I have pursued three primary research thrusts: reconfigurable computing, design of parallel architectures, and media processing.

1 Reconfigurable Computing

One major focus of my research activities has been reconfigurable computing. In particular, I have concentrated on the engineering and application of computing systems comprising field programmable gate arrays (FPGA) and similar devices. In these devices, computing structures are organized by spatially wiring a fabric of simple logic units (similar to logic gates), as opposed to serially executing instructions, as with conventional processors. Field Programmable Gate Arrays (FPGAs) and other reconfigurable devices provide the flexibility of software processors with their ability to be customized or specialized on a per application basis, along with the power, performance, and cost efficiency approaching that of fixed function application specific integrated circuits (ASICs). It has long been understood that reconfigurable devices can provide significant advantages over conventional processors; and because of their fine-grain parallelism and flexibility, they can be adapted over a wide range of applications and scale over a variety of problem sizes.

The primary challenge in reconfigurable computing is ease of use. Past success has only come after custom development of reconfigurable computing platforms followed by laborious hand-mapping and tuning of applications. The approach to mapping an application for execution on a reconfigurable device has been equivalent to designing a low-level logic level circuit. My research has focused on experimentation with techniques for improving the ease of use of reconfigurable devices for a variety of computing tasks. In particular we have developed 1) application domain use models and abstractions for reconfigurable computing, 2) programming models and convenient programming tools, 3) novel reconfigurable chip architectures, and 3) hardware platforms with associated system level support (similar to operating systems). This research has been driven by a few major application areas, including tasks in computer architecture emulation and simulation and a set of compute intensive tasks drawn from signal processing in radio astronomy and data-processing in systems biology research. Furthermore, the platforms and tools we have developed for our own research have been exported and adopted worldwide in a wide variety of research areas.

1.1 Programming Models and Tools for Reconfigurable Computing

We have taken several approaches to developing easy-to-use programming models and tools for reconfigurable platforms.
In early work, we invented a programing model based on Kahn-process-networks and used it as the basis of a complete execution framework, including a custom chip architecture with rapid run-time reconfiguration. Several journal publications representing this work appeared:


The second paper describes a novel technique using reconfigurable devices themselves to accelerate the routing phase during the mapping of applications. This is a significant milestone, as it is one of a few instances of using reconfigurable devices to run their own development tools. We were granted a patent on this technique:


We also pioneered another approach, which became particularly successful, based on automatic translation of Mathlab/Simulink block diagrams into programming information for FPGAs. This work was originally in collaboration with Mathworks and Xilinx, and formed the basis of the commercially successful “system generator” tools from Xilinx and Mathworks. We have used this approach in a variety of applications at BWRC (Berkeley Wireless Research Center); for instance in specification and validation of novel radio systems and coding algorithms, and with signal processing for radio astronomy. This approach is summarized in the following book chapter:


As tool-result quality is a key factor in the ease of use of reconfigurable devices, we have continued to work on techniques to reduce the runtime and improve the results for conventional FPGA design tools. In the following paper we show how to improve the very challenging problem of circuit mapping and placement on FPGAs:

1.2 FPGA Overlay Architectures

Despite much work on novel programming models and tools by us and others, a huge gap remains in the programming experience on reconfigurable devices versus software processors. Among other problems, FPGA mapping tools are slow, with designs mapping taking hours and sometimes days. Inspired by our earlier work on mapping manycore architectures to FPGAs for emulation, we postulated that a manycore architecture could form the basis of an abstraction layer or execution model for FPGA-based computing. By adapting this abstraction, we could enable the use of parallel programming languages such as openCL and CUDA. Also, because with this abstraction layer the structure of the FPGA configuration could stay relatively fixed, and therefore the entire compilation process could be sped up by orders of magnitude.

In a series of papers, we pioneered a new methodology to design FPGA-based computing systems by combining a many-core architectural template, a high-level imperative programming model, and modern compiler technology to efficiently target FPGAs for general-purpose, compute-intensive applications. On significant applications, we demonstrated that performance competitive with hand optimized FPGA solutions is attainable with days versus months of development time.


