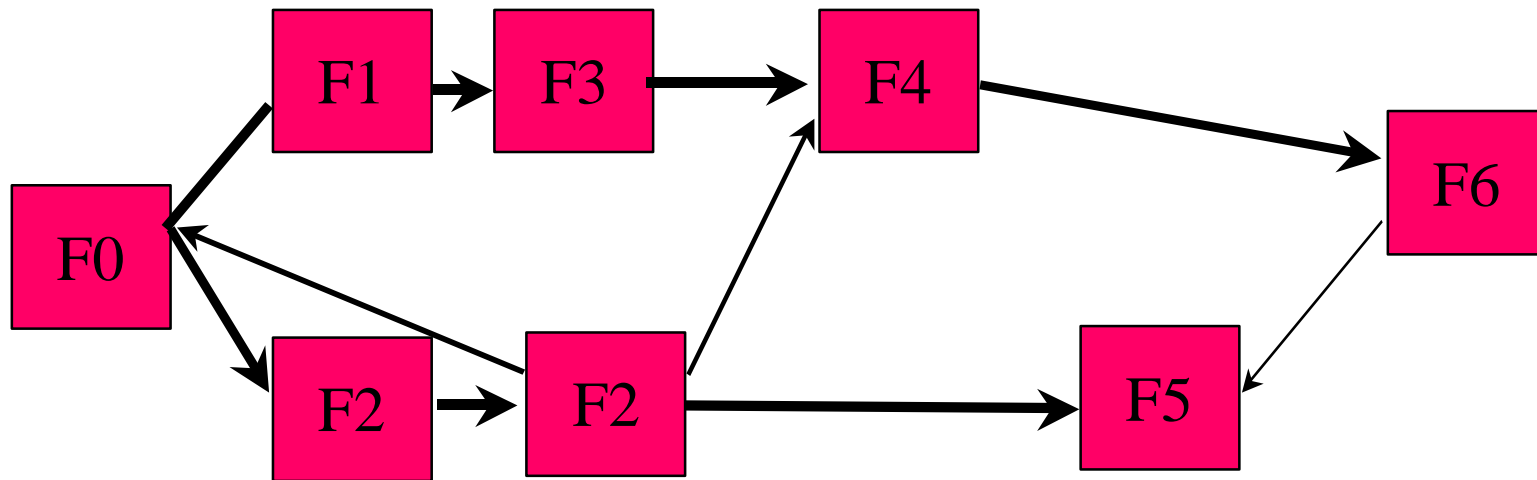
- 1997 ACS start, 3 year program
- Goal: To develop
 - programming model
 - language constructs
 - compiler technology
 - runtime library modules
 - reference programming examplesfor stream-oriented ACS applications.

Background (con't.)

- Year 1 - applications study and simulator development
- Year 2 - compiler development
- Present status: From C input we can generate Wildforce Streams-C applications
 - pipelined processing
 - stream communication
 - on-board sequencer
 - pipelined memory access

Programming Model

- Collection of processes, separate address spaces
- High bandwidth synchronous data streams communicated among modules
- Low bandwidth asynchronous signals for phase coordination and flow control
- Flow may be a graph rather than simple pipeline
- Covers virtually all ACS applications - high data rate front end processing



Streams-C vs. MPI

Similarities

- Each process has a separate address space
- Processes communicate through messages

