A Research Agenda for Improving Reconfigurable Computing Design Productivity

http://www.chrec.org/ftsw/

Mike Wirthlin

Brigham Young University

The views, opinions, and/or findings contained in this presentation are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

This work was supported by the Defense Advanced Research Projects Agency - Information Processing Techniques Office (DARPA/IPTO) under contracts FA8650-07-C-7745 and FA8650-07-1-7742 and administered by AFRL/RYDI.

FPGA Design Productivity Crisis

The cost of creating FPGA designs in both high-performance computing and embedded environments limits the ability of DoD to exploit FPGA and reconfigurable technology.

- DARPA interested in understanding and addressing this topic
  - Studies (BYU/VT, UF/GWU, and others)
  - SBIR tool efforts
  - Possible program?

Study Objectives

This study will investigate the *full* FPGA tool flow and identify potential solutions in *all* stages of the tool flow to provide *revolutionary* improvements in design productivity

Objectives:

1. Summarize the state of existing FPGA design tools,
2. Identify the limitations of the existing tool flows,
3. Identify techniques to address these limitations,
4. Estimate the impact of the identified techniques, and
5. Recommend a strategy for future research and investment.

Future FPGA Trends

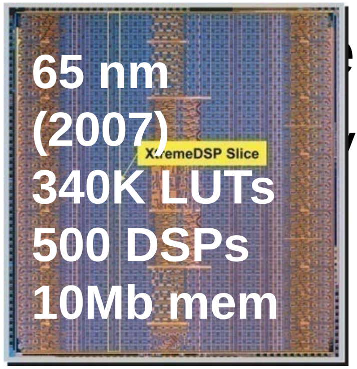
Technology	Year	LUTs	DSPs	Memory
65 nm	2007	340 k	500	10 Mbit
45 nm	2010	700 k	1000	21 MBit
32 nm	2013	1,400 k	2000	42 MBit
22 nm	2016	2,900 k	4300	89 MBit

- Coarser granularity objects & hard IP
- Adopt highest-speed I/O standards
- Increasingly heterogeneous (similar to multi-core)

