Many-core and Reconfigurable Supercomputing Conference (MRSC)
April 1-3, 2008

CHREC
NSF Center for High-Performance Reconfigurable Computing

Reformation and Formulation:
Key Challenges for Reconfigurable Supercomputing

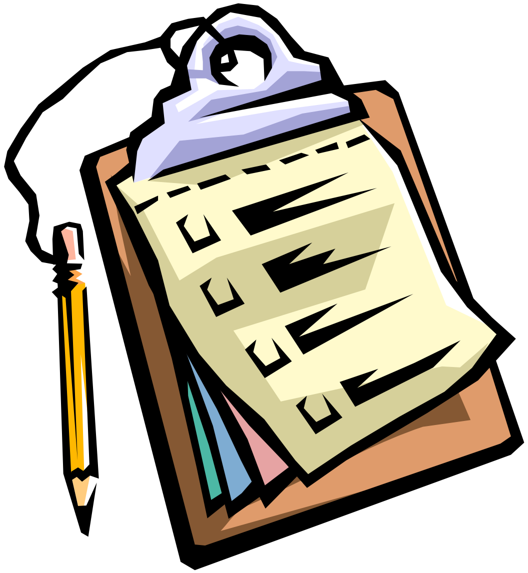
Alan D. George, Ph.D.

Director, NSF CHREC Center

Professor of ECE, University of Florida

Outline

- What is CHREC?
- Architecture Reformation
- Application Reformation
- Formulation Examples (CHREC)
- Conclusions



What is CHREC?

What is CHREC?

- NSF Center for High-Performance Reconfigurable Computing
  - Unique US national research center in this field, established Jan'07
  - Leading research groups in RC/HPC/HPEC @ four major universities
    - **University of Florida** (lead)
    - **George Washington University**

      } *founding sites (2007-)*
    - **Brigham Young University**
    - **Virginia Tech**

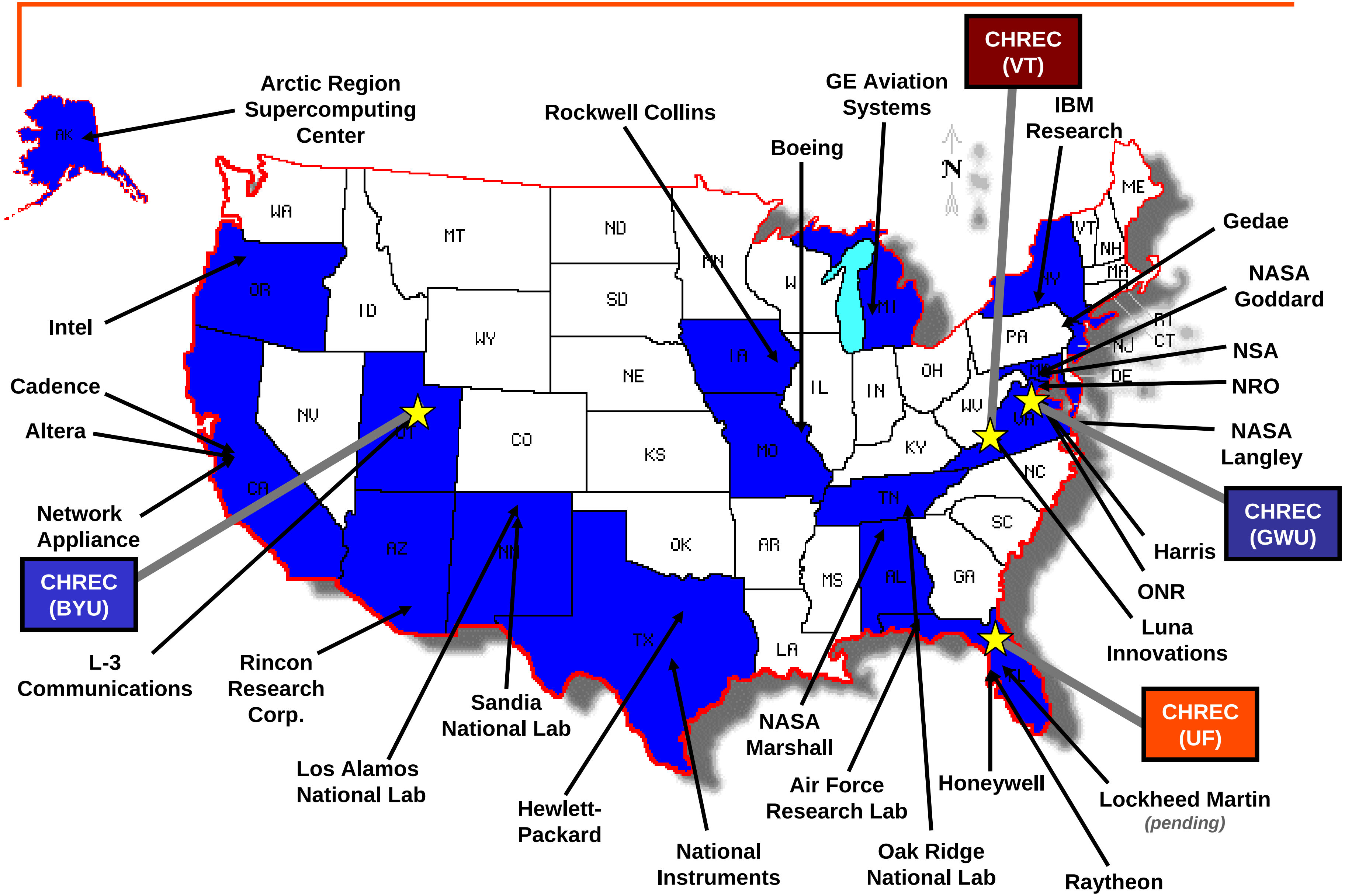
      } *expansion sites (2008-)*
- Under auspices of I/UCRC Program at NSF
  - **Industry/University Cooperative Research Center**
    - CHREC is supported by CISE & Engineering Directorates @ NSF
  - CHREC is both a National Center and a Research Consortium
    - University groups serve as research base (faculty, students, staff)
    - Industry & government organizations are research partners, sponsors, collaborators, advisory board, & technology-transfer recipients

CHREC Members

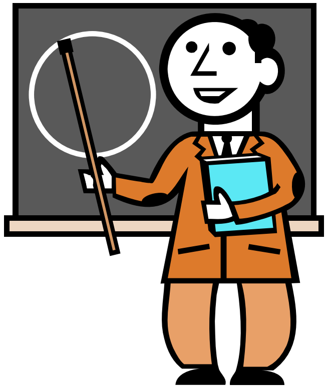
1. Air Force Research Lab
2. Altera
3. Arctic Region Supercomputing Center
4. Boeing [new]
5. Cadence
6. GE Aviation Systems
7. Gedae [new]
8. Harris Corp. [new]
9. Hewlett-Packard
10. Honeywell
11. IBM Research
12. Intel
13. L-3 Communications [new]
14. Los Alamos National Laboratory [new]
15. Luna Innovations [new]
16. NASA Goddard Space Flight Center
17. NASA Langley Research Center
18. NASA Marshall Space Flight Center
19. National Instruments [new]
20. National Reconnaissance Office
21. National Security Agency
22. Network Appliance [new]
23. Oak Ridge National Laboratory
24. Office of Naval Research
25. Raytheon
26. Rincon Research Corp. [new]
27. Rockwell Collins
28. Sandia National Laboratories

