

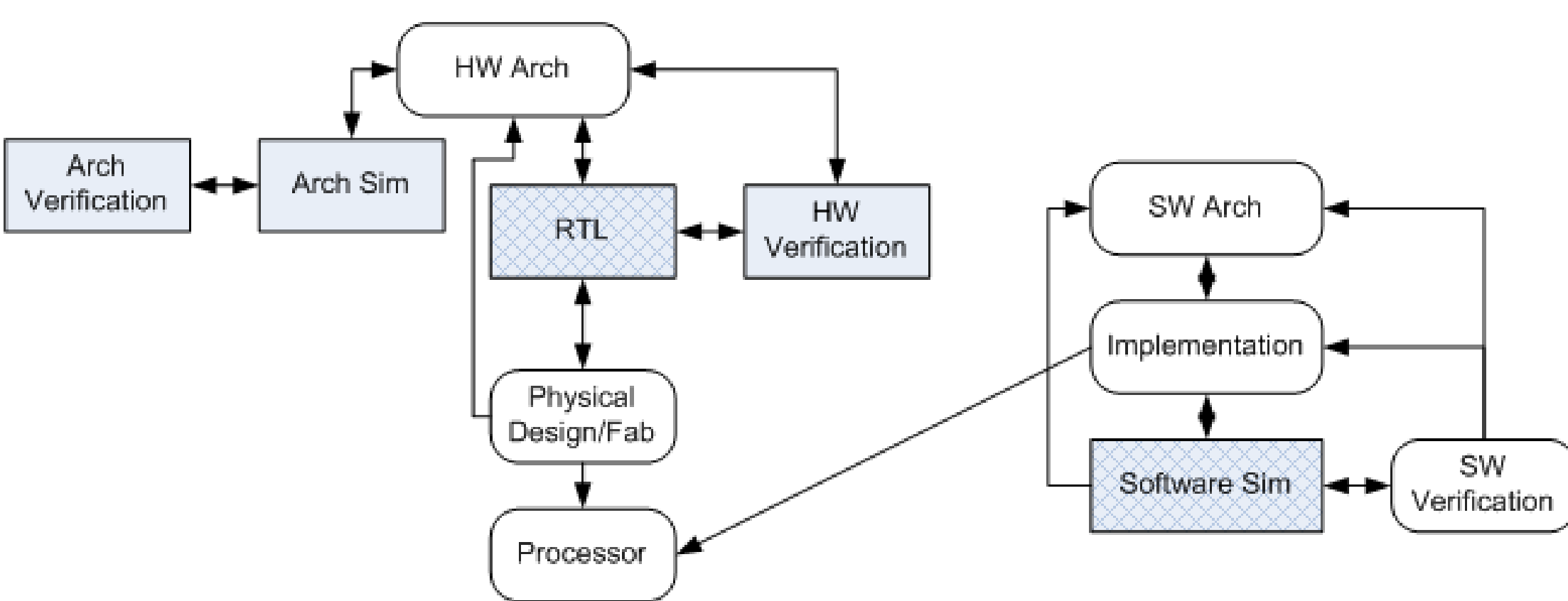
# Transforming Simulators into Implementations

Nikhil A. Patil and Derek Chiou

UTFAST Research Group, Electrical and Computer Engineering, University of Texas at Austin

## How Computers are Designed Today

Designing a next-generation computer system is a complex endeavor involving a huge investment of human and monetary resources. The development begins with architectural exploration, in which computer architects write performance models for the system they envision. Design decisions are made after extensive simulation, and the architecture is passed along to the RTL and Verification teams in the form of an extremely detailed document. Although the architectural simulator contains most of the information needed by the teams downstream, they *cannot* use the simulator to aid their development effort.



For a variety of reasons including bug-fixes, oversights, incorrect architectural assumptions and implementation constraints, as the project progresses, the gap between the proposed architecture and the actual implementation gets wider and wider. While it would be useful to re-simulate the newer design, it is rarely done to as much detail as earlier architectural simulations.

Since very little software development occurs before the product is made available in the market; all the new features introduced by the chip are essentially unused until software is developed.

## FAST simulation methodology

FAST simulators are a novel simulation technology that can produce full-system simulators that are simultaneously fast and accurate, while providing a high-level of visibility. FAST simulators are factored into a functional and timing model.

Our current prototype implements the functional model (FM) in software and the timing model (TM) on FPGA hardware. The functional model functionally executes x86 instructions, generates a trace (dynamically) and passes it on to the timing model. When the FM strays from the target path (e.g., due to a branch misprediction), the TM sends a command to the FM requesting it to roll-back its execution and send the “wrong-path” trace.

