Adaptive Computing Systems (ACS) Domain for Implementing DSP Algorithms in Reconfigurable Hardware

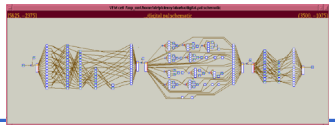
John Zaino, Eric Pauer, Ken Smith, Paul Fiore,
Jairam Ramanathan, Cory Myers
{john.c.zaino, ken.smith, paul.d.fiore, cory.s.myers}@lmco.com,
pauer@bit-net.com, jramanat@alum.mit.edu

Fourth Biennial Ptolemy Miniconference
22-23 March 2001

BAE SYSTEMS

Objective/Approach/Process

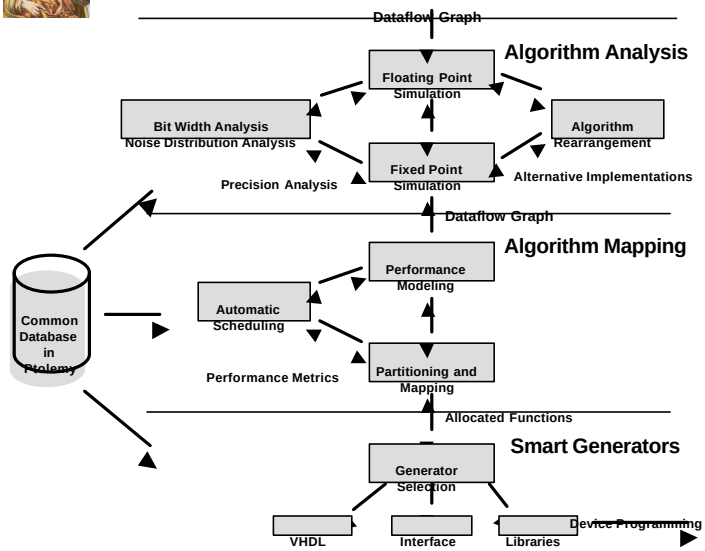
- Reconfigurable computing technology offers significant performance gains, e.g. 10X ops per watt and/or ops per cubic inch, over general purpose programmable solutions without the need to develop custom hardware. Today however, development of a working implementation requires hardware design expertise and generation of a good implementation requires many slow iterations between an algorithm designer and a hardware developer.
- Objective - reduce the design time for an initial implementation to hours and for an optimized implementation to days, for a range of signal processing applications
- Approach - provide the algorithm developer with tools to help analyze algorithms, understand their implications for hardware, and rapidly implement their chosen solutions
- In the process, isolate the algorithm developer from the hardware designer through a set of library elements that provide well-defined interfaces to both communities