28 members with 38 memberships in 2008

Lockheed Martin – pending


CHREC (VT)
Arctic Region Supercomputing Center
Rockwell Collins
GE Aviation Systems
Boeing
IBM Research
Gedae
NASA Goddard
NSA
NRO
NASA Langley
CHREC (GWU)
Harris
ONR
Luna Innovations
CHREC (UF)
Lockheed Martin
(pending)
Raytheon
Honeywell
Oak Ridge National Lab
Air Force Research Lab
NASA Marshall
National Instruments
Hewlett-Packard
Sandia National Lab
Los Alamos National Lab
Rincon Research Corp.
L-3 Communications
CHREC (BYU)
Network Appliance
Altera
Cadence
Intel

CHREC Faculty (17 & growing)



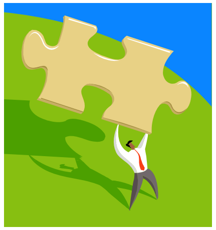
- **University of Florida (lead)**
  - **Dr. Alan D. George**, Professor of ECE – ***Center Director***
  - **Dr. Herman Lam**, Associate Professor of ECE
  - **Dr. K. Clint Slatton**, Assistant Professor of ECE and CCE
  - **Dr. Greg Stitt**, Assistant Professor of ECE
  - **Dr. Ann Gordon-Ross**, Assistant Professor of ECE
  - **Dr. Saumil Merchant**, Research Scientist in ECE
- **George Washington University (partner)**
  - **Dr. Tarek El-Ghazawi**, Professor of ECE – ***GWU Site Director***
  - **Dr. Ivan Gonzalez**, Research Scientist in ECE
  - **Dr. Sergio Lopez**, Research Scientist in ECE
- **Brigham Young University (partner)**
  - **Dr. Brent E. Nelson**, Professor of ECE – ***BYU Site Director***
  - **Dr. Michael J. Wirthlin**, Associate Professor of ECE
  - **Dr. Michael Rice,** Professor of ECE
  - **Dr. Brad L. Hutchings**, Professor of ECE
- **Virginia Tech (partner)**
  - **Dr. Shawn A. Bohner**, Associate Professor of CS – ***VT Site Director***
  - **Dr. Peter Athanas**, Professor of ECE
  - **Dr. Wu-Chun Feng**, Associate Professor of CS and ECE
  - **Dr. Francis K.H. Quek**, Professor of CS

CHREC features a strong team of ~40 graduate students spanning the four university sites.

Architecture Reformation

Architecture Reformation

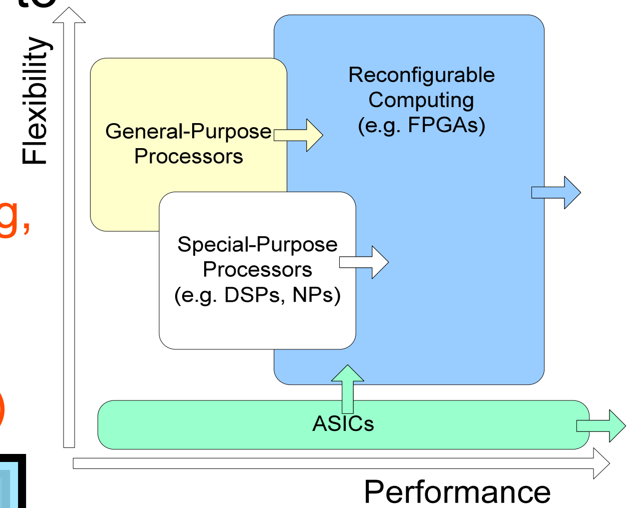
- End of wave (Moore's Law) riding f_{clk} + ILP (CPU)
  - Explicit parallelism & multicore the new wave
- Many promising technologies on new wave
  - Fixed & reconfigurable multicore device architectures
- Many R&D challenges lie on new wave
  - Tried & true methods no longer sufficient; complexity abounds
  - Semantic gap widening between applications & systems
    - e.g. App developers must now understand & exploit parallelism



- Inherent traits of fixed device architectures
  - App-specific: inflexible, expensive (e.g. ASIC)
  - App-generic: power, cooling, & speed challenges (e.g. Opteron)
  - Many niches between extremes (Cell, DSP, GPU, NP, etc.)
- Reconfigurable architectures promise best of both worlds
  - Speed, flexibility, low-power, adaptability, economy of scale, size
  - Bridging embedded & general-purpose computing, superset of fixed

What is a Reconfigurable Computer?

- System capable of changing hardware structure to address application demands
  - Static or dynamic reconfiguration
  - Reconfigurable computing, configurable computing, custom computing, adaptive computing, etc.
  - Often a mix of conventional fixed & reconfigurable devices (e.g. control-flow CPUs, data-flow FPLDs)
- Enabling technology?
  - Field-programmable multicore devices
  - FPGA is "King" (but space is broadening)
- Applications?
  - Vast range – computing and embedded worlds
  - Faster, smaller, less power & heat, adaptable & versatile, selectable precision, high comp. density



FPGA
ECA
FPCA
FPOA
MPPA
TILE
XPP
et al.

Opportunities for RC?

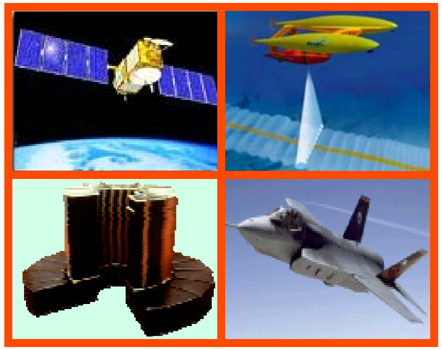
10-100x speedups with
2-10x energy savings
not uncommon

From Satellites to Supercomputers!

When and Where to Apply RC?