Differences- unlike MPI

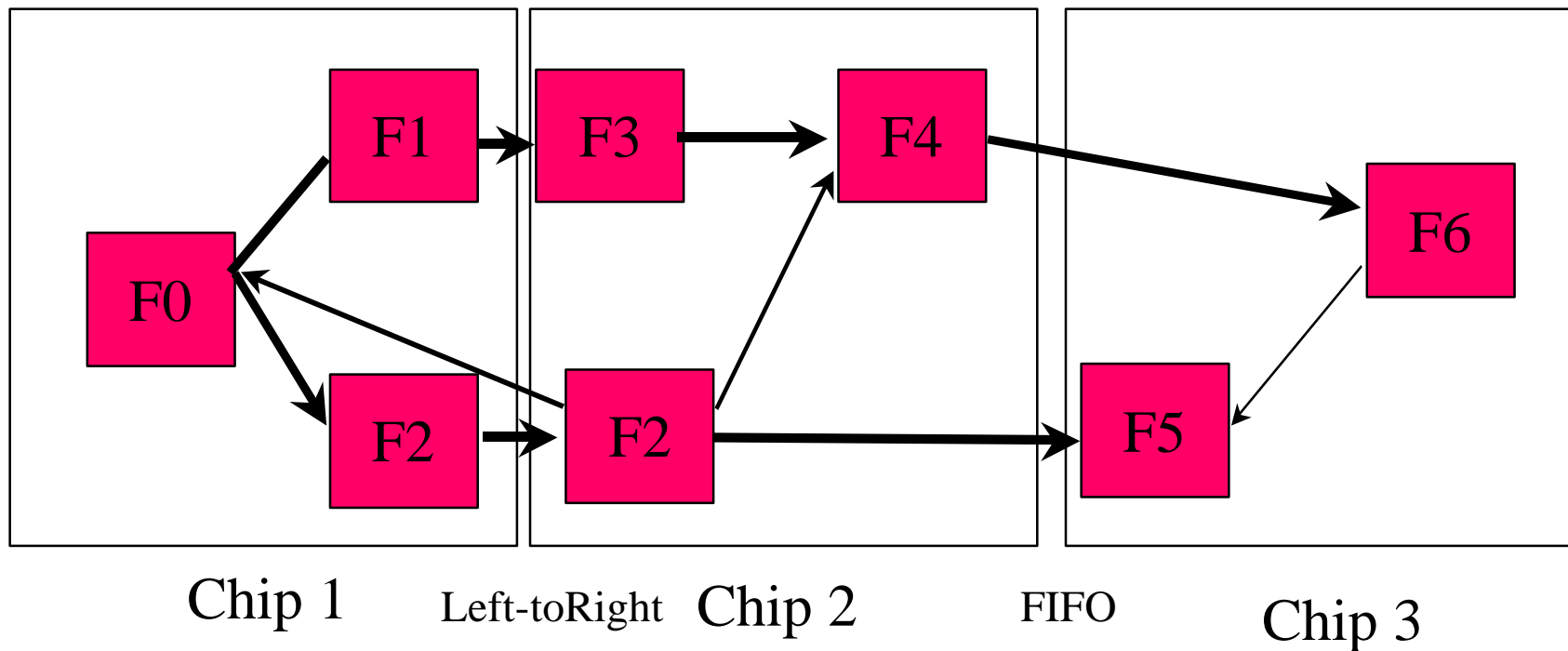
- Streams-C supports highly synchronous communication
- Streams-C data size is typically small
- Streams-C does not define aggregate communication patterns