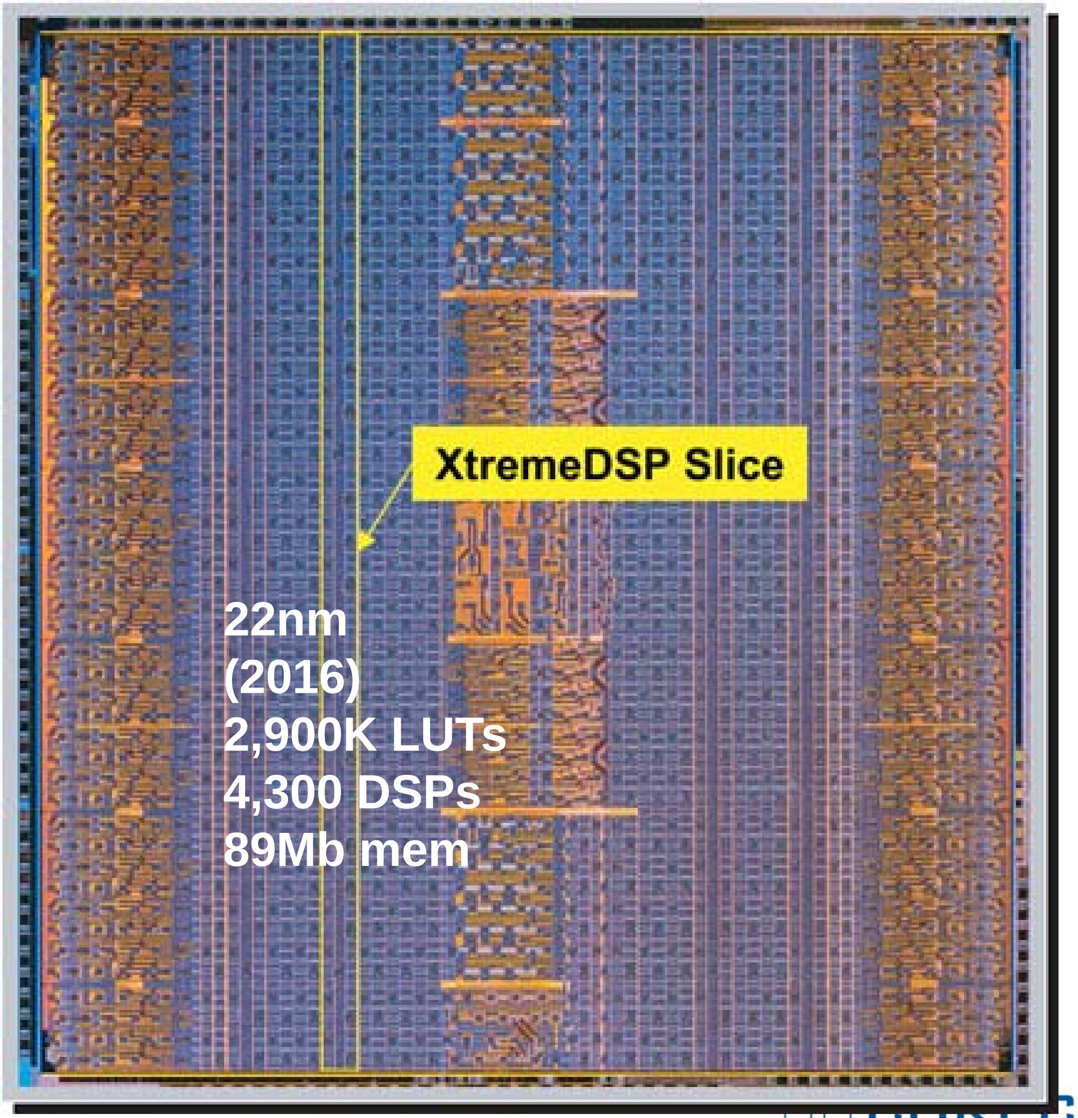
Future FPGA Trends

Technology	Y
65 nm	2
45 nm	2
32 nm	2
22 nm	2

- Coarser granularity
- est-spee
- y hetero

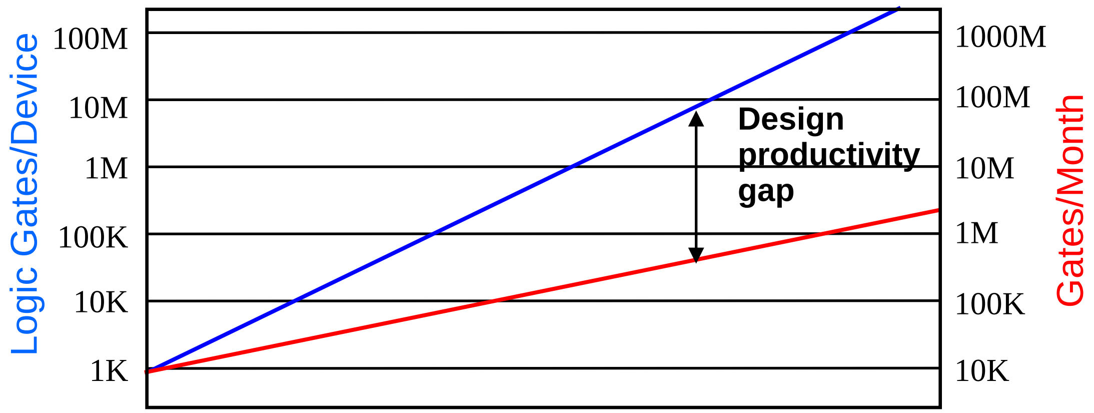


65 nm
(2007)
340K LUTs
500 DSPs
10Mb mem



22nm
(2016)
2,900K LUTs
4,300 DSPs
89Mb mem

Design Productivity Gap



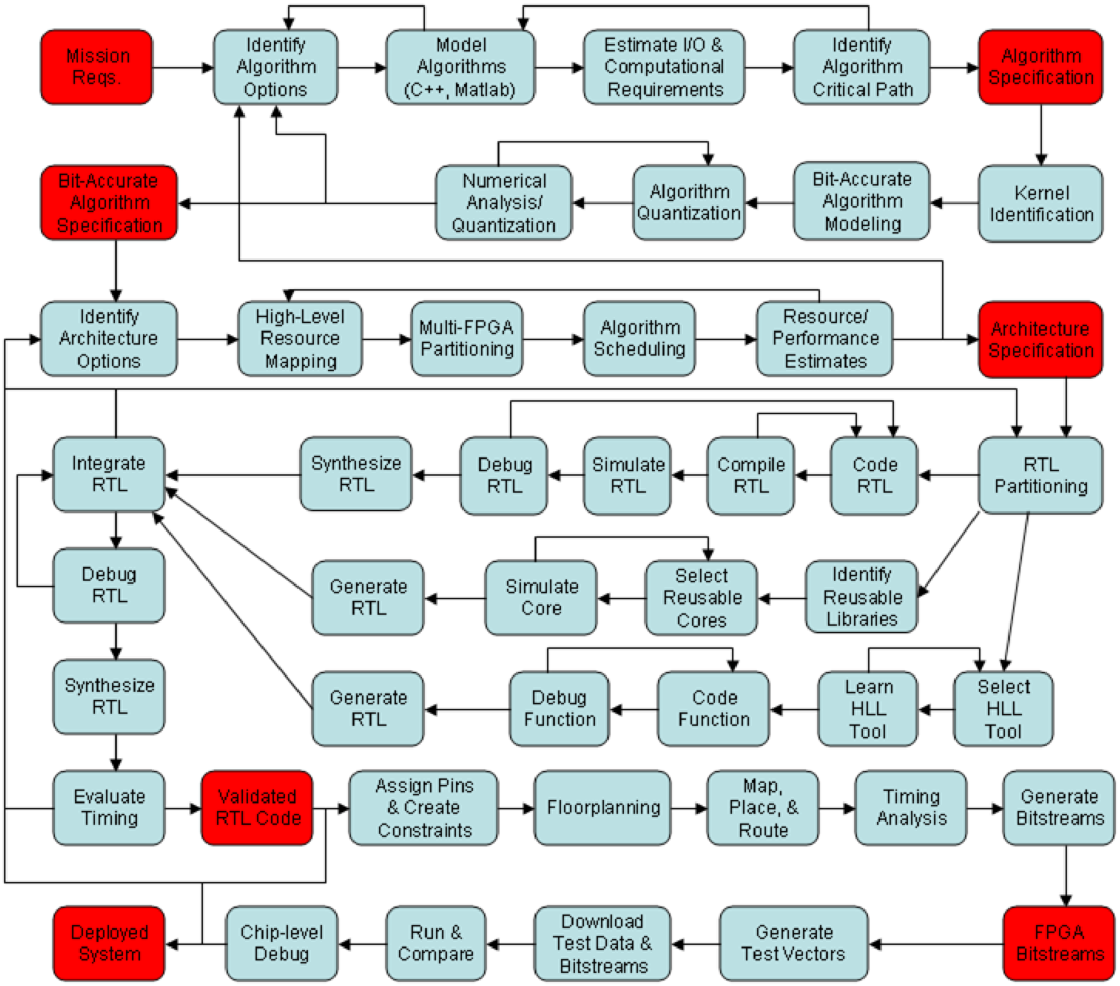
There is no equivalent law for design productivity

- Density of FPGAs growing rapidly (Moore’s law)
- Improvements in design productivity are modest

- Problem: growing gap between FPGA density and design productivity

How are we going to design future FPGAs?

State of the Art Design Flow



Mission Reqs.

Identify Algorithm Options

Model Algorithms (C++, Matlab)