Technical Attributes

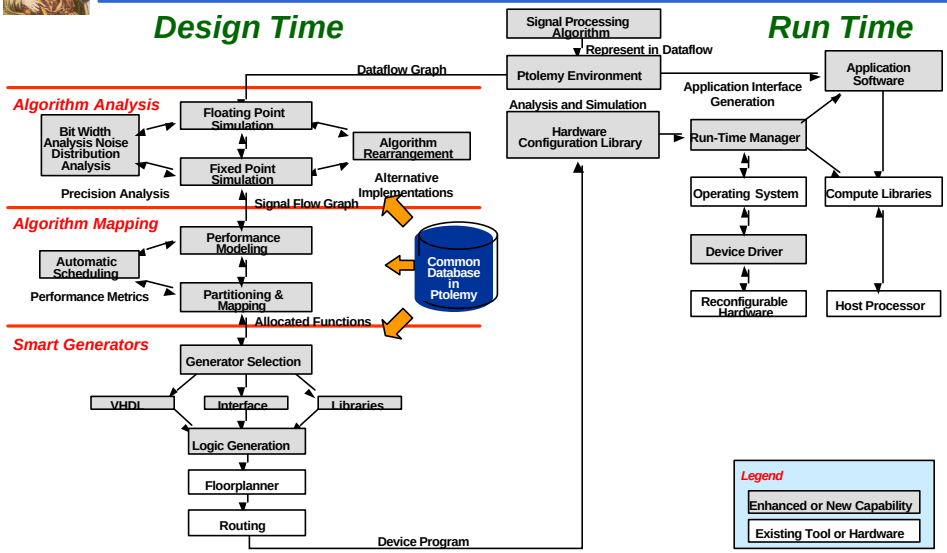
- Development of Adaptive Computing Systems domain under Ptolemy Classic
  - Allows alternative implementations from same dataflow graph
  - Provides floating point simulation, fixed point simulation, C code generation and VHDL code generation
  - Released first three versions of ACS domain in Ptolemy Classic
- End-to-end capability to map signal processing dataflow graph to working reconfigurable computing implementation
- Design space exploration automated
  - Bit width optimization theory (Markovian modeling) developed for algorithm analysis
  - Bit width optimization tool implemented to trade signal to noise ratio versus hardware complexity
- Pipeline alignment and scheduling algorithms implemented
  - Automatically generate “algorithm-specific” sequencer and memory control logic
  - Uni-rate and multi-rate signal processing
  - Single and multi-FPGA implementations
- Smart Generators- parameterizable algorithmic blocks

Analysis and Mapping in ACS Environment



- SNR analysis
- Alternative implementations
- Functional approximations
- Timing and sizing estimation
- Scheduling
- Partitioning across multiple FPGAs
- Device program
- Interface program

Design Approach



Program Progress

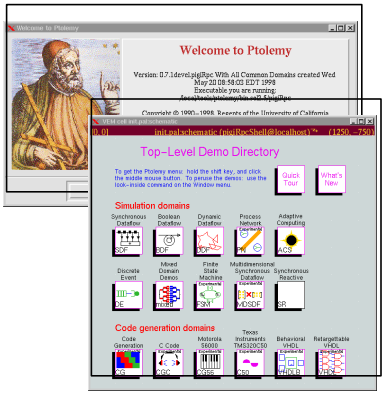
- **Algorithm analysis**
  - Representation for alternative implementations was incorporated as part of Ptolemy integration
  - Side-by-side simulation capability incorporated as part of Ptolemy integration
  - Developed bit width optimization theory for algorithm analysis and extended to include multiple devices and constraints
  - Implemented wordlength optimization tool
- **Algorithm mapping**
  - Cost analysis included as part of wordlength analysis
  - Implemented uni-rate and multi-rate pipeline alignment and scheduling algorithm for signal processing dataflow graphs
  - One-to-one and one-to-many mapping of functions to blocks supported

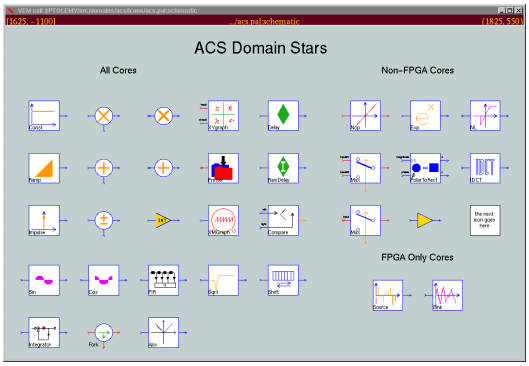
Program Progress

- Smart generators
  - Implemented portable logic synthesis methodology with VHDL as first target
  - Integrated Xilinx Core 4,000-series generators capability within VHDL code generation
  - Implemented smart generators for state machine sequencer and memory control (address generator)
- Ptolemy integration
  - Released initial version of "Adaptive Computing Systems" domain in Ptolemy in June 1998. Second release April 1999. Third release in August 2000.
  - ACS domain supports alternative implementations from a common interface. Floating point simulation, fixed point simulation, C code generation, and VHDL code generation.
- Demonstration
  - Selected Annapolis Micro Systems Wildforce™ board for demonstrations
  - Established ACS demonstration environment for Solaris
  - Integrated Wildforce™ board under Ptolemy
  - Demonstrated Winograd-based FSIC receiver and FFT-based signal detector
  - Procured and installed Annapolis Micro systems Wildstar™ board under Solaris
  - SHARP/HRR (High Range Resolution Radar ATR) algorithm modeled - hardware development & testing nearing completion