Programming Model

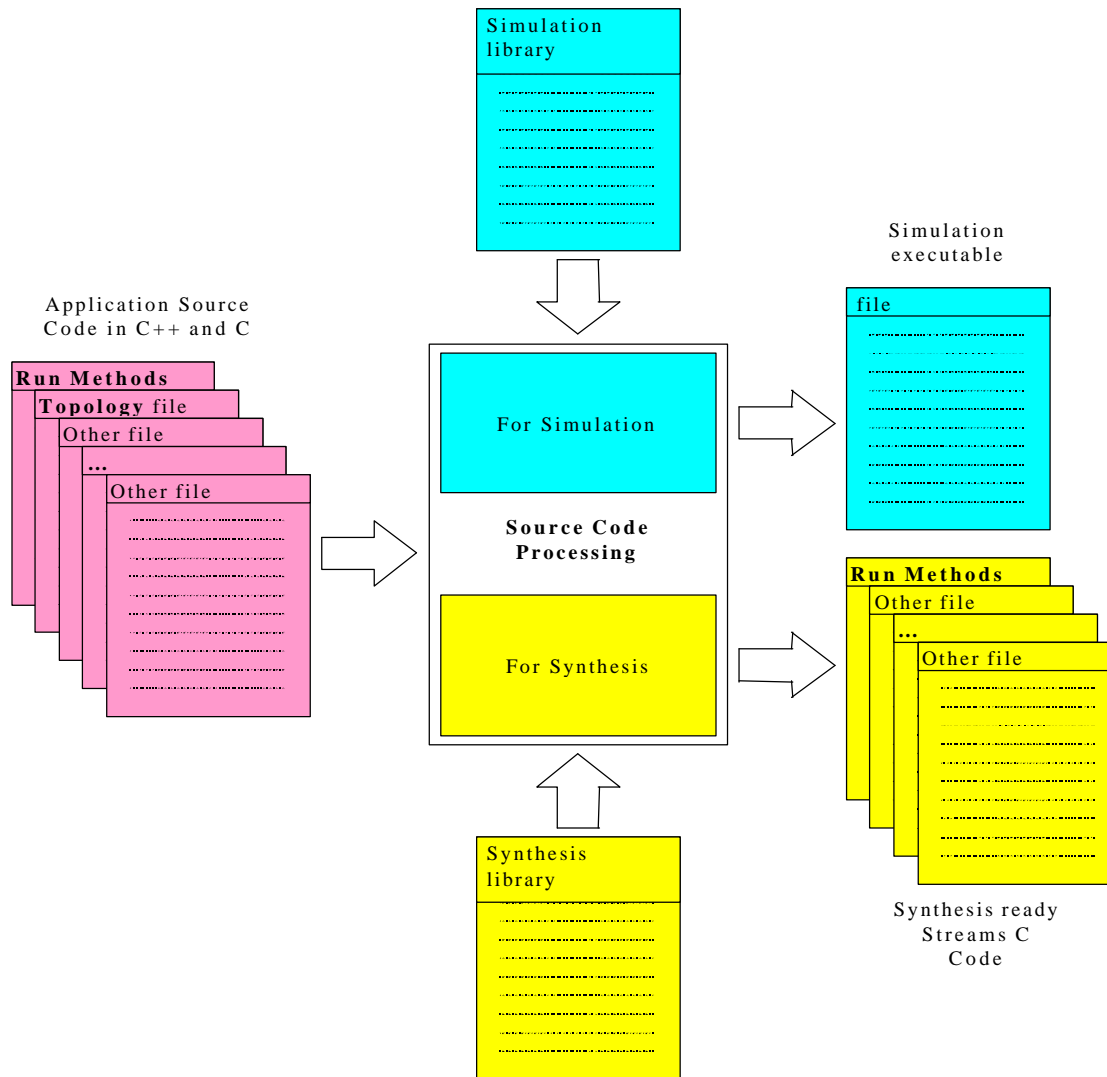
- Process
 - parameters are input and output streams, signals that the process will use
 - body is a C subroutine
- Stream
 - defined by
 - size of payload,
 - flavor of stream (valid tag, buffered, ...), and
 - processes being interconnected
 - operations are open, close, read, write, eos, [eoframe]
- Signal
 - optional payload parameter
 - operations are post, wait
- Topology
 - describes interconnection of processes, streams, signals
 - describes mapping of processes, streams, signals to hardware board

Physical Mapping Information

- Process
 - which chip
- Stream
 - which physical I/O pins on chip
- Signal
 - mapping to physical resources determined by our library



A Single Environment for Functional Simulation and Synthesis



```

/// PROCESS_FUN pel_proc_run
/// INPUT input_stream
/// OUTPUT output_stream
/// PROCESS_FUN_BODY

SC_FLAG(tag);
  SC_REG(data, 32);

  int i; int odata;

  IF_SIM(sprintf("Process pel_proc entered\n"));

  SC_STREAM_OPEN(input_stream);
  SC_STREAM_OPEN(output_stream);

  while(SC_STREAM_EOS(input_stream) != SC_EOS) {
    SC_STREAM_READ(input_stream, data, tag);
    odata = SC_REG_GET_BITS_INT(data, 0, 32);
    odata |= 0xff00;
    SC_REG_SET_BITS_INT(data, 0, 32, odata);
    SC_STREAM_WRITE(output_stream, data, tag);
  }

  SC_STREAM_CLOSE(input_stream);
  SC_STREAM_CLOSE(output_stream);

  IF_SIM(sprintf("Process pel_proc exiting\n"));
  printf("Process pel_proc exiting\n");

/// PROCESS_FUN_END