Estimate I/O & Computational Requirements

Identify Algorithm Critical Path

Algorithm Specification

Bit-Accurate Algorithm Specification

Numerical Analysis/ Quantization

Algorithm Quantization

Bit-Accurate Algorithm Modeling

Kernel Identification

Identify Architecture Options

High-Level Resource Mapping

Multi-FPGA Partitioning

Algorithm Scheduling

Resource/ Performance Estimates

Architecture Specification

Integrate RTL

Synthesize RTL

Debug RTL

Simulate RTL

Compile RTL

Code RTL

RTL Partitioning

Debug RTL

Generate RTL

Simulate Core

Select Reusable Cores

Identify Reusable Libraries

Synthesize RTL

Generate RTL

Debug Function

Code Function

Learn HLL Tool

Select HLL Tool

Evaluate Timing

Validated RTL Code

Assign Pins & Create Constraints

Floorplanning

Map, Place, & Route

Timing Analysis

Generate Bitstreams

Deployed System

Chip-level Debug

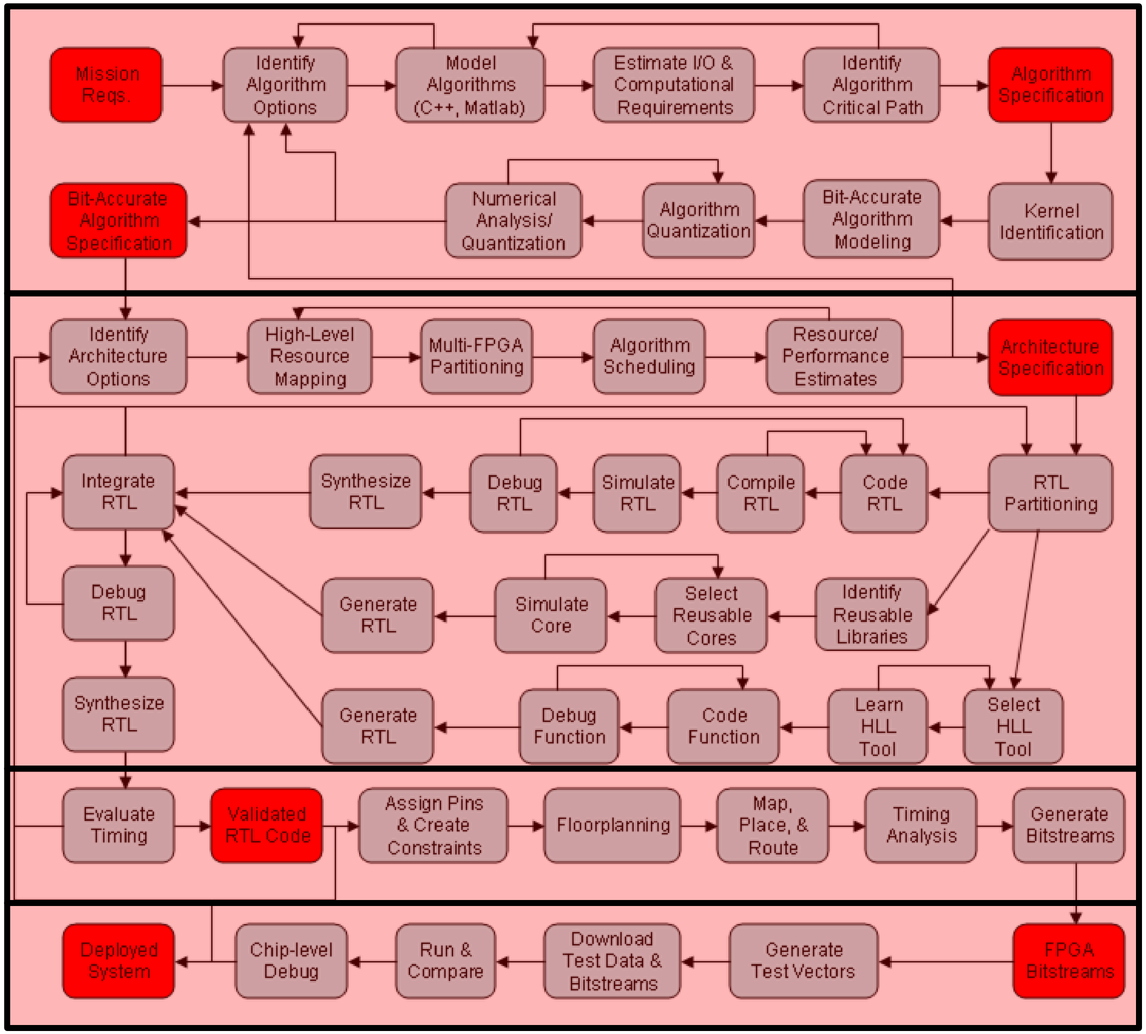
Run & Compare

Download Test Data & Bitstreams

Generate Test Vectors

FPGA Bitstreams

State of the Art Design Flow



Mission Reqs.

Identify Algorithm Options

Model Algorithms (C++, Matlab)

Estimate I/O & Computational Requirements

Identify Algorithm Critical Path

Algorithm Specification

Kernel Identification

Bit-Accurate Algorithm Modeling

Algorithm Quantization

Numerical Analysis/ Quantization

Bit-Accurate Algorithm Specification

Identify Architecture Options

High-Level Resource Mapping

Multi-FPGA Partitioning

Algorithm Scheduling

Resource/ Performance Estimates

Architecture Specification

RTL Partitioning

Code RTL

Compile RTL

Simulate RTL

Debug RTL

Synthesize RTL

Integrate RTL

Identify Reusable Libraries

Select Reusable Cores

Simulate Core

Generate RTL

Debug RTL

Select HLL Tool

Learn HLL Tool

Code Function

Debug Function

Generate RTL

Synthesize RTL

Evaluate Timing

Validated RTL Code

Assign Pins & Create Constraints

Floorplanning

Map, Place, & Route

Timing Analysis

Generate Bitstreams

FPGA Bitstreams

Generate Test Vectors

Download Test Data & Bitstreams

Run & Compare

Chip-level Debug

Deployed System

Formulation

Design

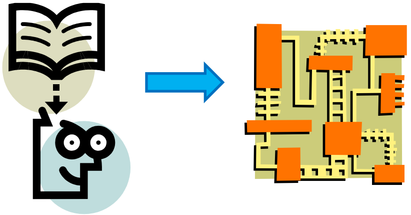
Translation

Execution

CHREC
NSF Center for High-Performance Reconfigurable Computing

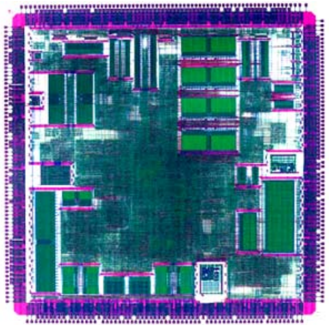
Challenges and Limitations

- Formulation
  - Few tools to support architecture exploration
  - Limited support for predictive analysis
- Design
  - Require understanding of circuit structures and timing
  - Limited support for high-level language constructs
- Translation
  - PAR times long and unpredictable
  - Difficult to integrate cores
- Execution
  - Limited visibility into FPGA state
  - Runtime systems for FPGAs lacking
  - Lack of support for multi-FPGA



FPGA Use Models

ASIC Replacement

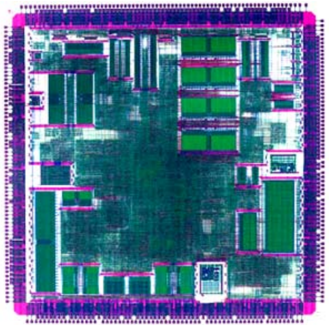


- General purpose digital circuits
- Low-level RTL design
  - Clock-cycle accurate
  - Carefully manage resources
- Goal: minimize cost

FPGA Use Models

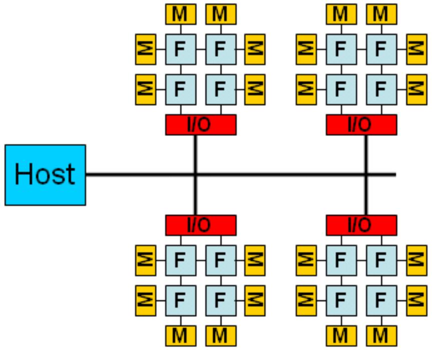
ASIC Replacement

- General purpose digital circuits
- Low-level RTL design
  - Clock-cycle accurate
  - Carefully manage resources
- Goal: minimize cost