ACS Domain

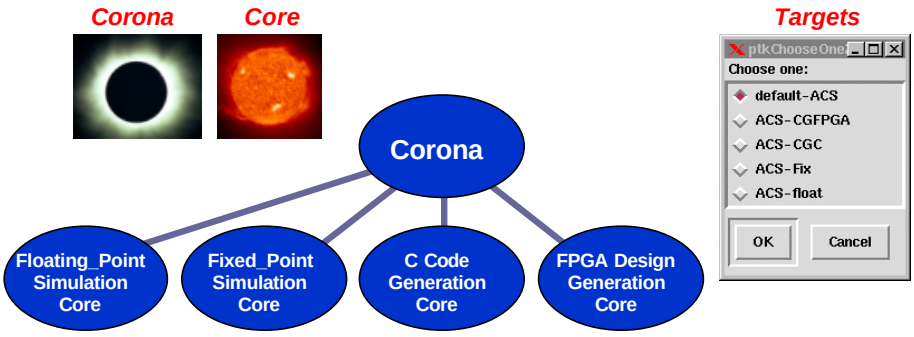
- Determined that extending old domains could not be justified
- New paradigm for Ptolemy, e.g. multiple implementations of a single star





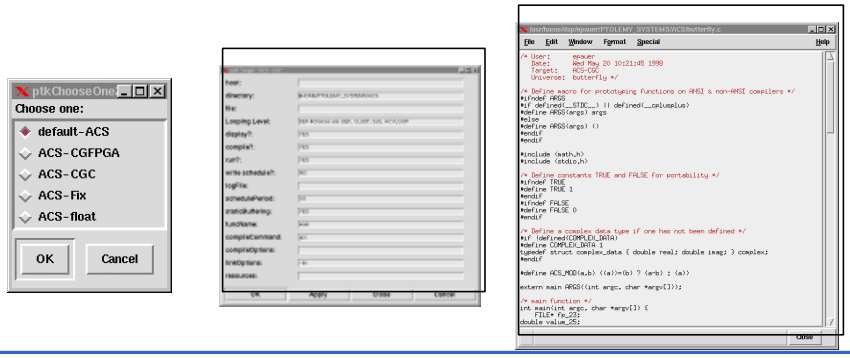
ACS Domain

- New ACS domain to facilitate movement among simulation and code/design generation
- Corona contains interface specification
- Core contains an implementation
- ACS Stars are composed of one corona and multiple cores
- Core selection via targeting defines implementation

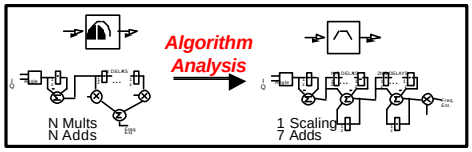


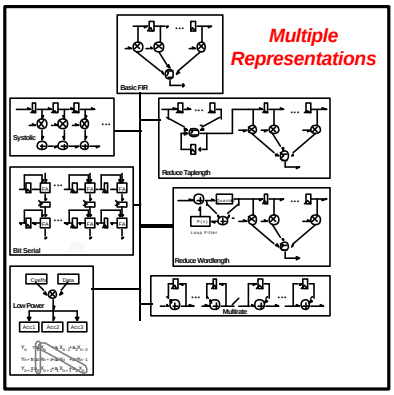
Selecting Among Alternative Implementations

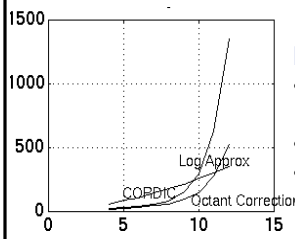
- Alternative implementations are represented as “targets” with cores for each star/functional block
- Targets can have parameters
- Floating point simulation, fixed point simulation, C code generation, and FPGA design generation are available.