The idea has since entered the mainstream, with both major FPGA vendors, Xilinx and Altera, offering OpenCL support for their devices.
1.3 Engineering Novel FPGAs

Another approach to advancing the effectiveness and ease of use of reconfigurable computing is to re-engineer the reconfigurable devices themselves. Commercial FPGA have evolved away from optimality because of the constraints of legacy tools and backward compatibility. Through the years my group has generated a series of novel reconfigurable devices and demonstrated improvements over commercial designs, going back to:


This paper was recently honored at the 20th anniversary of the International IEEE Symposium on Field-Programmable Custom Computing Machines (April 2013), and in a special volume of the most significant papers from the conferences. Honored were 25 papers across all years and major FCCM topics that best exemplify the contributions from the conference. Compared to 500+ papers that have appeared in the conference, the honored papers represent roughly 5% of all papers published in the conference to date. The following paper is a more recent example showing how to improve conventional FPGA internal routing architectures.


And this one on adding robustness to FPGA devices.


Lately I have been advancing the idea that the academic community should produce an open-source hardware design to enable research into novel future reconfigurable architectures. Commercial designs are proprietary, undocumented, and overly complex. I originally made my appeal to the community in my keynote address at ReConFig 2010 (International Conference on Reconfigurable Computing and FPGAs, 13-15 December 2010). Then followed up by organizing the evening panel at FPGA 2011 entitled “Should the academic community launch an open-source FPGA device and tools effort?” (International Nineteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays 27 Feb.-1 Mar, 2011). This is now the topic of an ongoing project in my group on developing an open-source FPGA hardware design supported by open tools and releasing it to the community.
1.4 Development of Reconfigurable Computing Platforms

During the period of this review, we designed a unique computing platform, called the Berkeley Emulation Engine 2 (BEE2), based on state-of-the-art FPGAs. BEE2 was designed to be scalable from a single computing module with a around 1 Tera-operations/sec performance to full-rack sized systems with 40 Tera-operations/sec – competitive with modern supercomputers. We completed extensive work on low-level gateware (reusable design components mapped to logic gates) modules and software for enhanced usability, including communication modules, DRAM interfaces, and front-end controlling software. At the BWRC, we assembled several FPGA computing clusters, with the largest being a rack of 21 BEE2 models and demonstrated them on a variety of applications. We adopted a server-based cloud model and offered FPGA-based computing online.

While the system was originally designed with only a few BWRC applications in mind, it attracted the attention of researchers worldwide and became a standard research platform for projects related to high-end emulation and reconfigurable computing. During this period, to the extent possible, we made systems available to researchers worldwide through collaborative research agreements. We manufactured and shipped approximately 80 BEE2 systems (each comprising a single board with 4 FPGAs) to universities and research institutes throughout the USA, with a few systems going to Europe, South America, and Asia.

A few of the BEE2 adopters with which we have collaborated are:

- Princeton University: “Enabling Research in Concurrent Architectures and Runtime Validation through Hardware Emulation”, David August, Sharad Malik, Margaret Martonosi, Li-Shiuan Peh.
- University of Illinois at Urbana-Champaign: “IMPACT Multicore Prototyping Environment,” Wen-mei Hwu.
- The Pennsylvania State University: “Multi-Core Thermal and Reliability Emulation”, Mary Jane Irwin.
- University of Michigan: “FPGA-Based Resiliency Analysis”, Kypros Constantinides, Todd Austin, Valeria Bertacco, Scott Mahlke.
• CMU: “Development of special purpose speech recognition systems”, Rob Rutenbar.
• University of Toronto: “Solution to problems in molecular dynamics for computational biology”, Paul Chow.
• UC Berkeley, CMU, MIT, Stanford, UT Austin, University of Washington, UCSC, and Intel: “Research Accelerator for Multiple Processor (RAMP).”
• Stanford University Medical Center: “Automatic Determination of Cell Signaling Network”, Garry Nolan, Wing Wong, A. Teresa Ming.