```

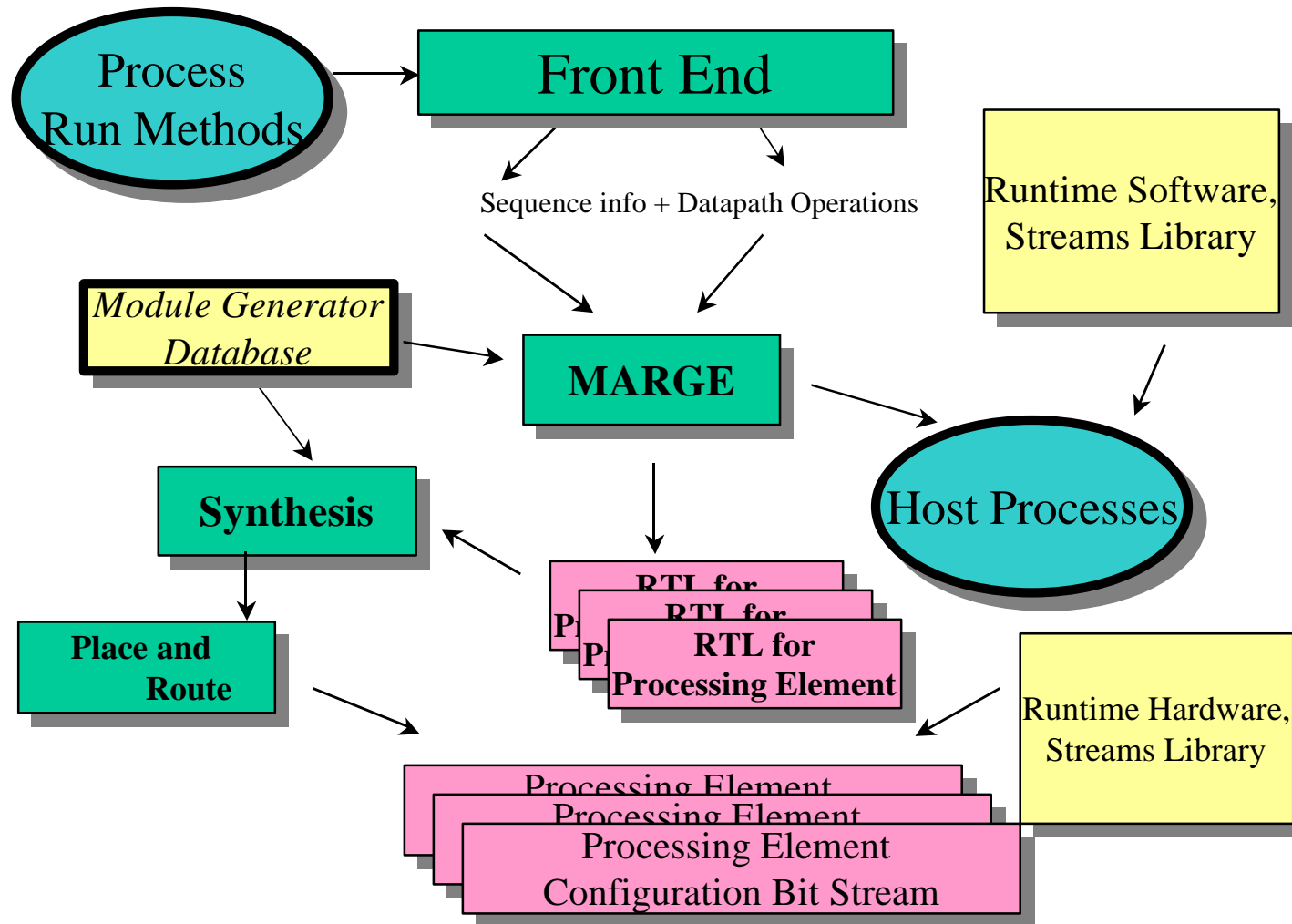
Process Declaration Stream Declaration

Stream Operations

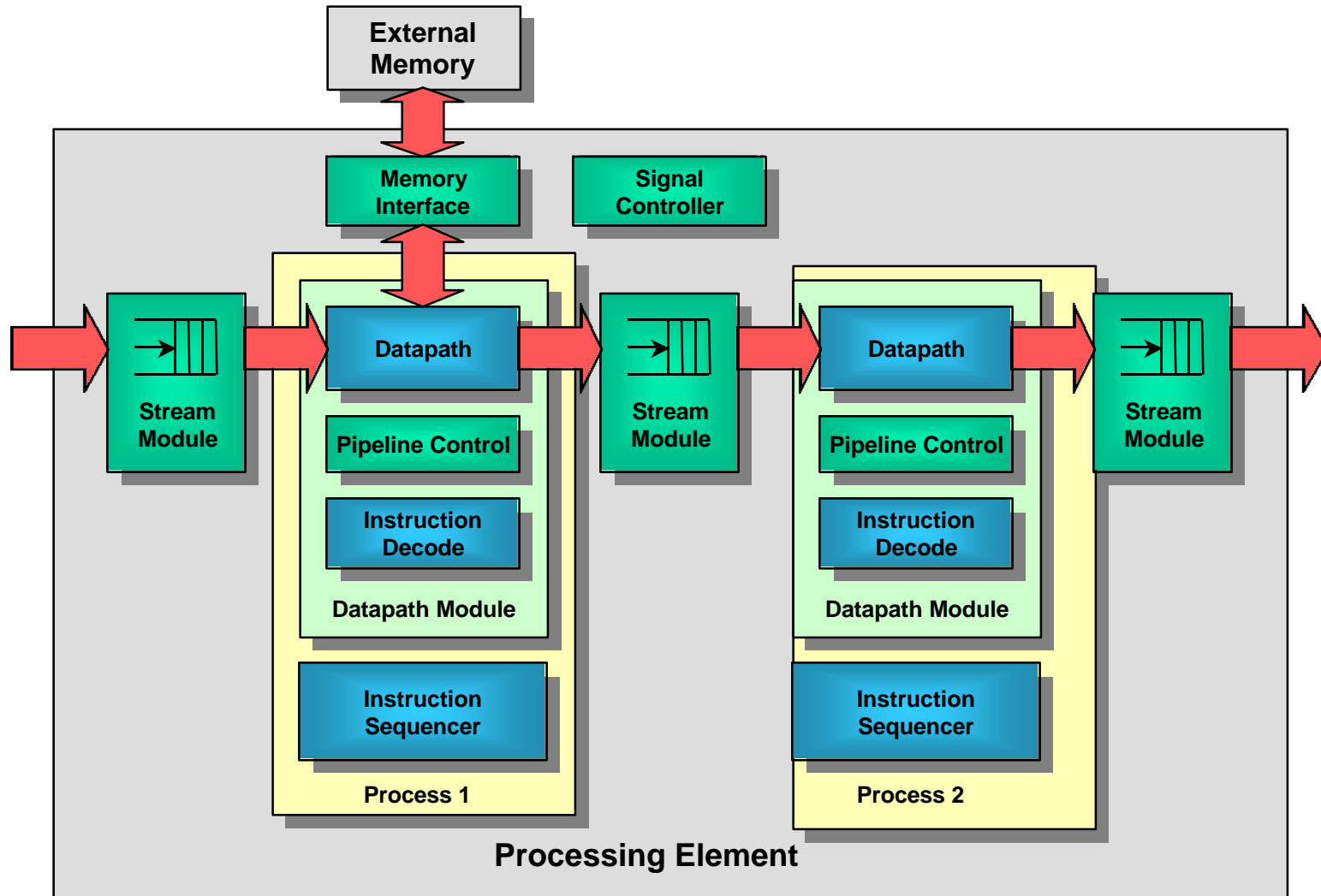
Streams-C Compiler

- Builds on Malleable Architecture Generator (MARGE) synthesis technology
- Builds on NAPA C SUIF-based pragma infrastructure and analysis
- Unique extensions to streams processing
 - process/stream directives
 - mapping directives