Configurable Computing

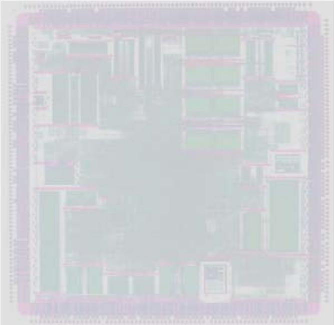
- Application-specific *computing*
- Fixed topology and I/O interfaces
- Higher level design (focus on behavior)
- Goal: maximize performance



FPGA Use Models

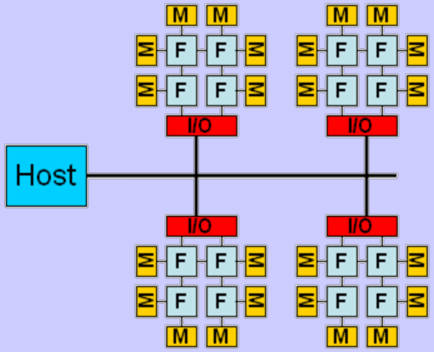
ASIC Replacement

- General purpose digital circuits
- Low-level RTL design
  - Clock-cycle accurate
  - Carefully manage resources
- Goal: minimize cost



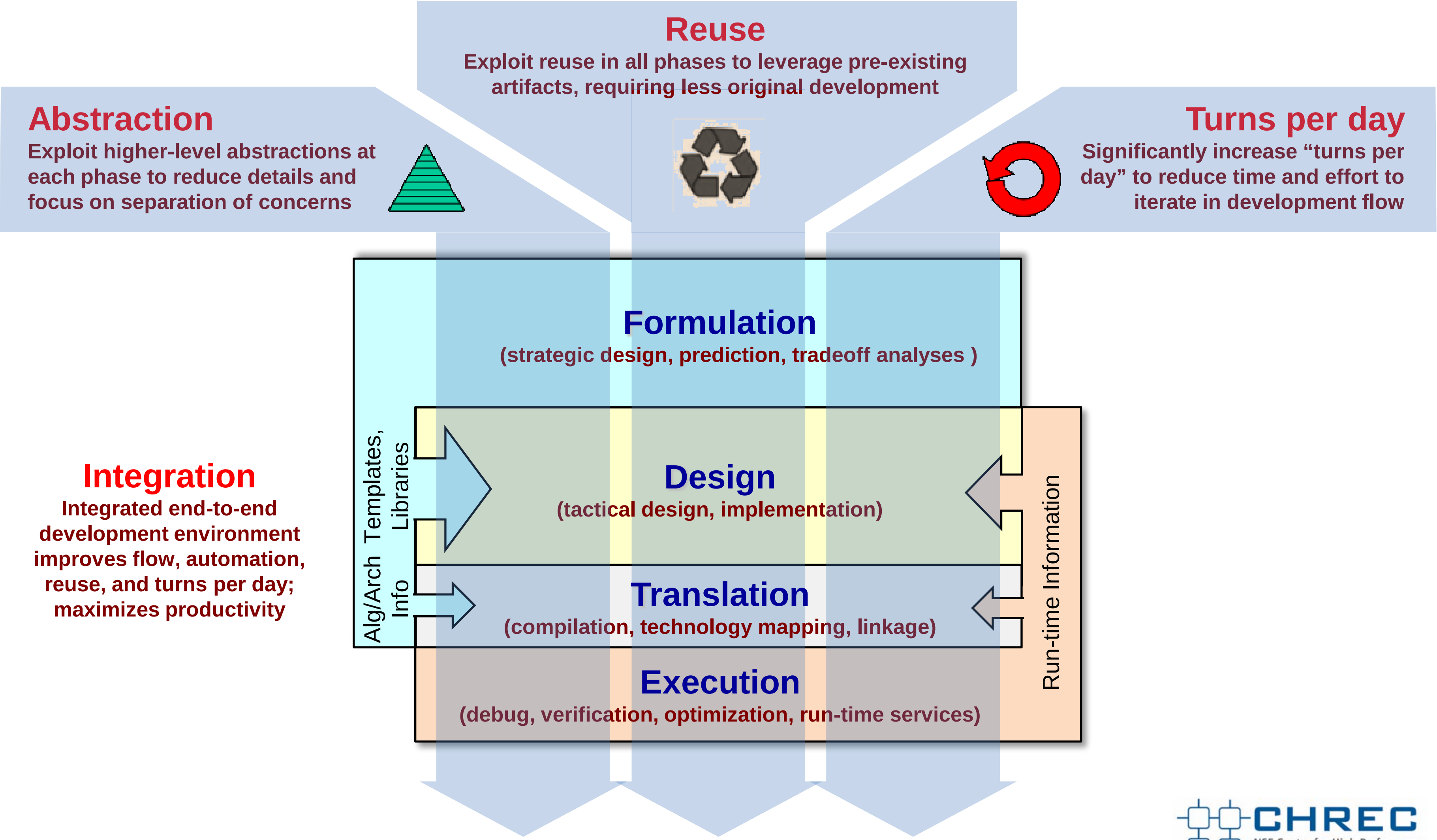
Configurable Computing

- Application-specific *computing*
- Fixed topology and I/O interfaces
- Higher level design (focus on behavior)
- Goal: maximize performance



This study will address the Configurable Computing Use Model

Research Agenda



Reuse

Exploit reuse in all phases to leverage pre-existing artifacts, requiring less original development

Abstraction

Exploit higher-level abstractions at each phase to reduce details and focus on separation of concerns

Turns per day

Significantly increase “turns per day” to reduce time and effort to iterate in development flow

Formulation

(strategic design, prediction, tradeoff analyses)

Design

(tactical design, implementation)

Translation

(compilation, technology mapping, linkage)

Execution

(debug, verification, optimization, run-time services)

Integration

Integrated end-to-end development environment improves flow, automation, reuse, and turns per day; maximizes productivity