1.5 BioInformatics Applications

The collaboration with Stanford lead to me being a co-PI on a large grant (4 years) from the NIH Cancer Research Institute. We used FPGA computing technology to speed up the calculations associated with determining protein signaling networks. We demonstrated speed-ups versus general purpose processors in excess of 2000 times. Our paper won the Best Student Paper Award at International Conference Supercomputing.


The success of the project with Stanford spurred the formation of the startup company, Bina Technologies, Inc. (www.binatechnologies.com), to provide high performance computing solutions to problems in genomics and bioinformatics. The company was incorporated in Spring 2011. I am a co-founder and technical advisor.

1.6 BEE3

Based on the success of BEE2 and the introduction of newer FPGA devices, in Fall of 2006 we began the design of BEE3. Microsoft Research (Chuck Thacker) had agree to donate engineering expertise and cover the costs of system prototyping. We proceeded to do the high-level design of the new system as a joint effort between UC Berkeley and Microsoft Research. Low-level system design and prototyping was carried out by Microsoft’s contract manufacturer. Besides the desire to take advantage of the advanced features of the newer FPGA devices, the design objective of the BEE3 was to make it more reliable and less expensive to manufacture. The intent was to design a system that would be appropriate for higher volume production. By Fall of 2007 we had a stable design and working prototype modules.

Our goal from the beginning of the BEE3 project was that a company would take over the manufacturing, distribution, and support of the BEE3. In summer of 2007, I co-founded BEEcube, Inc., a small company to commercialize the BEE3 system, (http://www.beecube.com). I continue to advise the company as a consultant. BEEcube is now profitable and ships BEE systems worldwide, with a focus on the
telecommunications market. As a result, in BWRC at UC Berkeley, we no longer have the hassle of managing the manufacturing, assembly, test, and shipping of BEE systems. This has allowed us to focus on development of novel high-productivity programming models and tools, and applications.

2 Design of Parallel Architectures

2.1 Computer System Modeling and Emulation

In Fall of 2005 I teamed with David Patterson and other members of the computer architecture community (five universities, plus Intel) to form a collaboration to tackle the challenges facing researchers in parallel system design. Processor architectures had crossed a critical threshold. Manufacturers had given up attempting to extract ever more performance from a single core and instead have turned to multi-core designs. While straightforward approaches to the architecture of multi-core processors are sufficient for small designs (2-4 cores), little was really known on how to build, program, or manage systems of 64 to 1024 processors. The computer architecture community lacked the basic infrastructure tools required to carry out this research. While simulation had been adequate for single-processor research, novel approaches are required for architecture research at the level of 64-1024 cores. Our approach was to apply FPGAs to the problem, by building architecture level prototypes of future parallel computing systems. We named the project “Research Accelerator for Multiple Processor (RAMP)”.

In Spring of 2006, at Berkeley we were awarded a 3-year NSF grant to develop RAMP. I was the PI of that grant and consequently assumed most of the leadership and management roles for the overall RAMP project collaboration. We developed a number of reference RAMP prototype systems. Each system was designed to exercise a particular approach to FPGA-based system emulation, and to provide models to serve as the starting point for an entire community of architecture researchers. Each reference design employed a complete gateware/firmware configuration of a scalable multiprocessor consisting of hardware components (such as processor cores, memories, and switches) and software components (such as operating systems). The designs were modular to allow quickly interchanging modules to modify reference designs or create new ones.

The overall approach and some early results appear in:


At Berkeley, we developed a system called “RAMP Blue” to emulate a family of message-passing machines that run parallel applications written for either the Message-Passing Interface (MPI) standard or for partitioned global address space languages such as Unified Parallel C (UPC). RAMP Blue proved to be agile, evolvable, and highly scalable. In rapid succession, we produced a 256, 768, and ultimately 1008 core system – using a total of 21
BEE2 modules. This demonstration was the first ever 1000+ core emulator, and clearly demonstrated the feasibility of the RAMP approach.