Streams C Compiler Structure

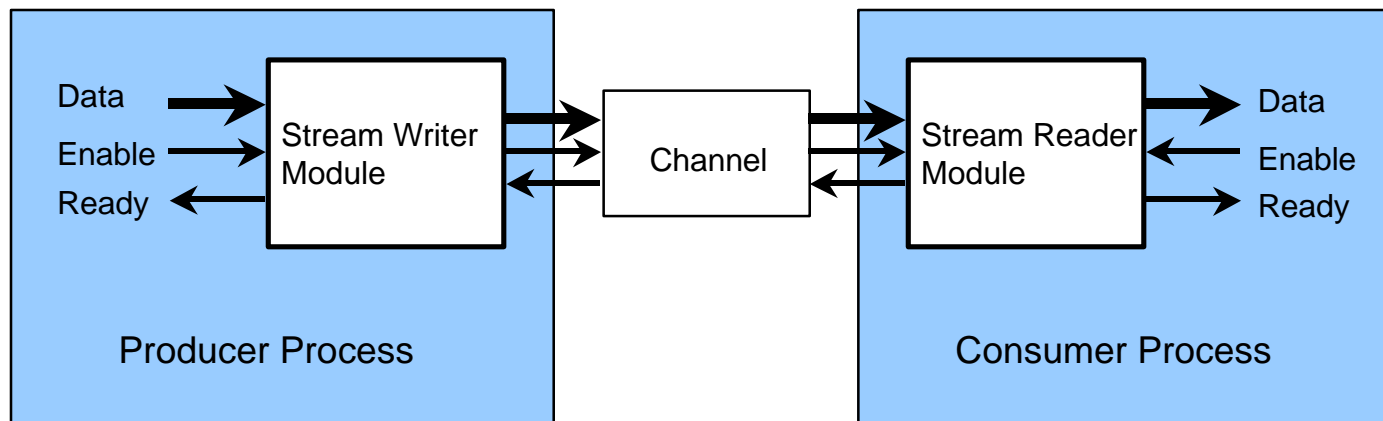


Processing Element Structure



Stream Hardware Components

- High bandwidth, synchronous communication
- Multiple protocols: “Valid” tag, buffered handshake
- Parameterized synthesizable modules
- Multiple Wildforce channel mappings:
 - Intra-FPGA, Nearest neighbor, Crossbar, Host FIFO