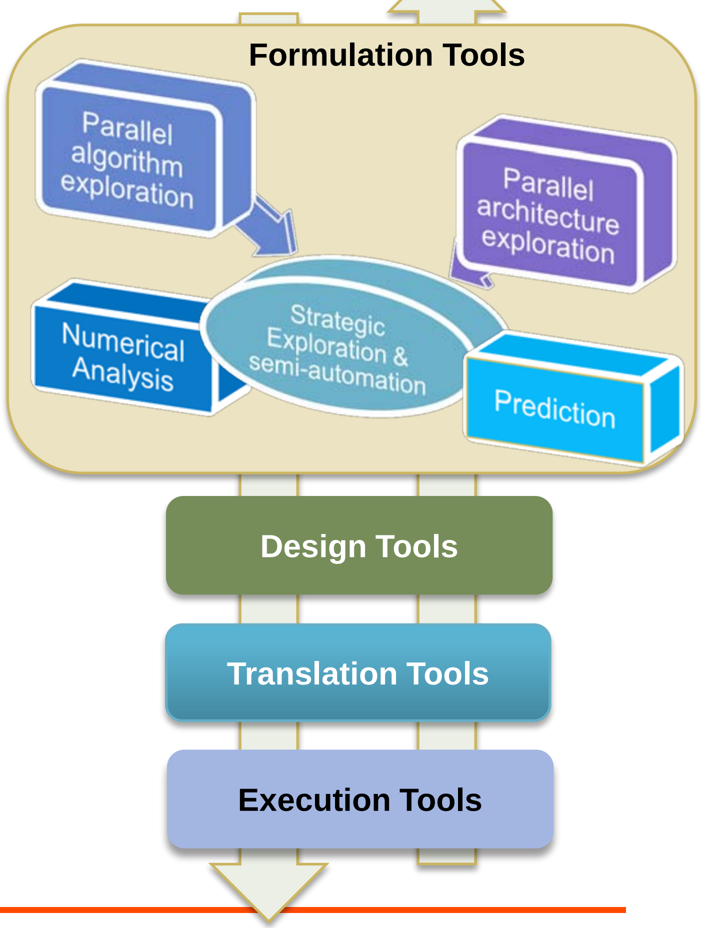
Projected productivity impact on order of 20×

I. Formulation: Abstract Modeling & Exploration (1)

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

Research Thrusts

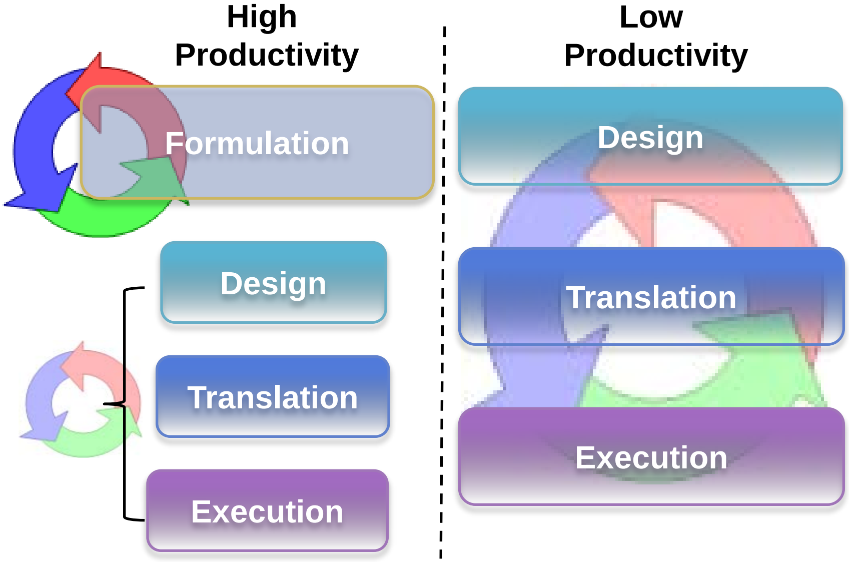
- Abstract app modeling for strategic exploration
  - ***Not*** coding in traditional sense
  - Abstraction layer above Design amenable to domain scientists
- Parallel algorithm exploration
  - Application modeling, expressing deep & wide parallelism
  - Graphical, textual, or hybrid modeling techniques
  - Pattern-based exploration for alg. modeling & mappings
- Parallel architecture exploration
  - Abstract platform models for target explorations & mappings
    - Arch. resources, I/O bandwidths, alg. mapping options
  - Iterative algorithm / architectural exploration for viable combinations, tradeoffs
- High-level performance and resource prediction
  - Supports tradeoff analysis (alg., arch., both)
  - Memory hierarchy, data locality, bottlenecks
  - Analytical, simulative, or combo
- Resurgence of concepts in numerical analysis
  - Accuracy vs. resource tradeoffs
  - Concepts, methods, & tools to aid arithmetic precision analysis
- Feeder to Design stage
  - Must bridge into Design, or else users will not adopt
  - Patterns, templates, code generation, libraries


Formulation Tools
Parallel algorithm exploration
Parallel architecture exploration
Numerical Analysis
Strategic Exploration & semi-automation
Prediction
Design Tools
Translation Tools
Execution Tools

I. Formulation: Abstract Modeling & Exploration (2)

Qualitative Impact of Formulation Innovations

- 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 reductions in DTE costs
  - Reduction in Design time
    - Transitions from F to D (automation, patterns, templates, code)
  - Reduction in DTE frequency
    - Better strategic choices in Formulation mean less design & re-design
- Significant utility gain for non-experts
  - Methodologies amenable to domain scientists & system designers; shorter learning curve
- Maximum impact on overall productivity
- Formulation concepts & methods potentially applicable to heterogeneous, many-core world

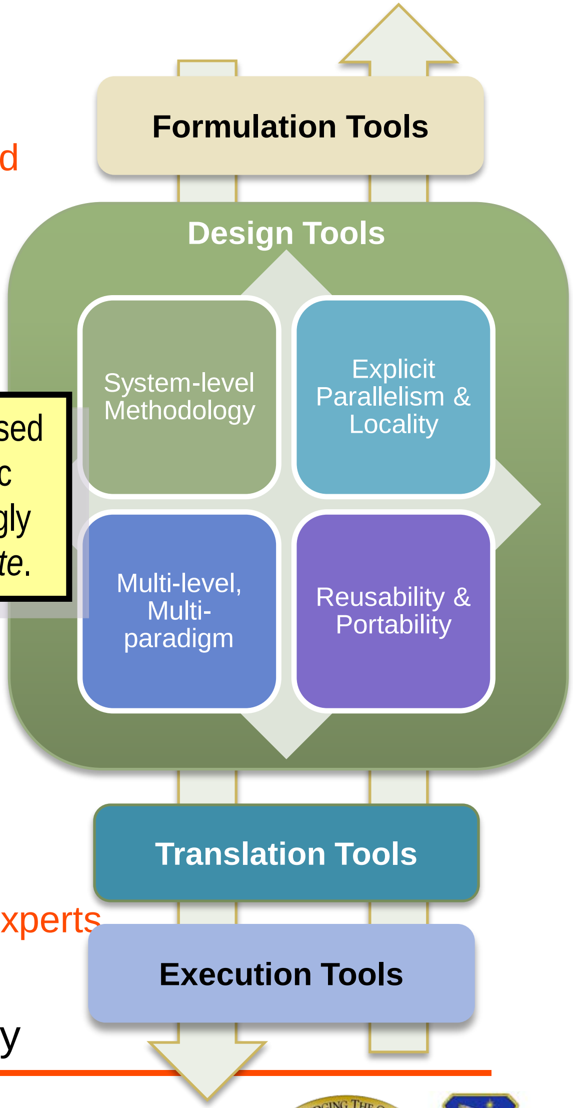


II. Design: Concepts, Languages, Patterns

Research Thrusts

- System-level design languages & methods
  - Constructs for explicit deep & wide parallelism, & locality
  - Integrated hardware-software co-design methodologies for automated mappings on target architecture
  - Multi-paradigm & multi-level design abstractions for efficiency & productivity tradeoffs
  - Incremental and iterative design refinements for debugging & optimizations
  - High-level synthesis to produce better inputs for PAR tools
- Reusability & portability
  - Pattern-based design methodologies & core libraries to maximize reusability & portability
  - Standard FPGA system & sub-system interfaces for better design portability