Algorithm Analysis





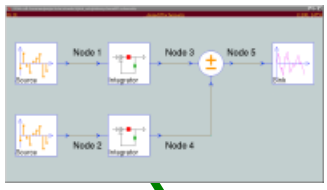


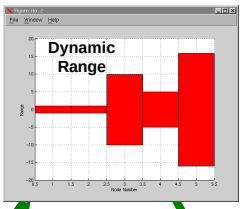
Implementation Trades

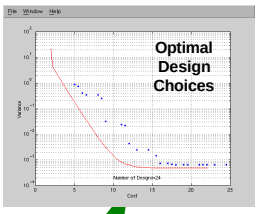
Perform Trade-Offs

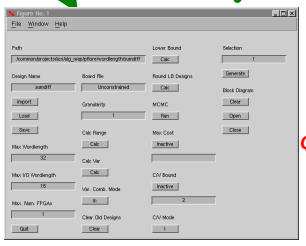
- **Precision (float vs. fixed, wordlengths)**
- **Speed**
- **Size/Area**
- **Latency**

Wordlength Optimization Analysis

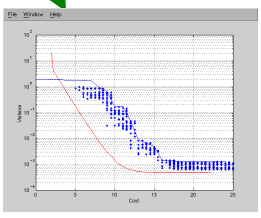








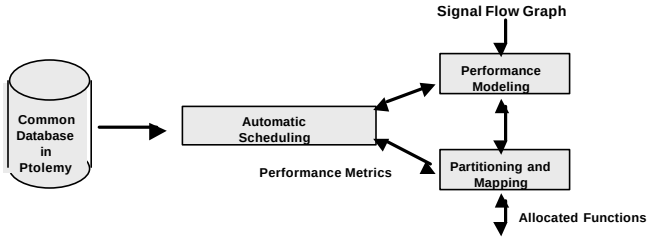
Quantization Noise (SNR)



Hardware Cost

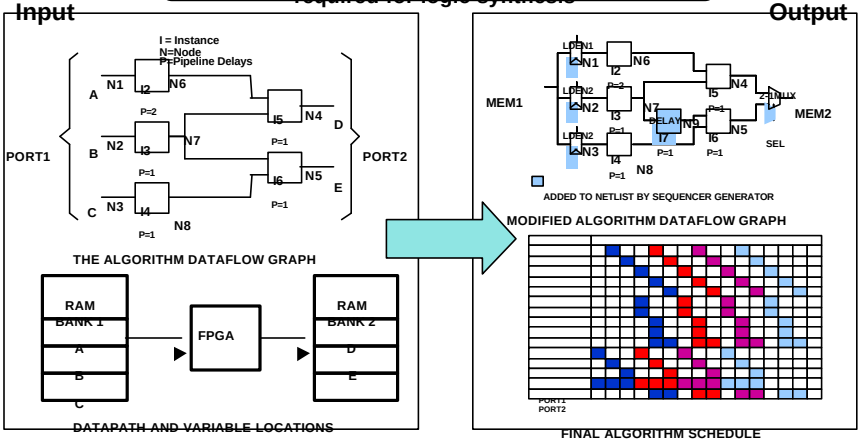
Algorithm Mapping

- Objectives
  - Performance Modeling – provide feedback on utilization, throughput, efficiency, etc. Feedback should be used by algorithm analysis capabilities.
  - Partitioning and Mapping – break large dataflow graphs into groups and map those groups across multiple devices and across time
  - Automatic Scheduling – automatically determine firing sequence, optimal mappings and sequence of configurations
- Progress
  - Cost analysis included as part of wordlength analysis
  - Implemented uni-rate and multi-rate pipeline alignment and scheduling algorithms
  - Memory allocation support