Details of RAMP Blue appeared in:


Within the RAMP project, in addition to developing a set of system prototypes and a large base of libraries and other modules, we engaged in a variety of outreach activities. We developed an archive of publicly available designs, and a website with detailed descriptions of various projects, publications, and other materials. During the height of the project, the website recorded consistent external accesses in excess of 5000 unique visits per month. Furthermore, the design archive typically logged over 1500 monthly repository downloads, the majority from unique sites, pointing to ongoing and widespread community interest in the effort. This public success underscored the validity of the direction and relevance of the research effort.

Student involvement was facilitated through incorporation of RAMP infrastructure in lectures and presentations, as well as in courses. David Patterson’s graduate Computer Architecture course helped create hardware components which were utilized in the design of the RAMP Blue system. In another effort, the RAMP-1 hardware was as used as a verification platform for coursework that developed the novel highly concurrent FLEET computer architecture (led by Ivan Sutherland of Sun Research).

During the second project year, dissemination of RAMP concepts to external academic and industry researchers was accomplished in two highly successful tutorial/workshops held in conjunction with leading computer architecture conferences:

- ISCA 2007. We held an all-day, hands-on tutorial at the International Symposium on Computer Architecture (ISCA) at the 2007 Federated Computer Research Conference (FCRC) in San Diego that was attended by over 50 researchers. RAMP Blue was demonstrated. The participants also developed, debugged, and optimized parallel programs with transactional memory using work developed by Stanford under RAMP.

- ASPLOS 2008. At the Architectural Support for Programming and Operating Systems (ASPLOS) 2008 conference in Seattle, we held another all-day tutorial attended by over 20 registered attendees. We provided several demonstrations of RAMP prototypes, and operated a hand-on workshop.
We also had considerable interactions and collaborative work with a number of companies and with other universities not part of the official RAMP collaboration.

- Sun Microsystems-OpenSPARC. We worked with Sun, to develop a complete FPGA system was built around a single OpenSPARC T1 core and implemented on a number FPGA boards, including two generations of BEE platforms (http://www.opensparc.net).

- Xilinx, Inc. Xilinx Research Labs worked closely with us and Sun Microsystems to implement OpenSPARC T1 on Xilinx FPGAs. Xilinx has also made a significant commitment to the RAMP effort by donating FPGA devices, tools, and expertise, along with funds to acquire FPGA boards.

- Microsoft Research. Microsoft contributed to the design and prototyping of the BEE3. They also use the BEE3 systems internally for their own research in computer architecture and acceleration of algorithms, and exploration of future operating systems, and are a regular contributor of design blocks.

The RAMP approach has proven to be an effective approach to experiment with architecture, micro-architecture, implementation alternatives, and as an early platform for software developers. However, perhaps the biggest strength in this approach is the ability to co-optimize both software and hardware. It is this property that spawned two other research initiatives:

- Architecture design within the Berkeley’s Parallel Computing Laboratory (Par Lab). The use of RAMP became one of the founding principles behind the ParLab research agenda. A Par Lab project called “RAMP Gold”, led by Krste Asanović, developed a FPGA based emulator for the Par Lab manycore architecture design and used it as a simulator for software and operating systems development.

- Lawrence Berkeley National Lab (LBNL). In a project called “Green Flash”, we developed energy efficient supercomputers for climate modeling. Based on extrapolations of existing climate modeling code and current day supercomputers, was estimated that computer systems consuming around 100 MWatts would be required to meet the needs of earth scientist for accurate climate modeling. Such machines would be too expensive to operate. We developed a design methodology, based on a combination of RAMP emulation and software tuning, to co-tune the hardware and software for efficient operation within particular application domains.