Performance ↑
Power ↓

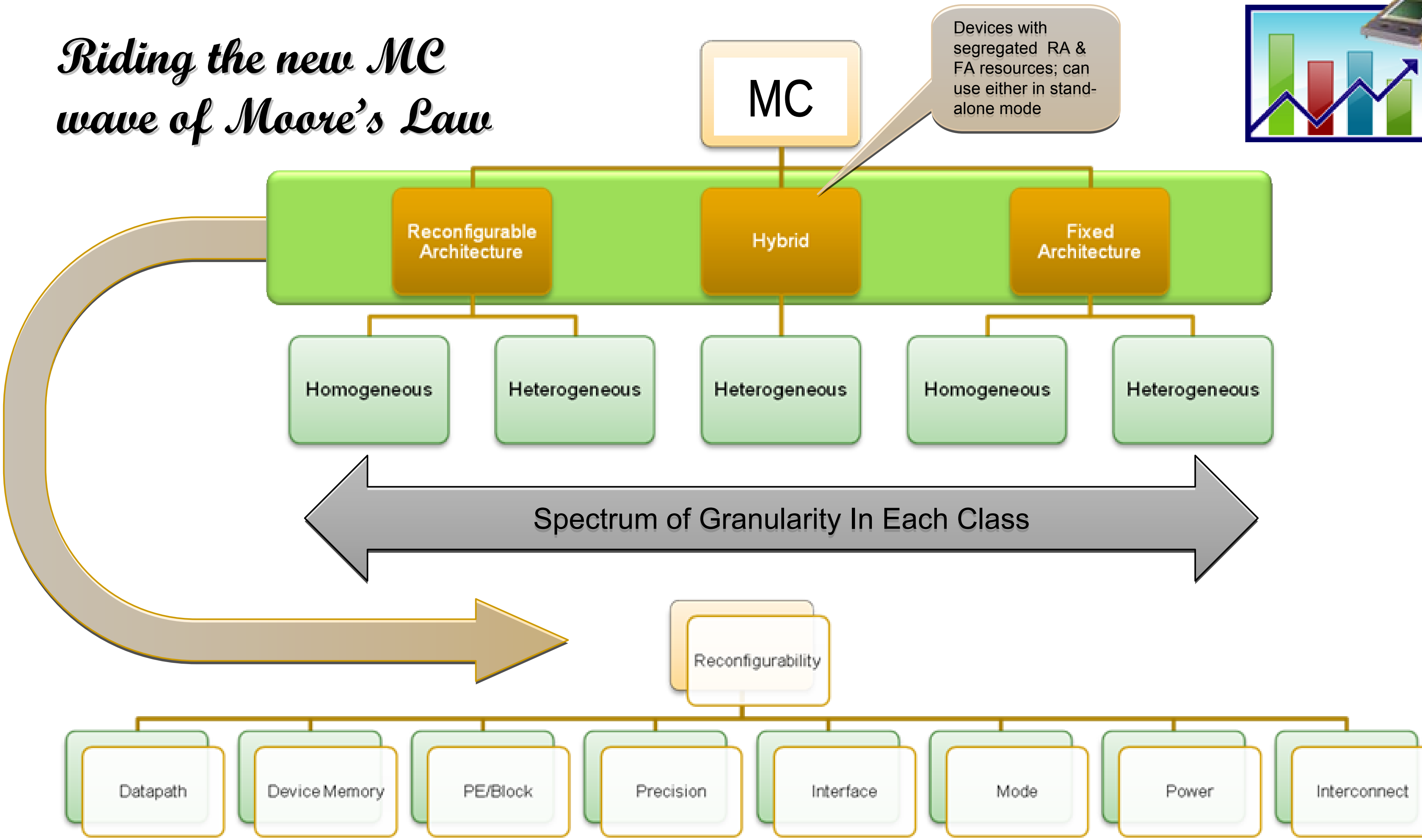
- When do we need?
  - When performance & versatility are critical
    - Hardware gates targeted to application-specific requirements
    - System mission or applications change over time
  - When the environment is restrictive
    - Limited power, weight, area, volume, etc.
    - Limited communications bandwidth for work offload
  - When autonomy and adaptivity are paramount



- Where do we need?
  - In conventional servers, clusters, and supercomputers (HPC)
    - Field-programmable hardware fits many demands
    - High DoP, finer grain, direct data-flow mapping, bit manipulation, selectable precision, direct control over H/W (e.g. perf. vs. power)
  - In space, air, sea, undersea, and ground systems (HPEC)
    - Embedded & deployable systems can reap many advantages w/ RC