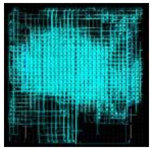
DTE phases traditionally used for "seat of pants" ad-hoc formulation, but increasingly *inefficient* and *inappropriate*.



Qualitative Impact of Design Innovations

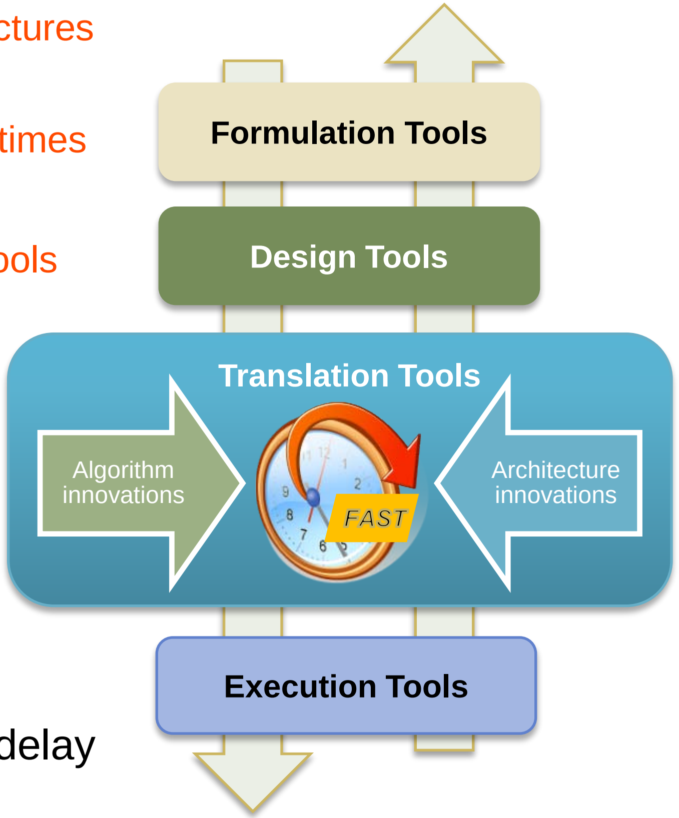
- Higher user productivity due to intuitive development methods
  - User-friendly high-level languages to reduce learning curve for non-experts
- Considerable reductions in Design time & DTE frequency
- Reusable and portable designs significantly improve productivity

III. Translation: Algs & Target Architectures



Research Thrusts

- Translation algorithm innovations
  - Parallel PAR tools to expedite Translation using parallel architectures
  - Region-based PAR tools to reduce routing complexity
  - Improved PAR techniques to tradeoff routing effort & execution times
- Device architecture innovations
  - CAD-amenable architectures to improve quality of Translation tools
  - Coarse-grained RC architectures as better Translation targets
  - Hierarchical routing with tool emphasis on locality & pipelining
- Architecture-aware Translation
  - Outside of scope for SIRCA (*see DARPA AACE program*)
  - Challenges abound for FMC devices, premature for RMC



Qualitative Impact of Translation Innovations

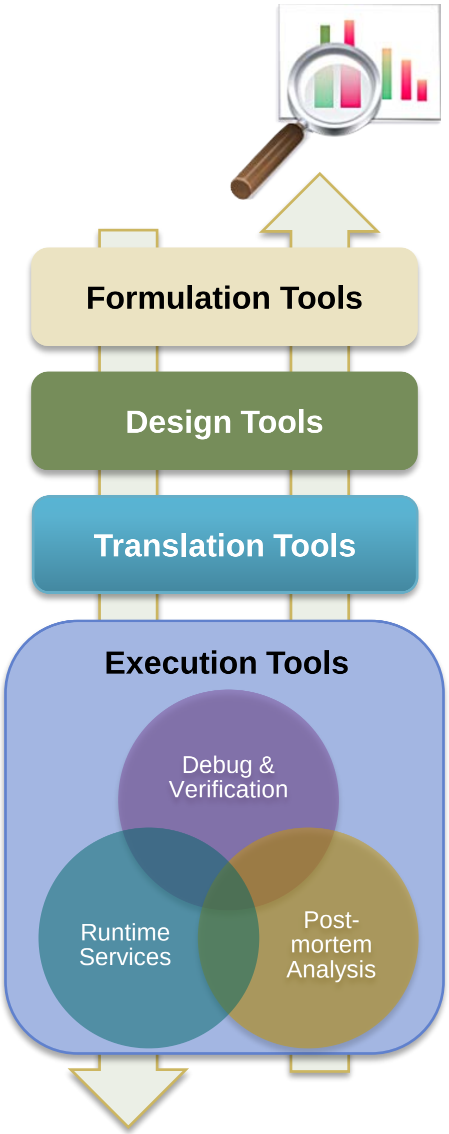
- Reduction in Translation times, but not frequency
- *Common misconception*: Translation cost is all from PAR delay
  - T_{time} reduction beneficial, but often overlooked yet equal is T_{freq} (development cost of T stage = $C_T = T_{\mathrm{time}} \quad T_{\mathrm{freq}}$)
  - T_{freq} reduction achieved via innovations in **F**, **D**, and **E** phases

Note: emphasis here should be upon revolutionary breakthroughs, not incremental vendor advances in PAR

IV. Execution: Runtime Analysis & Services

Research Thrusts

- Runtime debug & verification
  - Runtime, system-level, in-circuit debug methods & tools for design validation
- Post-mortem analysis & optimization
  - Runtime, post-mortem analysis methods & tools for critical optimization
  - Feedback to Formulation, Design, & Translation phases for analyzing bottlenecks
- Runtime services
  - Services to support varied mission scenarios such as FT, RT, RTR, PR, SoC; e.g. checkpoint & heartbeat services, PR configuration manager
  - Load balancing & job scheduling for multi-node systems
  - Secure configuration mechanisms for sensitive missions