The ideas behind our co-tuning approach and initial results appeared in:

2.2 Parallel Processing and Multi-core Architectures Design

From Fall 2005 through Spring 2007 I participated in biweekly meetings within the Computer Science Division discussing the future of parallel processing systems. These discussions led to the following technical report, and directly to the proposal to establish the Berkeley Parallel Computing Laboratory (Par Lab), multidisciplinary research project exploring the future of parallel processing:


Within my group, some of our early work associated with the Par Lab was on investigation into yield enhancement of multi-core systems:


Our work with LBNL on co-tuning, described in section 2.1 was of great interest to the Department of Energy. In Spring 2009, we submitted a large proposal to extend this work, and funding commenced in Fall of 2009 for three years (with me as PI, and Asanovic as co-PI). In this project, we continued to develop and demonstrate the approach of hardware/software co-tuning, and extended the RAMP technique to include automatic generation of custom chip (ASIC) implementations of computer architectures. A major outcome of this project was the development of a new hardware description language and associated tools, called “Chisel”. Hardware/software co-tuning involves aggressive design space exploration, and existing tools make it too difficult to rapidly explore many design points. We had the notion that hardware design languages should adopt ideas from modern programming languages. I prototyped a hardware construction language in Ruby. Eventually we settled on using Scala as our base language, and recruited Jonathan Bachrach (now an Adjunct Assistant Professor in EECS) to be in charge of the language development.

We developed Chisel to deal with our frustration from using existing hardware description languages to build circuit generators for design-space exploration. Circuit generators must employ sophisticated programming techniques to make decisions concerning how to best specialize their output circuits according to high-level parameter values and constraints. Existing hardware description languages such as Verilog and VHDL lack the facilities present in modern programming languages, such as object-oriented programming, type inference, support for functional programming, and reflection. By embedding hardware construction constructs in the high-level programming language Scala, we enable the expression of flexible and scalable circuit generators. Not only are circuit descriptions expressed in Chisel more general, they are more compact – leading to easier to understand,
debug, and maintain code. Additionally, a key characteristic of Chisel is its ability, from a single source-level specification, to generate multiple implementation targets, such as ASICs, FPGAs, and software simulators (something that is nearly impossible with existing hardware description languages).

The following paper describes the base Chisel system. We are continuing to evolve and expand the language, adding higher level abstractions and support for various implementation targets and simulators, and building out a large library of hardware blocks. We have had several public releases (https://chisel.eecs.berkeley.edu) and are attracting outside users.


As Parlab was winding down I became part of a group to form a new lab, called ASPIRE, for “Algorithms and Specializers for Provably-optimal Implementations with Resiliency and Efficiency” (https://aspire.eecs.berkeley.edu), led by Krste Asanović. We applied for and were awarded a large DARPA grant and now have additional funding from Intel, Google, Oracle, and Nokia.

A key component of the ASPIRE approach is to achieve energy efficiency by combining specially designed hardware accelerators with manycore processor architectures. Fundamental to this approach is our ability to rapidly explore design alternatives. Our ongoing work with Chisel and associated design blocks, will give us this ability. Also, my years of work on application specific architectures and reconfigurable computing will also nicely complement the project. In my group, we are engaged in a line of work where we automatically analyze application memory requests to determine independence, then exploit this independence to parallelize execution within hardware accelerators.


3 Music, Audio, and Media Processing

3.1 RTP MIDI

At the 52nd IETF meeting (December 2001), we presented a proposal for a RTP (Real-Time Protocol) payload format for MIDI (the standard for coding the physical gestures that underly musical performance; playing piano keys, striking drum pads, pushing faders, etc.):
Shortly after the 52nd IETF meeting, RTP MIDI was accepted as an AVT working group item. Many industrial and academic experts participated in this working group process, and defined a format suitable for use in a wide range of present and future MIDI applications. Participants include members of the MIDI Manufacturers Association (MMA) and the Motion Pictures Expert Group (MPEG). We worked with the Audio/Video Transport (AVT) of the IETF, refining our RTP MIDI proposal, towards making it a standard on the internet. Our proposed RTP MIDI payload format and implementation went through many drafts. Eventually it was elevated by the Internet Engineering Steering Group (IESG) to an RFC – making it a de facto standard.

Apple’s implementation of our proposed RTP MIDI standard has been included in their operating system since May 2005; it has shipped with every Mac Computer since then. Lazzaro and I continue to collaborate on topics related to audio signal processing. Our recent paper presents an approach to filtering, intended to serve as a starting point for the design of dynamic filters based on a large set of audio descriptors, including some found in MPEG7.