Automatic Scheduling

Pipeline alignment and schedule determination required for logic synthesis

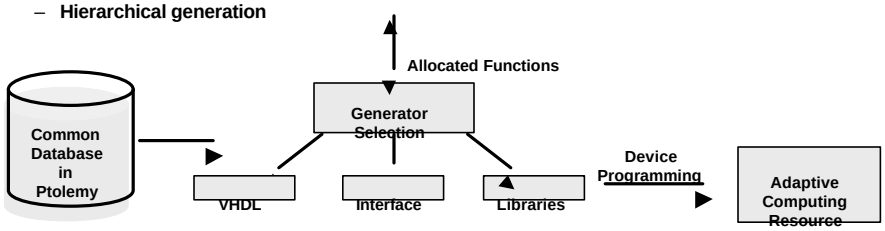


Processing Model

- Well-matched to Ptolemy Synchronous Dataflow (SDF) Domain
  - Unit or block token produce and consume amounts
  - Netlist structure determines execution order constraints
  - Pipeline delay information required to determine absolute timing
  - Delays are set to align pipelines for maximum throughput
  - Delay can be automatically determined from block parameters
- Combination of fully synchronous model and tagged synchronous models
  - No handshaking or tags but data is not always valid
  - Data validity is implicit in timing of latch signals
- Memory access fits same model
  - Data from common memory demuxed into separate streams running at lower rate
  - Data to common memory multiplexed to a single port
- Multiple FPGAs introduce additional pipeline delays
- Multi-rate parameterized execution

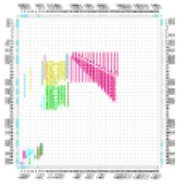
Smart Generators

- Objectives
  - Parameterized libraries – generate node implementations for specified bit widths and parameter values
  - Hierarchical representations – provide generators that can recursively call other generators
  - Interface generation – automatically generate software to move data between general-purpose processor and reconfigurable platform and to manage sequences of configurations
  - General synthesis – provide device independent representation of implementation
- Progress
  - Implemented portable logic synthesis methodology with VHDL as first target
  - Integrated Xilinx Core Generators (4000 Series) capability within VHDL code generation
  - Implemented smart generators for state machine and memory control
  - Hierarchical generation



Multi-FPGA Capability

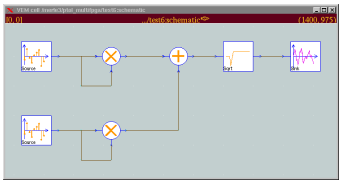
Design Generation for Single or Multiple FPGAs