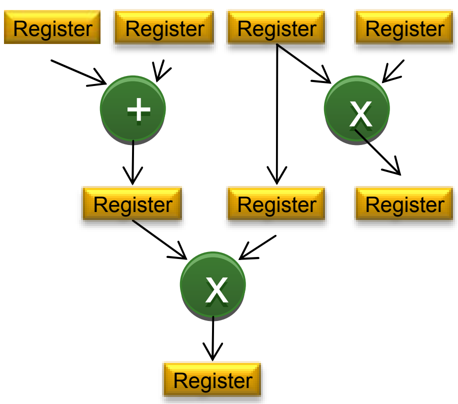
Multicore/Many-Core Taxonomy

Riding the new MC wave of Moore's Law

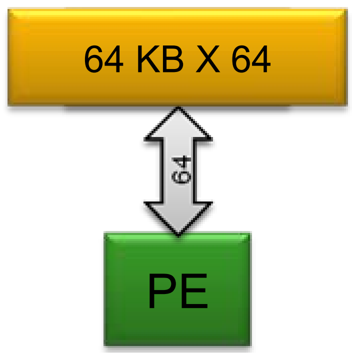


Reconfigurability Factors

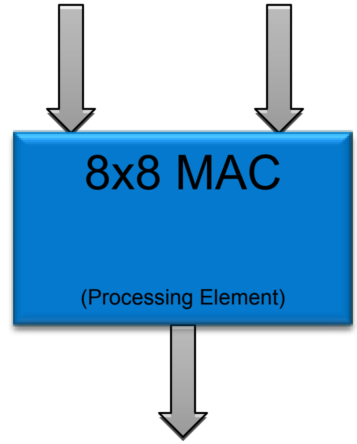
Datapath



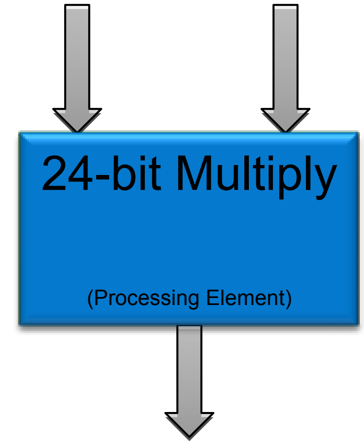
Device Memory



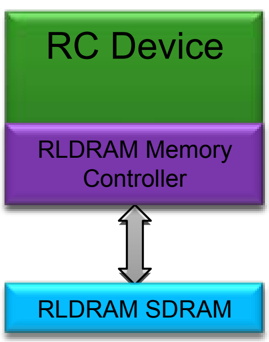
PE/Block



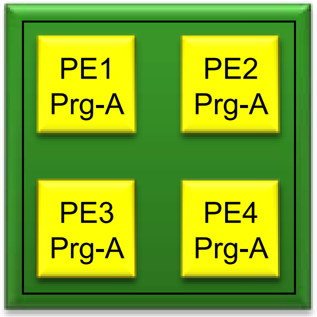
Precision



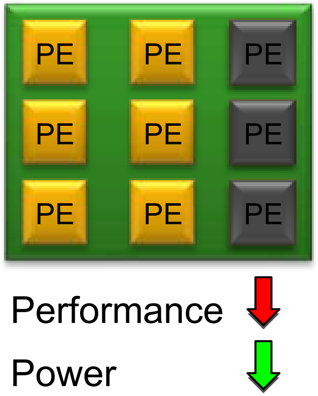
Interface



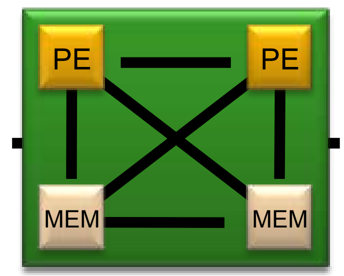
Mode



Power

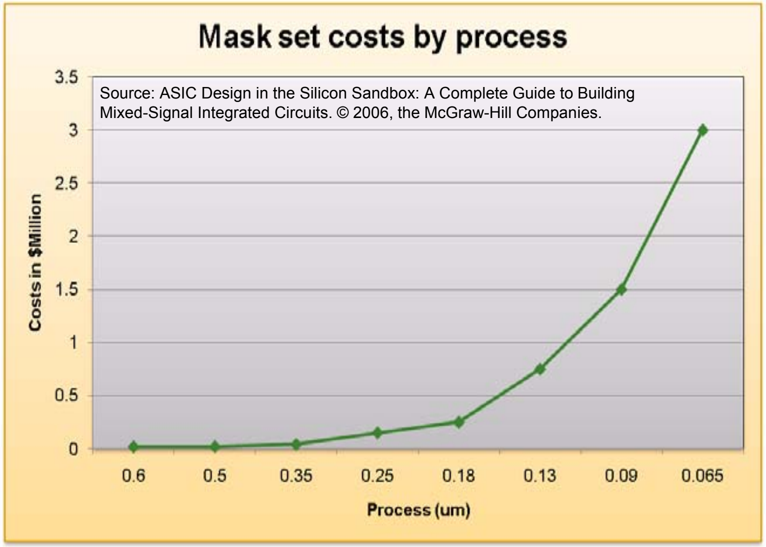


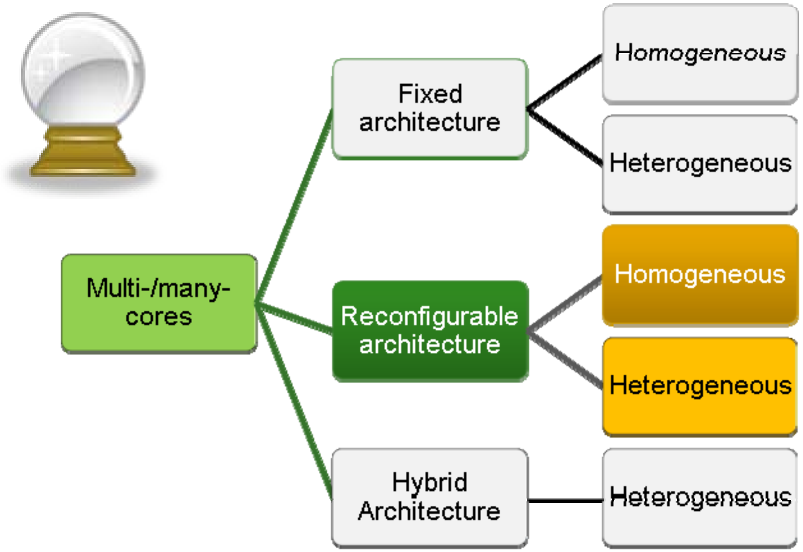
Interconnect



Future Convergence

- Rising development costs & other factors drive convergence
  - As seen in many other technologies
- Device architecture convergence?
  - Many-core driven by densities
  - Heterogeneous?
    - Cell as initial example
    - Intel and AMD both cite heterogeneous MC in their future
    - To extent complexity is manageable
  - Reconfigurable
    - Performance + versatility
    - Adaptive for many apps, missions
    - Avoid limitations of fixed architectures
    - Manage issues of heterogeneity

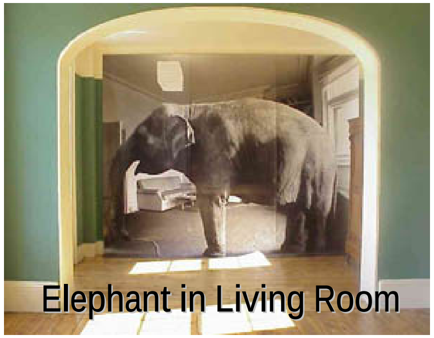




Application Reformation

Application Reformation

- Dawn of reformation in application development methods
  - Driven by architecture reformation; complexity management
  - Holistic concepts, methods, & tools must emerge
- Semantic gap widening between apps & archs
  - MC world (fixed or RC), explicit parallelism
    - Architectures increasingly complex to target by apps
    - New to fixed MC world, familiar to RC/FPGA & HPC worlds
  - Optimizing compiler ≠ parallelizing compiler
    - ***Domain scientist*** involved in comp. structure of their app
- How do we bridge semantic gap?
  - Focus upon computational fundamentals
    - Formal models, complexity management via abstraction, encapsulation
  - Learn lessons from other engineering fields
    - e.g. aerospace engineers do not flight-test first, why must we?
  - Build basis for an RC engineering discipline
    - Leverage where practical for fixed MC world


Elephant in Living Room

Reformation in App Development

- **FDTE** as formal model
  - **I. Formulation**
    - Strategic design playground, abstraction, prediction
  - **II. Design**
    - Tactics, coding, details
  - **III. Translation**
    - Conversion to executable form
  - **IV. Execution**
    - Services, debug, optimization
- Applies throughout computing
  - We focus on RC, which involves hardware & software design