3.2 Home Gateway / GSRC

From September 2005 through September 2009 I served as Theme 1 (Design Drivers) Leader for the Gigascale Systems Research Center (GSRC), funded by the Focus Center Research Program (FCRP) administered by the Semiconductor Research Corporation (SRC).

The focus of Theme 1 was to build and support common research infrastructure and to work on projects that would integrate the various researchers throughout the center. We had a major effort to distribute FPGA-based prototyping hardware and software throughout the center, see Section 1.4.

In the “home gateway” project we attempted to exploit the promised opportunities offered by ubiquitous and unlimited wireless connectivity combined with the vast computational
capabilities of deeply scaled CMOS. Our goal was to create an environment that seamlessly integrated all forms of residential media and broadband services and optimizes the user experience.

Our approach for dealing with the myriad of protocols and formats was to put the intelligence in the network, rather than in the (often portable and energy-constrained) end-devices. We developed the concept of a Universal Content Router. Our prototype used BEE2 as a central hub and a set of simpler FPGA modules for interfacing to display and audio systems. The hub routed data between AV source and sink devices, and provides multimedia data processing, including trans-coding, filtering, and transformation (for example, picture-in-picture).

The organizing principle behind our system design and our results in the project were described in:


Another major activity related to GSRC was to understand and articulate the range of emerging applications for Gigascale systems and IT platforms of the future. Towards that end, we organized a GSRC workshop titled Workloads of the Future, held in Santa Clara, California, on 30 November 2007. Conclusions from the workshop were published in:


### 3.3 BWRC “unPad”

Continuing on the general theme of emerging multi-media systems and applications, at the BWRC we embarked on an activity to define a long range research vision for the center. The “unPad” is our vision of the future where pads and other handheld devices cease to be necessary. The pad disappears, but its functionality (plus more) remains, in the form of unpackaged communication, computation, and storage. People seamlessly interact with data, the environment and one another through an interconnected set of sensor, actuators, and computing and storage devices. Sensor and actuators in the environment are used as input/output devices, and along with computing, storage, and communication bandwidth, will be commissioned on the fly as needed to provide the desired functionality. As a person moves through their environment, say from home to their car and then to their
office, a set of their data moves with them, and the sensors and actuators in whatever environment they happen to occupy, are clustered in an ad hoc way to support whatever services are currently needed. This strong sense of seamless mobility, and the opportunistic assembly and allocation of resources is what has been missing in earlier similar visions, such as ubiquitous computing.

As this latest work has represented a fairly substantial deviation from my other research activities, publications have yet emerged from it. However, we are engaged in several projects at the BWRC and the SwarmLab to advance various aspects of this vision, including work on wall scale RF-based sensor platforms (we call “eWallpaper”), universal radio platforms, low-latency audio/video/sensor data delivery thought IP networks, and operating systems mechanisms for distributed resource allocation and control (SwarmOS). I have made a number of presentations to a variety of companies, including Toshiba, Qualcomm, and others. I also gave a keynote talk at the ICT Industry and Technology Trend Forum hosted by Huawei. The forum was held at Huawei Headquarters – Shenzhen, China, Jan 16, 2013.

3.4 TerraSwarms

The unPad vision also became part of the TerraSwarm Research Center (www.terraswarm.org) proposed in Fall of 2012 and launched in January 2013. In this multi-university project we are addressing the huge potential (and associated risks) of pervasive integration of smart, networked sensors and actuators into our connected world. It is funded by the STARnet phase of the Focus Center Research Program (FCRP) administered by the Semiconductor Research Corporation (SRC). Funds come from the Defense Advanced Research Projects Agency (DARPA) and the SRC industry partners, including Applied Materials, GLOBALFOUNDRIES, IBM, Intel Corporation, Micron Technology, Raytheon, Texas Instruments, and United Technologies.

I am now Theme 1 co-leader and member of the executive committee. We are actively working on several center-wide applications (a “smart jukebox” and “smart buildings”) to help integrate and drive the center research activities. Working towards the unPad vision will be a major focus of my research for the next several years.