Stream Modules

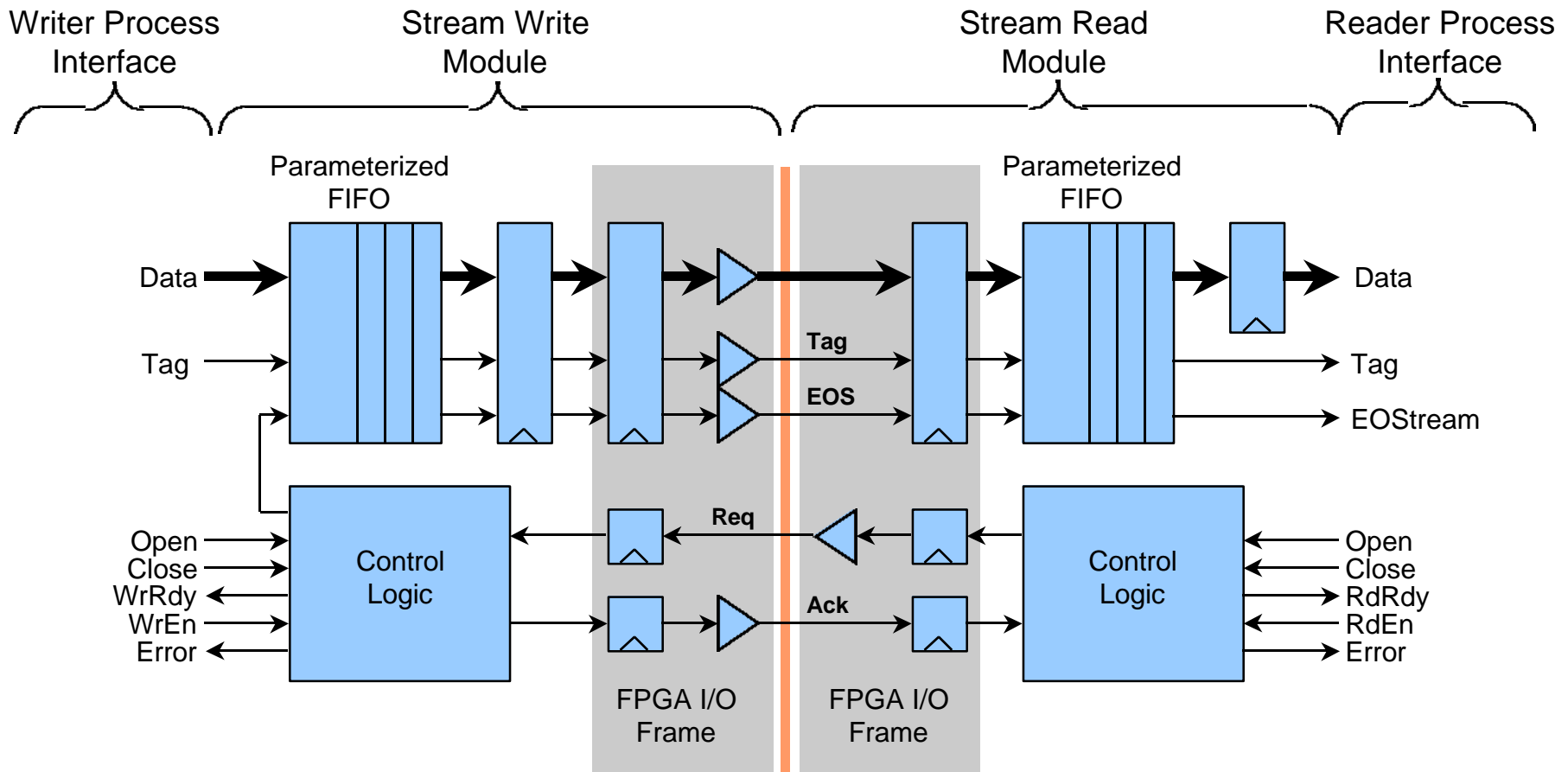
Stream Module Selection

- Stream Type
 - Valid Tag
 - Buffered
 - Full
- Parameters
 - Element Width
 - Buffer Depth
- Context
 - Intra-process
 - Nearest Neighbor
 - Crossbar
 - Host FIFO