Spectrum of application development phases

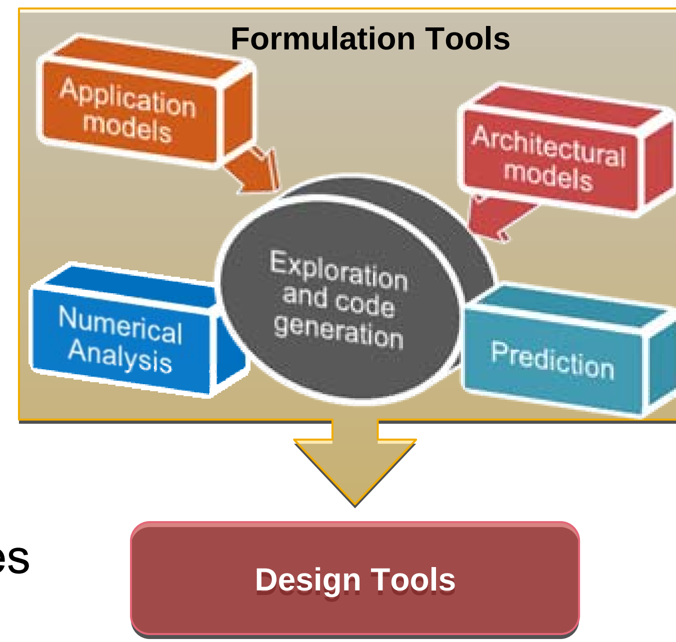
I. Formulation
(a) Algorithm design exploration
(b) Architecture design exploration
(c) Performance prediction (speed, area, etc.)
II. Design
(a) Linguistic design semantics and syntax
(b) Graphical design semantics and syntax
(c) Hardware/software codesign
III. Translation
(a) Compilation
(b) Libraries and linkage
(c) Technology mapping (synthesis, place & route)
IV. Execution
(a) Test, debug, and verification
(b) Performance analysis and optimization
(c) Run-time services

FDTE Model

> "We need a change in mindset, not simply another programming language."

I. Formulation

- Strategic exploration
  - ***Not*** coding in traditional sense
- Parallel algorithm exploration
  - Control structures (wide, deep)
  - Data structures (elements, precision, layout)
- Parallel architecture exploration
  - As mapping targets of parallel algorithm
  - Base characteristics (e.g. DoP, OPS, B/W)
- High-level performance prediction
  - Supports tradeoff analysis (alg, arch, both)
  - Memory hierarchy, data locality, bottlenecks
  - Analytical, simulative, or combo
- Feeder to **D**esign phase
  - Patterns, templates, code generation, libraries
- *Theme: strategic design decisions*



FDTE Model (continued)

II. Design

- Linguistic design semantics & syntax
- Graphical design semantics & syntax
- Hardware/software coding, co-design

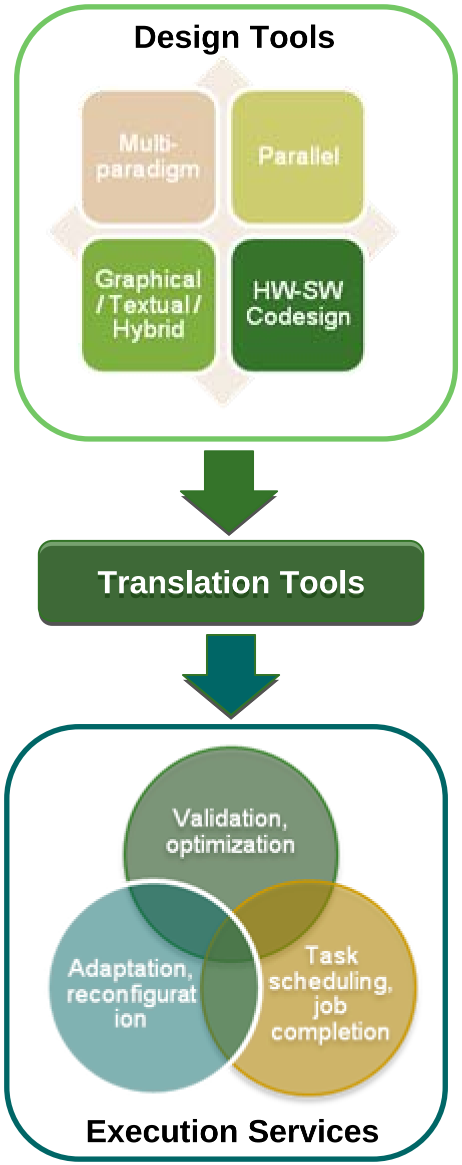
III. Translation

- Compilation
- Libraries & linkage
- Technology mapping (synthesis, PAR)

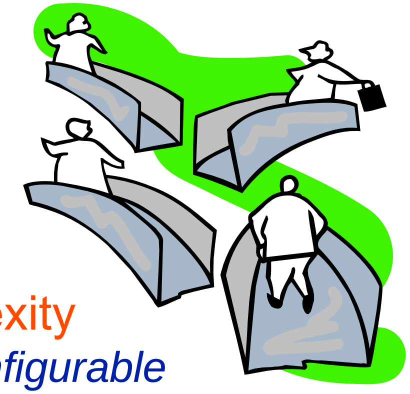
> DTE phases traditionally used for "seat of pants" formulation, but increasingly *inefficient* and *inappropriate*.

IV. Execution

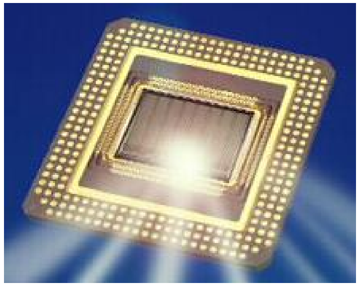
- Test, debug, & verification
- Performance analysis & optimization
- Run-time services



Benefits of Formulation



- Management of increasing complexity
  - Ideal level for exploring structures, mappings, tradeoffs
  - Major (strategic) decisions prior to coding, manage complexity
    - *Increasingly important in MC: fixed or heterogeneous or reconfigurable*
  - Basis for achieving semi-automation
- Major reduction in **DTE** costs
  - **Design**
    - Reduction in D_{freq} and D_{time} and thus cost ($D_c = D_{\text{time}} \times D_{\text{freq}}$)
      - Better strategies incoming means less design & re-design
      - Transitions from **F** to **D** (automation, patterns, templates, code)
  - **Translation & Execution**
    - Reduction in T_{freq} and thus cost ($T_c = T_{\text{time}} \times T_{\text{freq}}$)
    - Similarly for E_c



- Notional example (with & without Formulation)

$$C_{dev} = \sum_i (F_c + D_c + T_c + E_c)_i = (0 + 202\,hrs + 20\,hrs + 5\,hrs)_{i=1} + (0 + 150\,hrs + 20\,hrs + 5\,hrs)_{i=2} = 402\,hrs$$