## Simulation/Implementation Duality

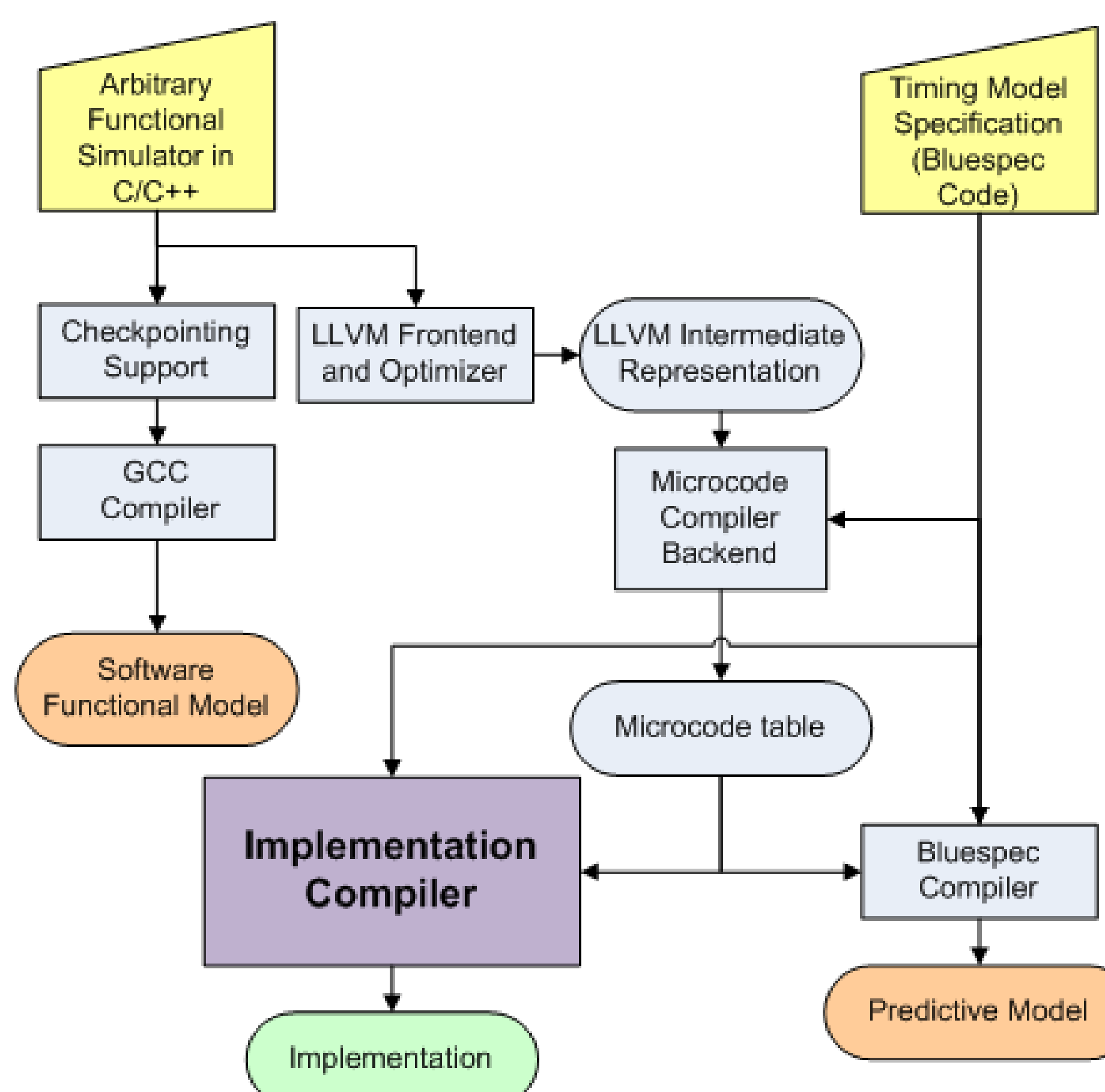
A FAST simulator could be made accurate enough to the extent that it contains all the information needed to create an implementation. This information is encoded in synthesizable HDL (FAST timing models are currently written in Bluespec SystemVerilog) and is therefore a good candidate for an automatic transform into an implementation.

To simplify targeting an FPGA, every module in the timing model is allowed to take multiple *host cycles* (FPGA cycles) to simulate a single *target cycle*. This enables FAST to model complex structures like 18-ported register files and timing critical structures like reservation station structures with relative ease.

Of course, a full implementation cannot take multiple host cycles for one target cycle. This would effectively mean that the generated implementation is several times slower than the intention. However, it will still be *accurate* – just with the speed scaled down.

We make the claim that while this implementation is not ideal, it is still suitable for several purposes downstream. We call such an automatically generated implementation, an *Imp*.

The *Imp* can be incrementally developed into the full implementation, by replacing every multi-cycle module with one that implements one target cycle in one host cycle.



## Applications of the technology

There are significant advantages of being able to transform a simulator into an implementation:

1. *Imp* can potentially be made available to software developers as an early release of an evenly scaled slower implementation, giving them time to take advantage of the newer features of the processor before the market release and provide valuable feedback to the architecture/hardware teams
2. *Imp* can be an effective golden model for the RTL and verification/testing teams
3. *Imp* can be verified using traditional methods, increasing confidence in the FAST simulator from which it was derived
4. *Imp* can serve as a fully functional starting point for RTL
5. FAST + *Imp* could encourage more robust implementation practices and a faster time to market

## Enabling this Technology

Our vision of how FAST *Imp* integrates with the FAST simulator allows us to influence core simulator design decisions:

1. Extremely clean interfaces between properties of target models and simulation structures allows the automatic tool to mechanically strip away simulation structures.
2. The FAST connector (the entity that connects multi-cycle modules) needs to be able to support models that take exactly one host cycle per target cycle.
3. Data-path placeholders: In theory, timing models do not require the presence of data-paths, since they do not affect timing. This saves area on the FPGA. We implement “data-path placeholders” using advanced abilities of the Bluespec polymorphic type system, which let us make the data-paths either 0-bit wide or 1-word wide with a single line of code.
4. FAST simulators don’t *need* to implement the ISA decoder in the timing model, since the functional model decodes the instruction anyway. However, to make *Imp* possible, our prototype embeds a microcode decode table automatically generated from the functional model in the timing model.

## References:

1. D. Chiou, D. Sunwoo, J. Kim, N. A. Patil, W. H. Reinhart, D. E. Johnson, J. Keefe, and H. Angepat. “FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators. In Proceedings of MICRO, Dec. 2007.