

Reconfiguring the future of computing

Benjamin Gerdemann

May 15, 2008

Reconfigurable computing is a new computing paradigm that bridges the gap between software and hardware, combining the high performance of hardware with the flexibility of software.

Since the beginning of computing there have been two primary methods of execution. The first is a hardware method that is “hardwired” for a specific algorithm and thus high performing and efficient when executing this exact computation. However, once such a circuit has been designed and manufactured any change requires a re-design and re-manufacture. A chip designed for a television or cell phone is an example of this technique. These specialized chips are optimized for their specific task and, for example, a new cellular communication protocol or video format would require a re-design and re-manufacture of the hardware.

The second method of computation is a general purpose microprocessor, such as a CPU found in a personal computer, that can run generic software instructions. This method is flexible enough to execute almost any arbitrary sequence of operations without the expensive re-design that a hardware circuit would require, but performance may suffer because of the overhead of

a generic design. Additionally, all functionality must be constructed from a base set of simple instructions.

Reconfigurable computing is a new computing paradigm that increases performance by performing computations in hardware while retaining much of the flexibility of software. This method also has the potential to bridge the computing gap between hardware and software. Reconfigurable hardware devices can be programmed into custom digital circuits which can then be interfaced to a general-purpose microprocessor. The microprocessor executes the generic software routines while the reconfigurable hardware accelerates the portions of the computation that can be done efficiently in hardware. Computations that have shown success with this approach include data encryption, string pattern matching, genetic algorithms, bioinformatics, remote sensing and molecular dynamics. [1] In addition to improving performance, the hardware can later be modified to introduce new functions and fix bugs which increases the longevity and versatility of the design.

An important characteristic of reconfigurable hardware is its level of granularity. Granularity is defined as the size of the smallest function the hardware can execute. Coarse-grained devices include complex functional units such as adders or shifters that can be connected by designer. The pre-designed circuits can be highly optimized, but also limit the design to the provided functions and can lead to wasted hardware. For example, if the smallest provided adder is 8-bits, a single-bit addition would waste seven bits. Unlike coarse-grained devices, fine-grained hardware can implement any arbitrary function down to the bit-level, but with this level of configurability comes increased power consumption and area usage due to the additional

connections required between the functional units. One solution to this problem is to combine coarse grained and fine grained arrays on the same chip. The most common and commercially successful reconfigurable hardware are field-programmable gate arrays (FPGAs) which are typically fine-grained.

The idea of reconfigurable computing has been around since the 1960s when Gerald Estrin proposed a computer with a standard processor attached to an array of “reconfigurable hardware.” [3] He imagined the processor would control the reconfigurable array which would be customized to perform a certain computation. Unfortunately, the hardware of the time was not sufficiently advanced to truly implement the hybrid computer that Estrin imagined.

The 1980s and 1990s saw a regrowth in the area of reconfigurable computing as the speed, size and capability of FPGAs increased and manufacturing technologies improved. The size of FPGAs has been increasing so rapidly that it is already possible to implement a complete microprocessor design on some FPGAs.

The first commercial reconfigurable computer, the Algotronix CHS2X4, was developed in 1991 and now several high-performance computers from Cray, SGI and SRC Computer have incorporated reconfigurable components in their architecture. They have FPGAs that are used as coprocessors which can be reconfigured by the microprocessors for specific hardware implementations. Depending on the application and configuration of the FPGAs, speedups from 20 to over 12,000 over a microprocessor only design have been demonstrated. [2]

Additionally, some consumer products have already been released with

reconfigurable architectures. Blaupunkt has developed an auto navigation system using FPGAs and the Sony Playstation Portable (PSP) has a reconfigurable architecture that dynamically switches between a general purpose CPU and “virtual mobile engine” to save power during audio and video decoding.

As these examples show, using a reconfigurable architecture can bring several advantages. First, computation performance can be improved by executing critical parts of a program in custom hardware. Custom hardware can be optimized to efficiently compute a calculation that might take many steps to perform in software code. For example, many cryptographic functions are expensive to compute in software, but can be directly calculated in a hardware circuit. In some algorithms, performance can also be improved by utilizing hardware to calculate in parallel what could only be done sequentially in software.

Additionally, a custom circuit can be more power and size efficient than than a generic processor, because of the designer’s ability to optimize a circuit including only the minimum necessary components with less overhead. In a portable media player, for example, a generic microprocessor could be used for the user interface, but when a audio or video stream is being played for a long period of time, an optimized reconfigurable chip could be used to save power. Different circuits could even be used depending on the type of media, for example audio versus video, or even the type of encoding; a low-bandwidth audio book could decoded more efficiently than a high-bandwidth pop song.

Finally, being able to reconfigure hardware expands the flexibility and longevity of a design. In a fixed hardware architecture, the ability to fix

bugs or add new functionality is very limited or even impossible, because the size and features of the design are effectively fixed when it is manufactured. However, a reconfigurable architecture could, like software, be extended with new functionality or have bugs fixed simply by updating the hardware. A cell phone with a reconfigurable architecture, could be reprogrammed with a new network protocol simply by automatically downloading a new configuration. Because the hardware is almost infinitely configurable, there would be virtually no limit to its capabilities.

Reconfigurable computing has the potential to revolutionize computing by increasing performance and efficiency of applications and extending the flexibility of hardware, but the technology is still in its infancy. Much research needs to be done to identify techniques and tools to enable easier and higher performing implementations of reconfigurable architectures.

For example, extracting a portion of a software code to be executed in hardware is still difficult. There exist some software compilers that attempt to do this automatically, but their accuracy and performance is typically much slower than one done by hand. Unfortunately, a hand-made design is typically tedious and error prone and requires a designer with intimate knowledge of both hardware and software development techniques.

Additionally, unlike the microprocessor world, where a few well-defined instruction set architectures exist and compatibility between processors is simple, there doesn't exist yet a similar standardization of reconfigurable components and interfaces and there is no "typical" reconfigurable architecture. Additionally, each vendor has their own hardware design and tool flow that is normally incompatible with others. An engineer of a reconfigurable design

is faced with a overwhelming myriad of choices when beginning a project.

At IST/INESC-ID there is a research team (ANCORA) focusing on compiler techniques and architectural schemes that can be used to bridge the gap between software programming languages and reconfigurable computing architectures. The main objectives of the team's research are to identify a basic analysis and compilation approach that can effectively exploit the potential of reconfiguration in current and emerging execution models and validate the approaches by developing early prototype implementation of tools. For validating the research applications from mobile robotics are used as they exhibit most embedded system requirements and expose many challenges not being currently addressed.

Reconfigurable computing has the potential to dramatically change how future computing systems are developed by providing an attractive compromise between the performance of hardware and the versatile programmability of software. There is a great deal of research interest in this area around the world and here at IST. Techniques being developed in academia now may soon allow this revolutionary technology to reach it's full potential.

References

- [1] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Computer Survey*, 34(2):171–210, 2002.
- [2] Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, and Duncan Buell. The promise of

high-performance reconfigurable computing. *Computer*, 41(2):69–76, 2008.

- [3] G. Estrin and C. R. Viswanathan. Organization of a “fixed-plus-variable” structure computer for computation of eigenvalues and eigenvectors of real symmetric matrices. *J. ACM*, 9(1):41–60, 1962.