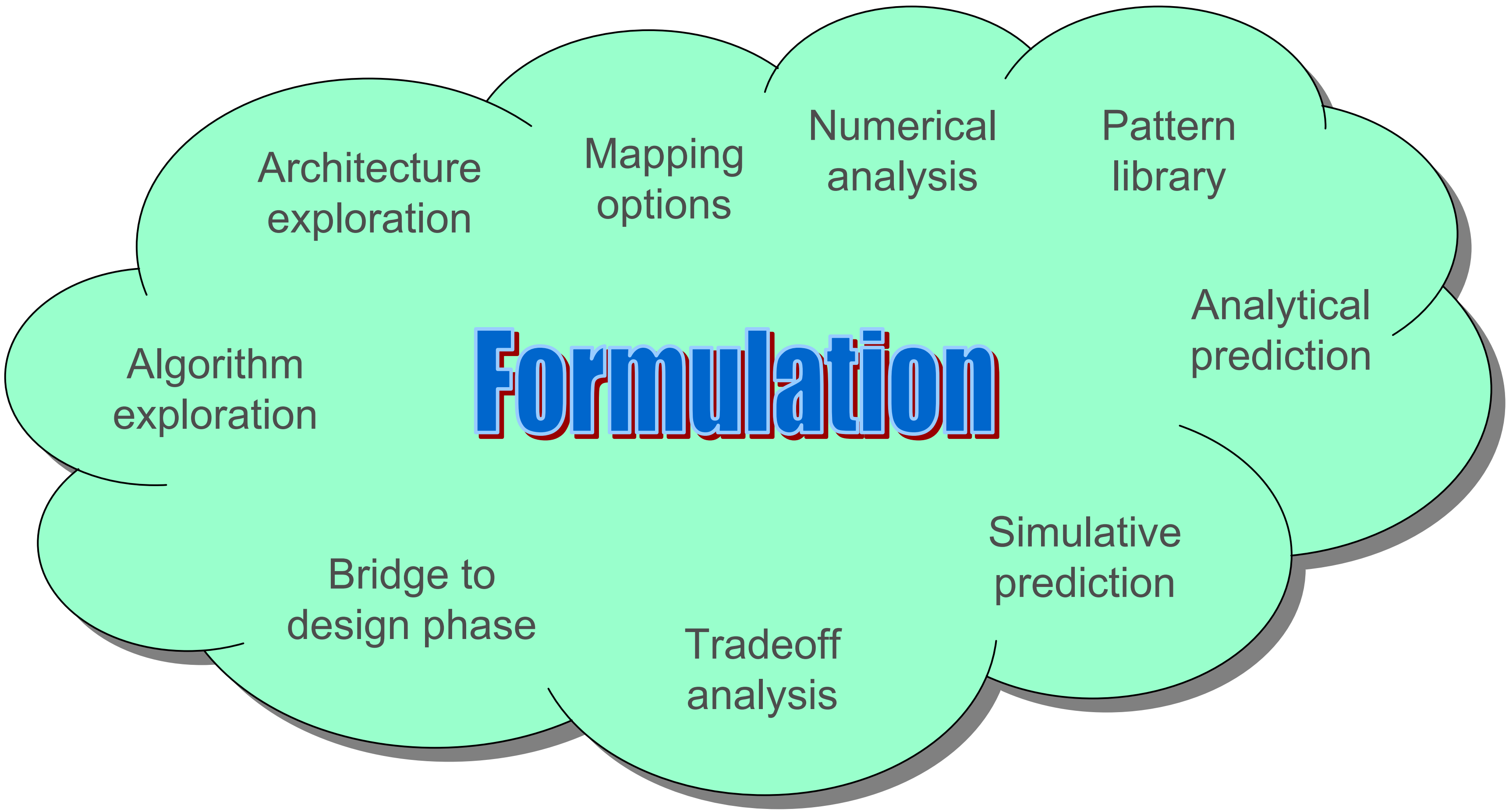
Without

$$\psi = \frac{U}{C}$$

$$C_{dev} = \sum_i (F_c + D_c + T_c + E_c)_i = 2\,hrs + 18\,hrs + 3\,hrs + 2\,hrs = 25\,hrs$$

With (16x better)

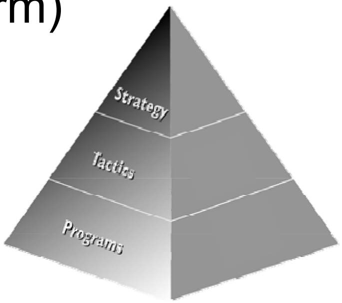
CHREC Examples



RAT: RC Amenability Test

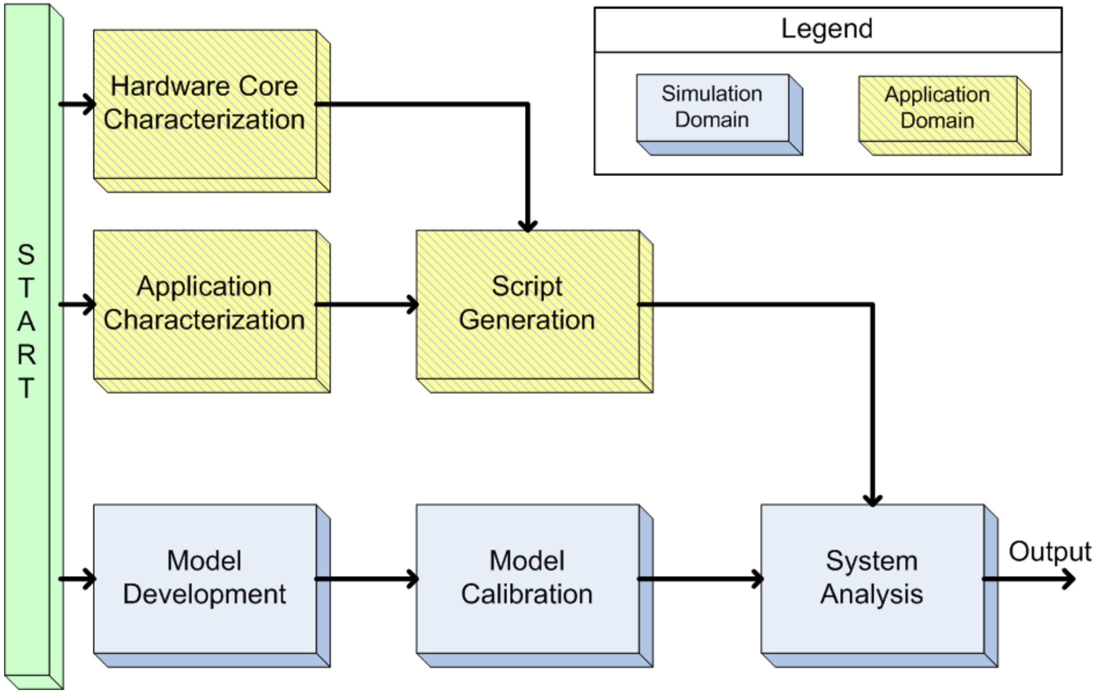
- Variety of single-FPGA apps studies conducted @ UF Site of CHREC in 2007
  - LIDAR processing, Molecular Dynamics simulation, PDF estimation, Multichannel DDC
- First, start with high-level formulation ("back of envelope") & prediction
  - Using **RAT**, developed by CHREC
    - Enter basic parms. of parallel alg. plus general platform data into RAT table
    - Outcome is predicted speedup of *that* algorithm on *that* platform
    - Iterative process with algorithm, precision, platform changes until satisfied