Stream Operations

- Open
- Close
- Read
- Write
- End of Stream
- Error

Stream Module Example

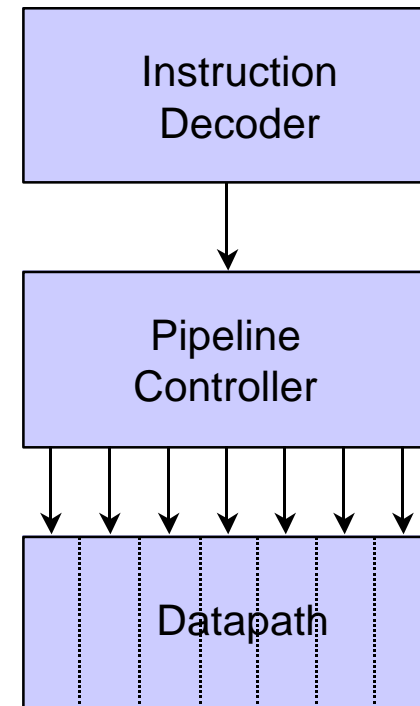


Hardware Control Structures

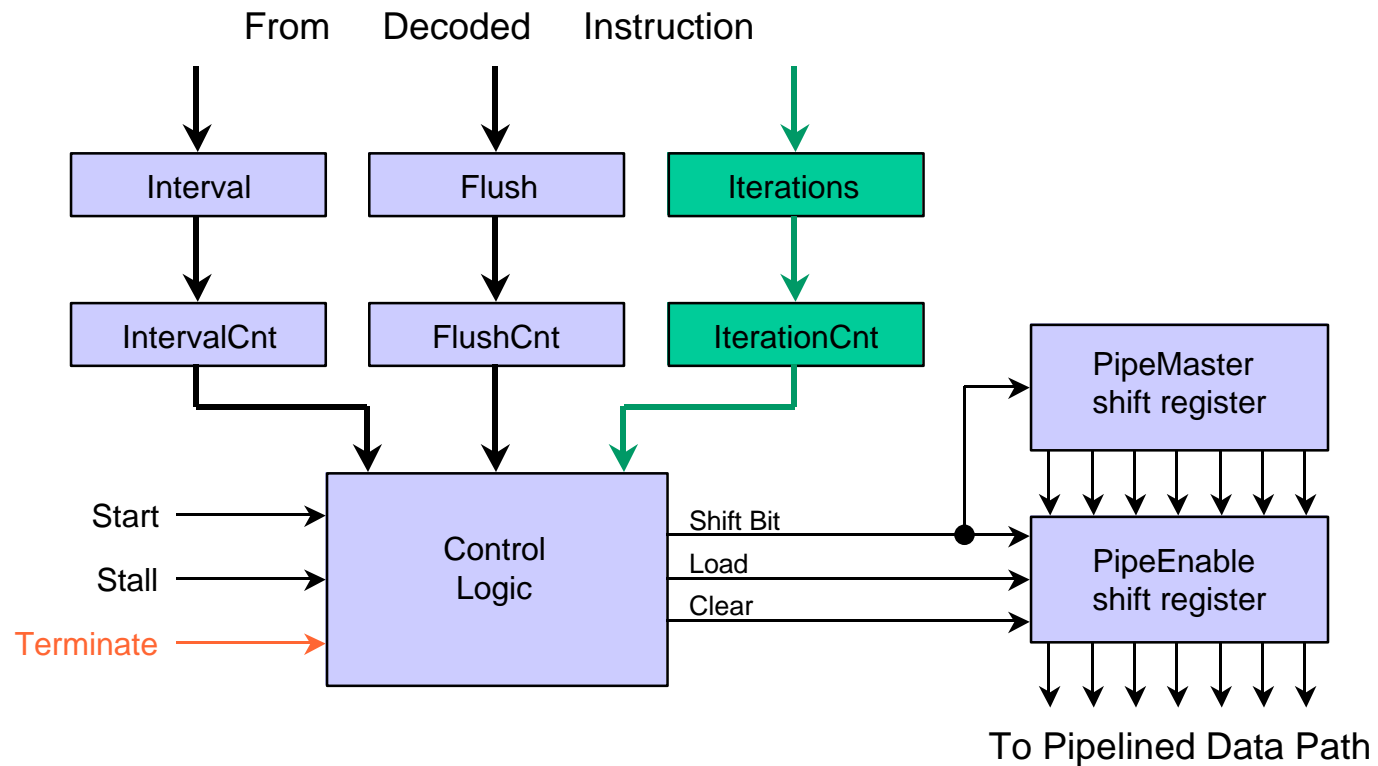
- Instruction Sequencing
- Instruction Decoding
- Loop pipeline control
 - Definite iteration
 - Indefinite iteration
- Memory access

Pipeline Control Modules

- Controls data path pipeline register enables
- Started by decode of “pipeline instruction”
- Automatic stall
 - Handles full/empty stream conditions
- Programmable:
 - Number of pipeline stages
 - Initiation Interval
 - Number of flush cycles
 - Number of iterations (definite iteration pipelines)



Pipeline Control Structure



Software Runtime Library

- Compiler synthesizes control program
- Automatically inserts calls to runtime library
- Runtime system controls:
 - Board initialization
 - memory initialization
 - PE and Crossbar configuration

Current Status

- Compiler-generated benchmarks and kernels run on Wildforce board
- We are transitioning hardware and software environment to LANL
- Jan Stone will continue on compiler effort
- Jeff Arnold no longer available, replaced by Dominique Lavenier, sabbatical visitor from IRISA, France (DEC Pallette experience)

Plans for FY 2000 (I)

- Gain *Quantitative* experience with Streams-C compiler
 - measure size/speed of compiler-generated vs. hand-generated designs
 - improve compiler performance
 - improve compiler reliability
- Goal: significant LANL applications coded, simulated, and run on hardware with Streams-C

Applications

- (Sarnoff) Contrast Enhancement
- (Honeywell) Wavelet
- (LANL) Hyperspectral Image Processing
 - Pixel Purity Index
 - K-Means

Plans for FY2000 (II)

- Integrate Streams-C into SLAAC environment
 - conform to SLAAC API for multi-board applications
 - re-target hardware libraries to SLAAC platform