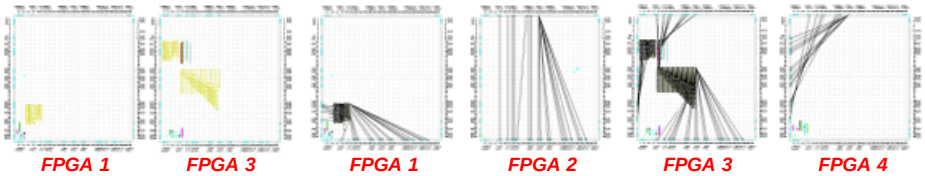


FPGA 1 Logic

Single FPGA Implementation

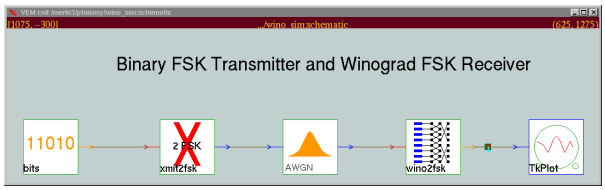
Multi-FPGA Implementation

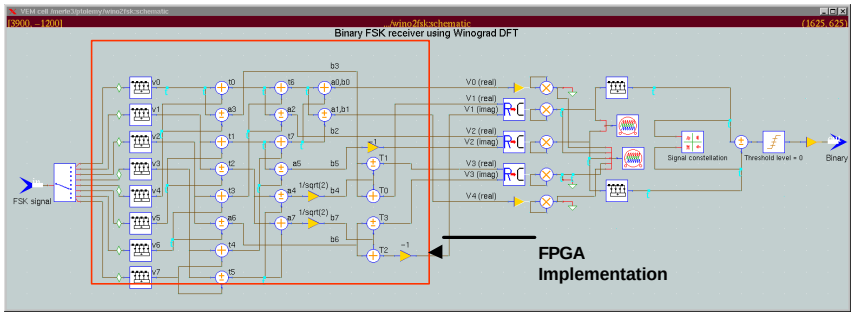




FPGA 1 Logic | *FPGA 3 Logic* | *FPGA 1 Routing* | *FPGA 2 Routing* | *FPGA 3 Routing* | *FPGA 4 Routing*

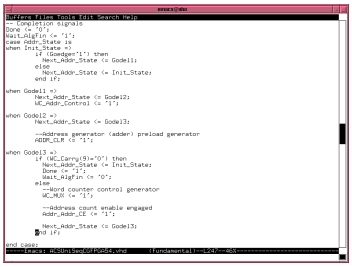
Winograd DFT-Based FSK Communications Receiver



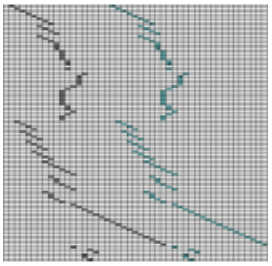


Results from FPGA-target / Back-end Tools

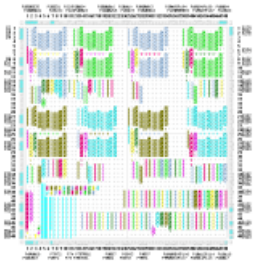
Generated VHDL



Generated Schedule

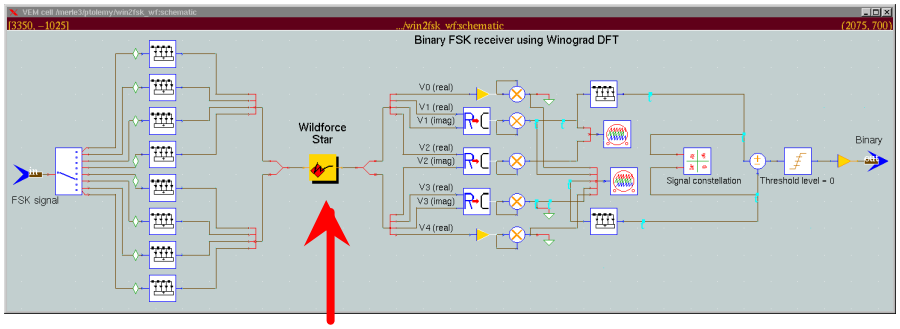


FPGA Design



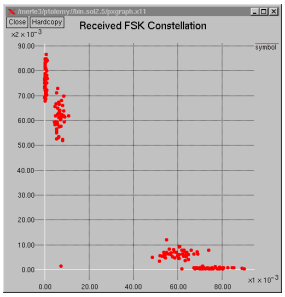
Hardware-in-the-Loop

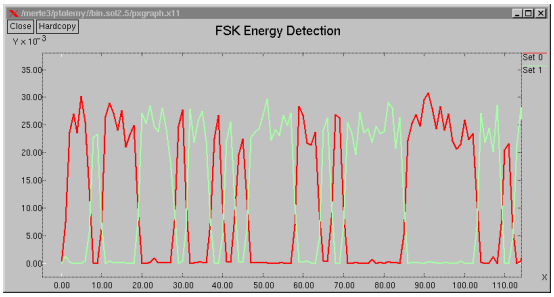
SDF Galaxy

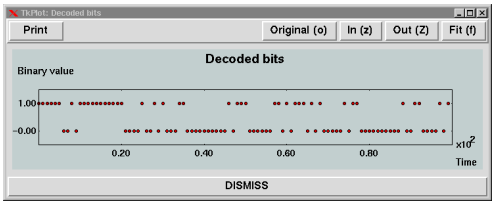


SDF Wildforce™ Star executes complete FPGA design in hardware on Annapolis Wildforce FPGA board