- Next, perform detailed design & coding of alg. in language of choice
  - LIDAR & DDC coded in AccelDSP, MD in Impulse-C, PDF in VHDL
- Translate & execute on platform of choice
  - Suitable platform already determined with aid of RAT, much fewer iterations in **T&E**
- Evaluate results (wall-clock speedup vs. fast CPU on same platform)
  - **LIDAR**: predicted = 11.2, actual = 13.1 [on Cray XD1]
  - **MD**: predicted = 10.7, actual = 6.6 [on XDI XD1000]
  - **PDF**: predicted = 13.0, actual = 20.6 [on Cray XD1]
  - **DDC**: predicted = 26.1, actual = 22.6 [on Nallatech H101-PCIXM]



RC Simulation Framework

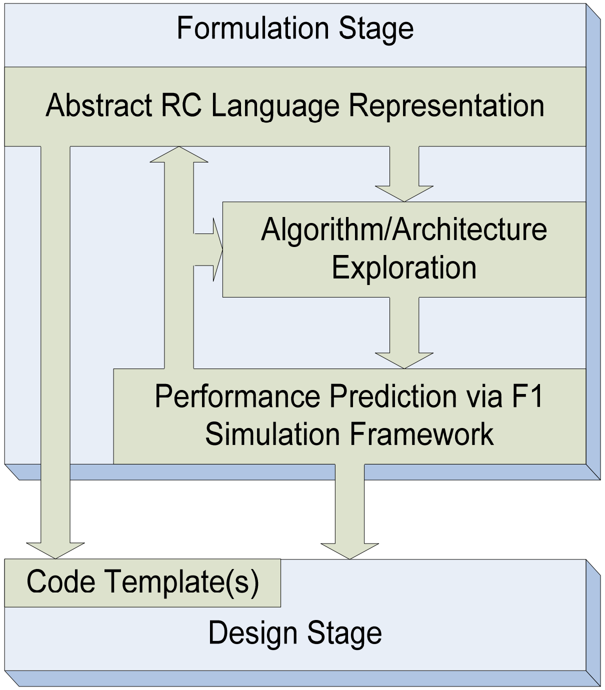
- 6 key components of framework depicted in figure
  - Many key tasks can be completed independently and in parallel
- Framework allows arbitrary applications to be simulated on any arbitrary systems
  - Component models & application scripts can be reused for rapid simulative analyses



RC Simulation Framework Diagram

- System models driven by application scripts produce simulative performance prediction results
  - Systems modeled in 2007 include socket-based FPGA platform (XD1000), PCI-based platform (Nallatech), and proprietary FPGA platform (SRC-6)

RCML Abstraction Layer



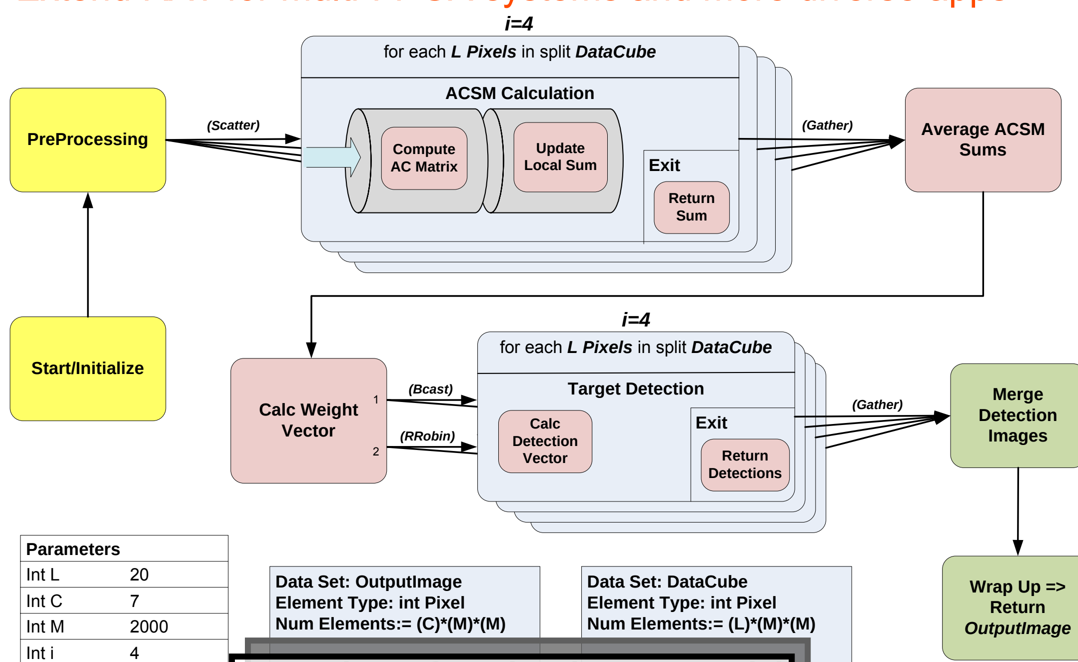
Conceptual flow of RC formulation stage under study

Motivations

- Formulation is often neglected, bypassed during RC development
- Provide user-friendly streamlined interface to simulative analysis tools under development in CHREC

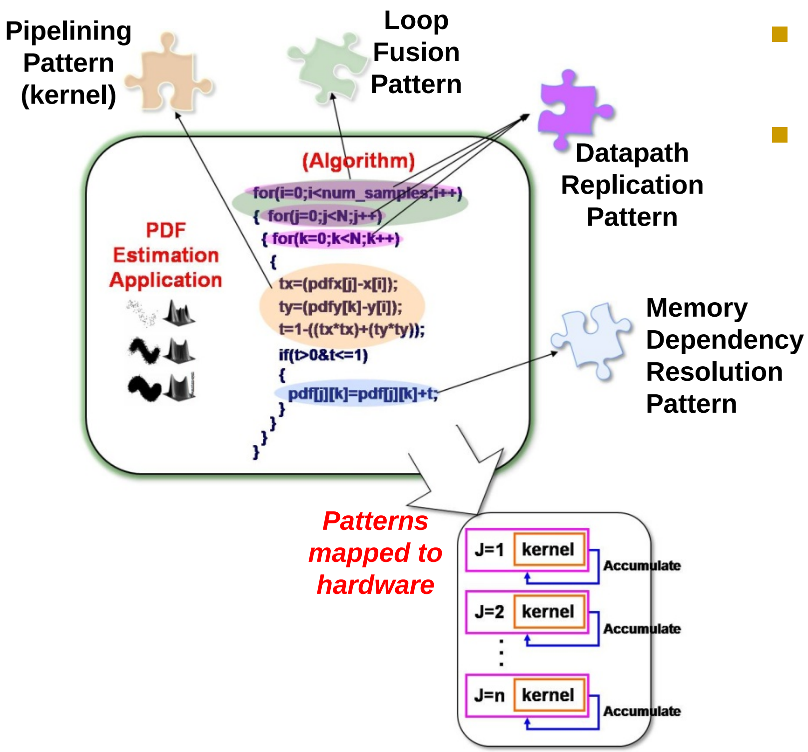
Primary goals and tasks

- Research concepts for RC abstraction layer in app formulation
  - Enable specification of algorithm and architecture via new formulation language, called RCML (on top of AADL)
  - Define mapping from RCML to performance prediction models (RAT, SIM) for exploration, tradeoff analysis
  - Demonstrate methods using proof-of-concept case studies
- Extend RAT for multi-FPGA systems and more diverse apps

