Hybrid Computer Architectures

By Dr. Philip Sweany, Dr. Hao Li, and Cameron Palmer

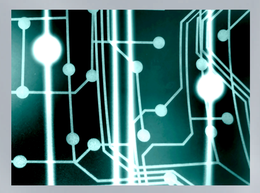
Motivations for Research

What Is a Hybrid Processor?

Hybrid processors combine two traditionally separate types of computational devices on a single chip. Current commercial hybrid chips provide fixed processing cores and Field Programmable Gate Array (FPGA) elements to permit customization. By including an FPGA with a fixed processor, you obtain a performance and power benefit over that which is available with generic processors, while retaining more flexibility than an Application Specific Integrated Service (ASIC) can provide.

How Do We Use Them?

Hybrid chips seem uniquely qualified to allow designers to optimize and configure the available resources for their applications. However, currently available tools require substantial knowledge of hardware and significant human interaction to obtain the performance and power improvements.



To realize the potential of hybrid computing systems, we need automated tools to map application programs to existing hardware. While the goal of hardware and software co-design is to design hybrid hardware specifically optimized for a single application, our goal is to map applications written in a procedural programming language to a hybrid architecture that has already been specified.

Using the fully automated tool Hy-C, we will evaluate the potential of hybrid computing to execute a wide range of programs that are efficient both in execution time and power consumption.

How Do We Evaluate The Hybrid's Potential?

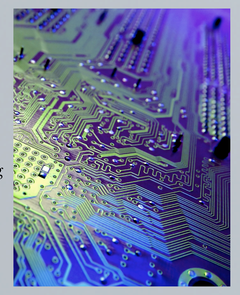
To evaluate our hybrid computing model, we need a tool to iteratively partition, analyze, and repartition input source code to execute in either FPGA or CPU. Since we wish to evaluate potential of different hybrid architectures, it is insufficient to build tools that are applicable to only a single architecture. Therefore, evaluation of hybrid computing potential requires the ability to easily target our code generation tools to a wide range of multiprocessor and multi-FPGA architectures to allow for experimentation.

The Hy-C tools will map the C language to a given architecture. Thus, Hy-C will provide the infrastructure necessary to evaluate the potential of hybrid computing.

The tools will facilitate research in hybrid computing. Our Hy-C infrastructure design relies on three principles:

- Re-use available research tools.
- Ease of use.
- Inserting new tools and modifying existing tools should be simple.

Design using these principles facilitates system-level and tool-level research within Hy-C.



The Hybrid Computing Model

Figure 1 illustrates our schematic view of a generic hybrid architecture. Current hybrid chips provide fixed processing cores and reconfigurable resources (represented as FPGAs in Figure 1), permitting customization.

Reconfigurable resources could just as easily be ASICs or some combination of ASIC and FPGA resources. By including an FPGA with a fixed processor, we achieve performance and power benefits while retaining more flexibility than an ASIC.

The biggest obstacle of a hybrid approach is the difficulty of mapping high-level programs to efficient machine-level structures to take full advantage of provided hardware. Currently available mapping tools are limited in their generality and scope. A major goal of our research is to provide this flexible and general mapping.

Figure 2. provides a high-level view of how our tools will generate code for hybrid architectures, based upon the generic target system architecture of Figure 1.

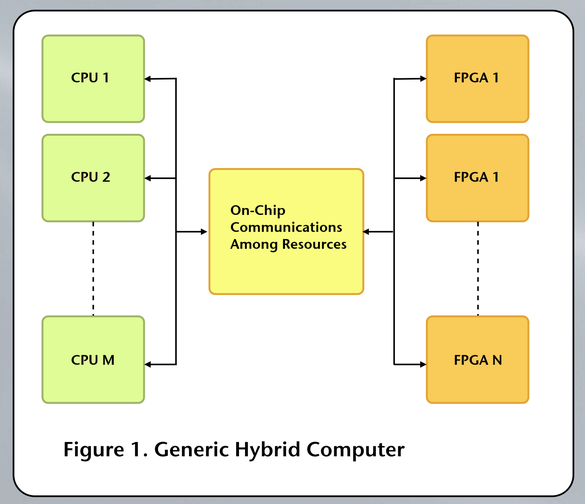


Figure 1. Generic Hybrid Computer

What Tools Generate the Hybrid Architecture?

As shown in Figure 2, our code-generation tools can be divided into six basic parts: a specification language, a partitioning compiler to assign computation to either

- fixed-CPU or FPGA elements
- fixed-CPU C compilers
- high-level synthesis tools
- SPARK
- maps part of the computation to be done in reconfigurable logic into VHDL
- a form that can allow for automatic generation of FPGA code
- an FPGA compiler that takes VHDL specifications of computation and generates the FPGA "program"
- performance and power estimation tools.

Our code-generation tools will make significant use of available compiler, CAD, and power-estimation tools. We plan to build only those tools necessary to fill in the gaps in current hardware and software co-design, shown in yellow in Figure 2. Those parts of Hy-C that will require significant new implementation are shown in green.

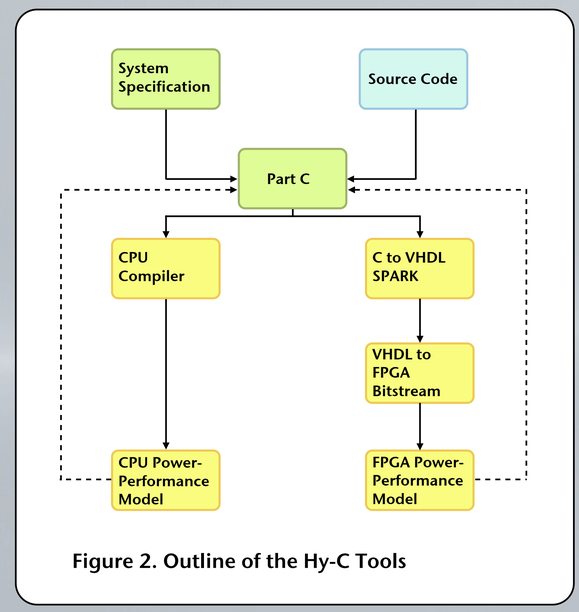


Figure 2. Outline of the Hy-C Tools

What Do the Tools Do?

Given a system specification and application source code, our tools first partition work among the processing resources. With successful partitioning, separate translation tools for both the general-purpose CPU, high-level synthesis transformations, and FPGA components generate a machine-level program for the specific computational resources.

At this point, each processing resource access performance power models to allow an accurate estimate of the performance and power characteristics of the runtime program. We can then feed this information back to the partitioning phase to consider a possible repartitioning of the computation. By supporting an iterative partitioning process, we can refine code generation for the hybrid system, selecting that partitioning which best meets the optimization requirements of execution performance and power constraints.