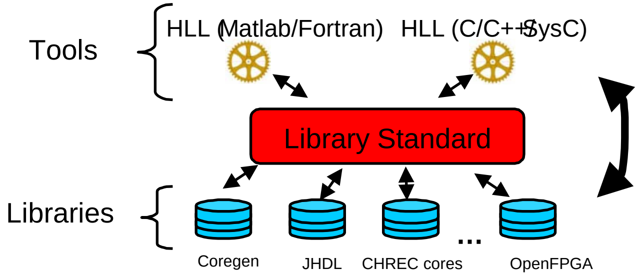
Qualitative Impact of Execution Innovations

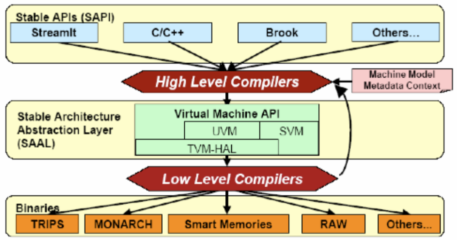
- Superior Execution tools can provide significant development cost savings, reduction of debug time & DTE frequency
- Optimization & bottleneck elimination can also result in critical utility gains, thus increasing productivity

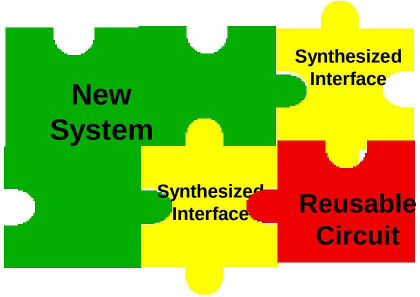
Reuse

Substantially increase the amount of design reuse at all levels of the design flow

- Library reuse standards
  - Establish standards for cores
  - Create reusable core libraries
- Dual Layer Compilation
  - Compile to common machine model
  - Architecture specific compilation
- Interface Synthesis
  - Simplify integration of cores into system
  - Abstract details of core from designer



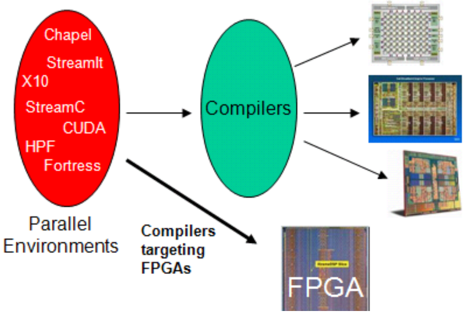


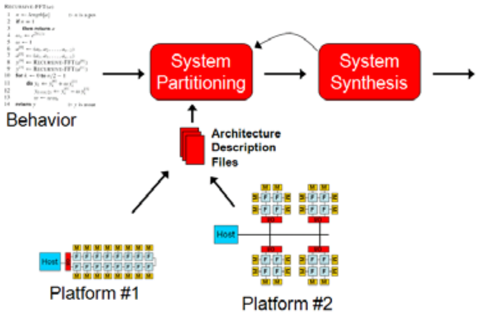


Abstraction

Reduce the amount of detail required to specify computations by raising design abstractions

- Leverage emerging concurrent models of computation
  - Multi-core programming models
  - MPP programming models
- Remove circuit-level details
  - Programmers do not need clocks
  - Automate resource allocation
- Support multi-FPGA synthesis
  - Automatic partitioning

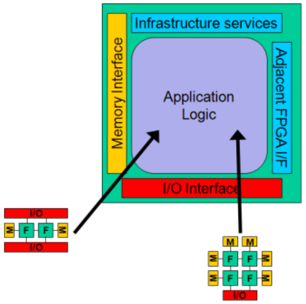


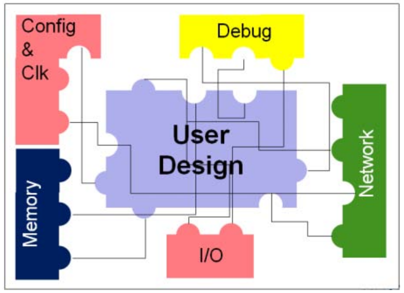


Turns-per-Day

Increase the ease of design debug and deployment by providing many more “turns per day”

- Platform services
  - Debug and monitoring support
  - Insert debug circuits
- Firmware
  - Support for standard services
  - Reuse placement & routing
- High-level abstraction debug
  - Debug HLL in hardware





Potential Impact: ~25x

Abstraction: 2x DP improvement
Increase CC/ILOC by 2x

$$DesignProductivity = \frac{CC \times TPD}{ILOC_0 \times [(1-R)+(O\times R)] \times \frac{Turns}{ILOC}}$$

Reuse: 4x DP improvement
(R=0.8, O=0.10)

Verification: 3x DP improvement
Increase TPD by 50%
Decrease Turns by 2x

Impact Summary

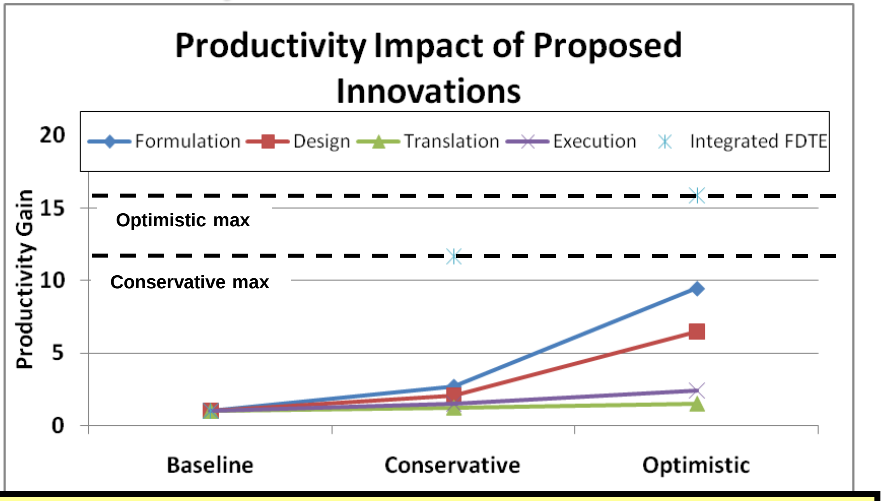
$\psi = \frac{U}{C}$

Innovations ranked based upon results of projected impact analysis

1. Formulation innovations have highest potential for impact on productivity
   - Potential to dramatically increase user base for FPGAs
   - Strategic exploration concepts central to RC, but also potentially applicable to FMC world
2. Design innovations without Formulation innovations do increase overall productivity
   - Formulation + Design innovations likely to generate maximum impact
3. Execution innovations important for providing critical runtime information
   - Reduction in optimization, debug, & verification costs
4. Translation time improvements beneficial, but less impactful on overall productivity model
   - But, intangible benefits may also result (human factors)