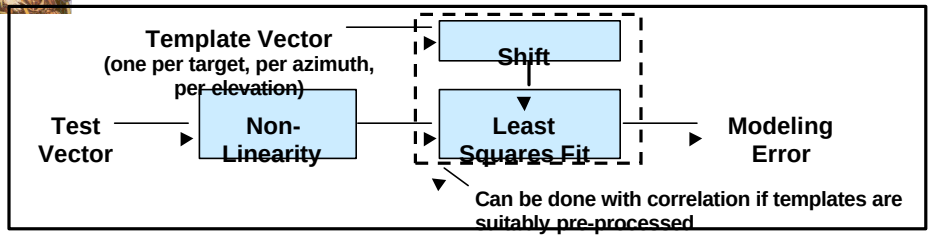
Processing Results







SHARP*/HRR Algorithm



Algorithm

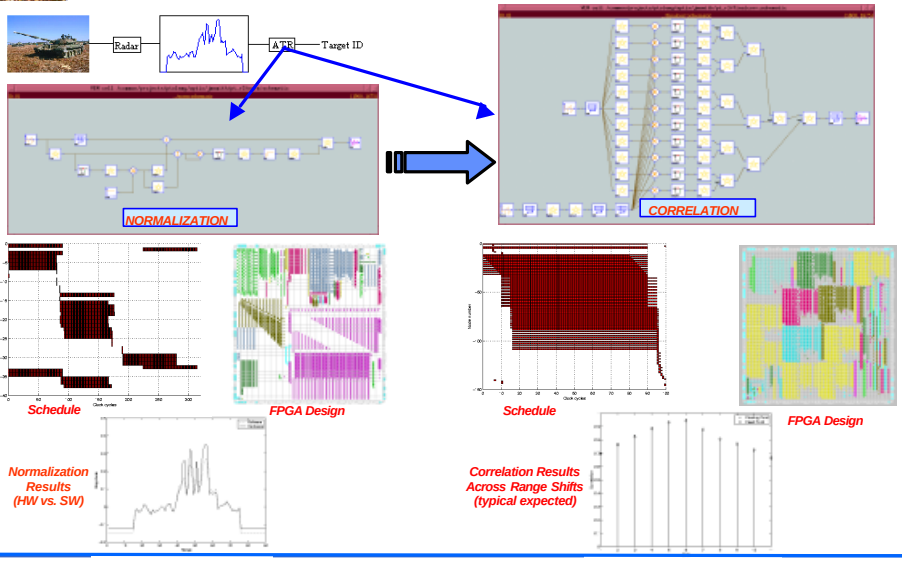
- Given test vector
  - For each template
    - For each shift
      - Compute least squares error
- Select template with minimum error

Complexity

- 70 data points per vector
- Number of shifts = 11 (in range)
- Number templates = 3,600/class
- 186 sec/class for 11 shifts on a Sun Ultra 5 (360 MHZ) workstation
- Expect 30x improvement

* System-oriented High Range Resolution (HRR) Automatic Recognition Program

SHARP/HRR Algorithm



ACS Tools - Facts and Figures

- 23 Functional Blocks (ACS stars) developed
- ~100 lines of code needed for new block/star
- Two ACS Architectures supported
  - Wildforce™ (4062XL)
  - Wildstar™ (XCV1000) (in progress)
- ~16,000 lines of C++ code developed
- ~15 min to generate VHDL for five FPGA design
- Explore ~1000 bit-width combinations in 1 minute
- Ptolemy Classic runs under Solaris and Linux