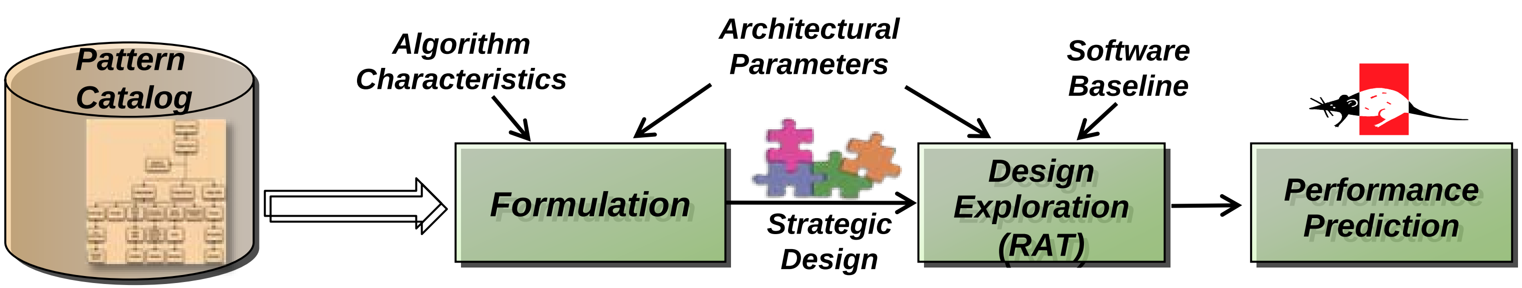


Mockup of top-level RCML model for Hyperspectral Imaging app

Formulation & Design Patterns



- Goal – Investigate methods for *strategic design* through pattern-based formulation and design
- Approach
  - Catalogue & classify patterns for use in a model-based RC design methodology
    - Computation patterns (exploit & express parallelism)
    - Communication patterns (regulate flow of data)
    - Interface patterns (define pattern interface/boundary)
  - Formulation:
    - Explore algorithmic and architectural alternatives (pattern-based preliminary design)
    - Parameterize design patterns to efficiently exploit performance prediction tools (e.g., RAT) and modeling languages (RCML)
    - Mapping to detailed design patterns (tactical design) to automate generation of code/code templates

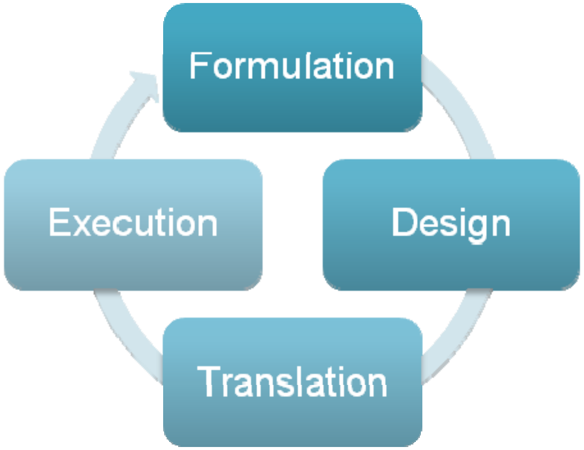


DARPA Studies @ CHREC

- Research roadmaps for app development on FPGA systems
  - Bridging app/arch semantic gap
    - Prevalent challenge of multi-core
  - RC to revolutionize DoD missions
- 2 DARPA studies by CHREC
  - One @ founding sites + Clemson
  - One @ expansion sites
- Focus areas
  - Study underlying tools limitations
    - Theory, practice, technologies
  - Formulate strategic research paths
    - Revolutionary, impactful
  - Craft proposed research roadmaps
    - Highlight DARPA-hard challenges

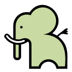
Titles of Two Studies for DARPA

- **Exploration of a Research Roadmap for Application Development & Execution on FPGA-based Systems**
- **Future FPGA Design Methodologies and Tool Flows**



Conclusions

Conclusions

- Computing technologies undergoing reformation
  - Architecture – MC, HC, RC, convergence as costs escalate
  - Application development – expressing parallelism
- "Elephant in living room" 
  - Widening semantic gap between apps & architectures
  - Traditional programming models & methods alone inadequate
- **FDTE** model
  - Many good concepts & tools in **DTE** to leverage
  - Missing link & potential salvation is **F**ormulation
    - Critically important for arch & app reformations
    - Bridge across semantic gap; complexity management
  - **F**ormulation as strategic playground of computational structure
    - Algorithm & architecture exploration, prediction, tradeoff analysis
    - Transitions from **F** to **D** (**F/D** patterns, templates, code generation, libraries)
    - More time spent in **F** leads to much less time in **D**, **T**, and **E**
      - **D** (coding), **T** (compile, PAR), and **E** (bugs & bottlenecks) increasingly expensive

Formulation
Design
Translation
Execution

Conclusions

- Many research challenges to enable **F**ormulation
  - Abstraction layer for strategic exploration (alg, arch, map)
    - Increasingly important for future: fixed, heterogeneous, or reconfigurable
    - New generation of theories, models, concepts & tools
    - Supportive of domain scientists, not merely EE/CS gurus
  - Preliminary results in CHREC projects show potential for **F**
    - Emphasis upon FPGA-based systems, but may apply more broadly
      - Simple & quick method to predict strategic performance (RAT)
      - Simulation infrastructure for strategic performance prediction (F1)
      - Abstraction layer to express & explore alg, arch, mappings (RCML)
      - Algorithm patterns for **F** and as bridge between **F** & **D** stages (FRS)
      - Research roadmap exploration for future R&D activities (DARPA)

- Many educational challenges as well, for example:
  - Formulation in computing curriculum
    - Common in most engineering fields, deficient in computing studies
  - Numerical analysis in computing curriculum
    - Understanding issues with dynamic range, resource usage, error

CHREC NSF Center for High-Performance Reconfigurable Computing

UF UNIVERSITY of FLORIDA · Virginia Tech · BYU BRIGHAM YOUNG UNIVERSITY · THE GEORGE WASHINGTON UNIVERSITY

Thanks for Listening! ☺

- For more info:
  - www.chrec.org
  - george@chrec.org

- Questions?