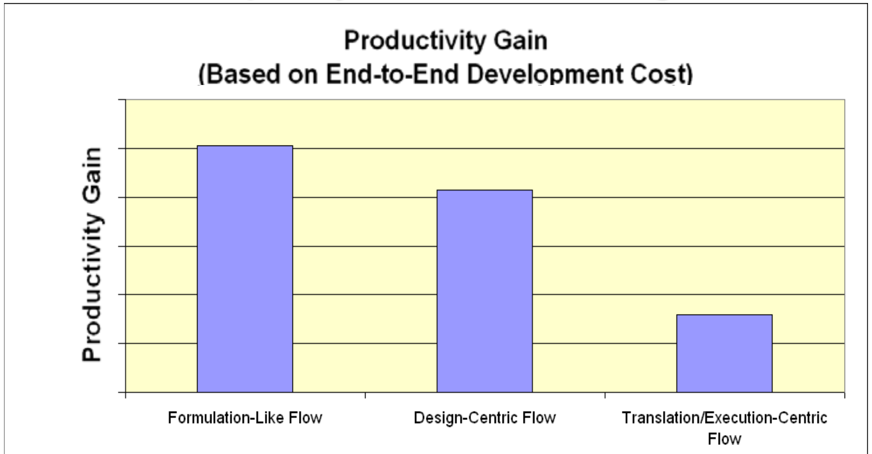
- Cumulative impact of integrated FDTE methodology on productivity
  - Theoretical max prod: As DTE → 0, total cost approaches learning curve + Formulation cost
    - Conservative: 12× productivity improvement
    - Optimistic: 16× productivity improvement

Projections from FDTE Innovations



**** Gains even more pronounced when utility (*U*) from RC is included ****

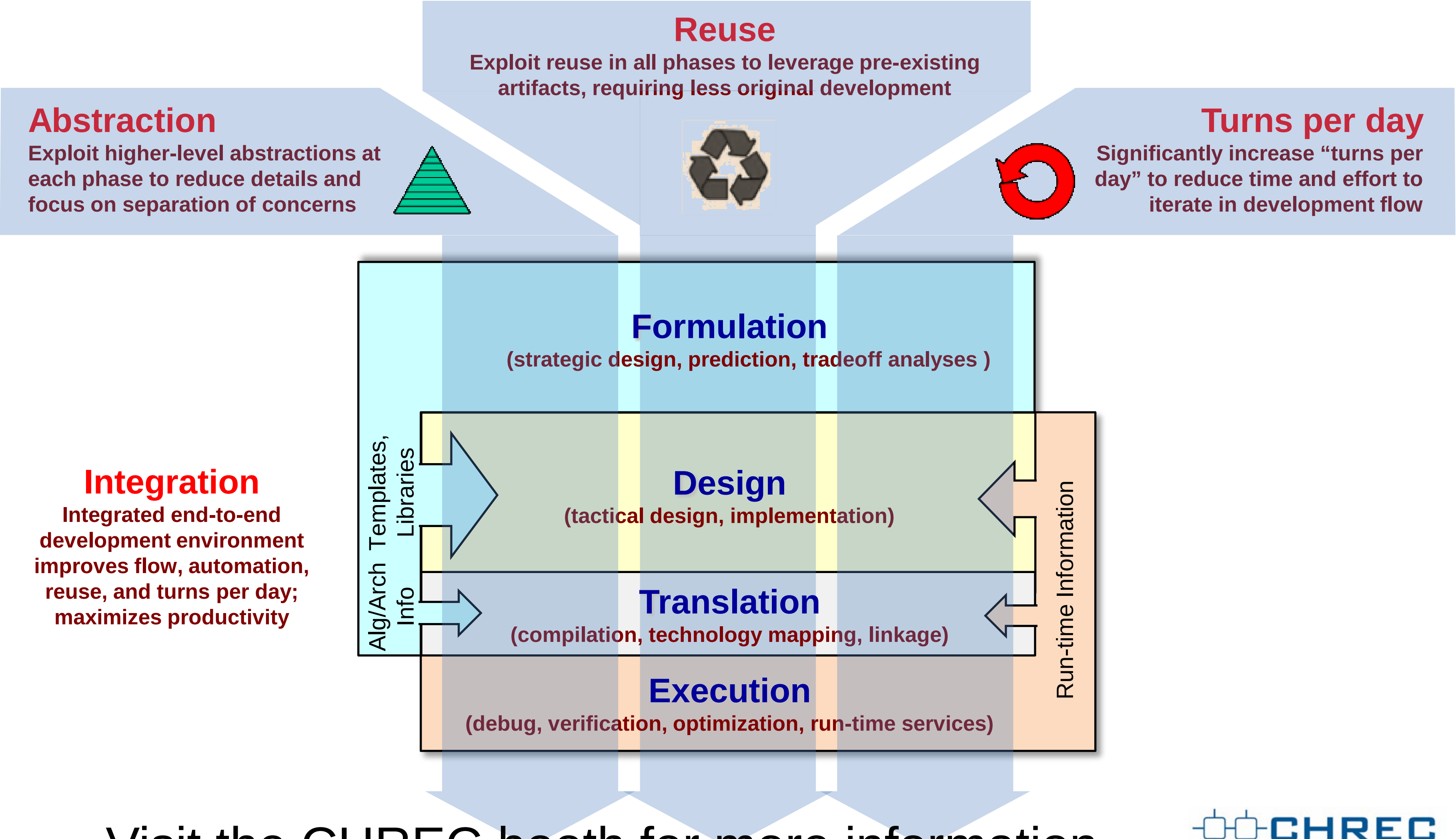
Recap: Projections from Existing Tools



Results consistent between both impact analyses

**** Productivity is unlike speedup; 12× reduction in cost (*C*) would be truly outstanding (e.g. man-year effort reduced to man-month) ****

Questions?



Reuse

Exploit reuse in all phases to leverage pre-existing artifacts, requiring less original development

Abstraction

Exploit higher-level abstractions at each phase to reduce details and focus on separation of concerns

Turns per day

Significantly increase “turns per day” to reduce time and effort to iterate in development flow

Formulation

(strategic design, prediction, tradeoff analyses)

Design

(tactical design, implementation)

Translation

(compilation, technology mapping, linkage)

Execution

(debug, verification, optimization, run-time services)

Alg/Arch Templates, Libraries

Info

Run-time Information

Integration

Integrated end-to-end development environment improves flow, automation, reuse, and turns per day; maximizes productivity

Visit the CHREC booth for more information

CHREC
NSF Center for High-Performance Reconfigurable Computing