

RAPID Prototyping Technology

Huy Nguyen and Michael Vai

Rapid Advanced Processor In Development (RAPID) is a prototyping technology that accelerates the development of state-of-the-art processor systems, particularly those involving custom boards and firmware. This technology enables large productivity gains in prototyping and a significant reduction of development times from system concept to operation.



In order to stay competitive in the high-tech electronics consumer market, companies must continue to offer new products with superior capabilities, higher power efficiency, and smaller form factors at accelerated design schedules. For example, consumers upgrade their cell phones about every two years. Manufacturers are thus challenged to rapidly develop and produce phones that offer more advanced capabilities and smaller form factors at lower costs to meet consumer demands.

Military applications also require high-performance systems that can be developed at low cost and function within stringent size, weight, and power (SWaP) budgets. Furthermore, the asymmetric warfare aspect of our current defense needs has accelerated the requirement for high-performance, embedded processors to incorporate state-of-the-art hardware and software capabilities.

As a cost-saving strategy, many military applications rely on commercial technologies (e.g., games, communications, medical equipment) in the development of new systems. These devices, which evolve rapidly because of the potential high-volume markets and thus high profits, are also used by adversaries to support their activities, such as the remote detonation of improvised explosive devices. It is thus critical that leading-edge hardware and software be incorporated rapidly and effectively into our defense systems to maintain an advantage.

Lincoln Laboratory has been contributing to rapid capability development in recent years and has pioneered a prototyping methodology called Rapid Advanced Processor In Development, or RAPID. This methodology systematically reuses previously proven hardware, firmware, and software designs to compose application-specific embedded systems. The RAPID technique provides

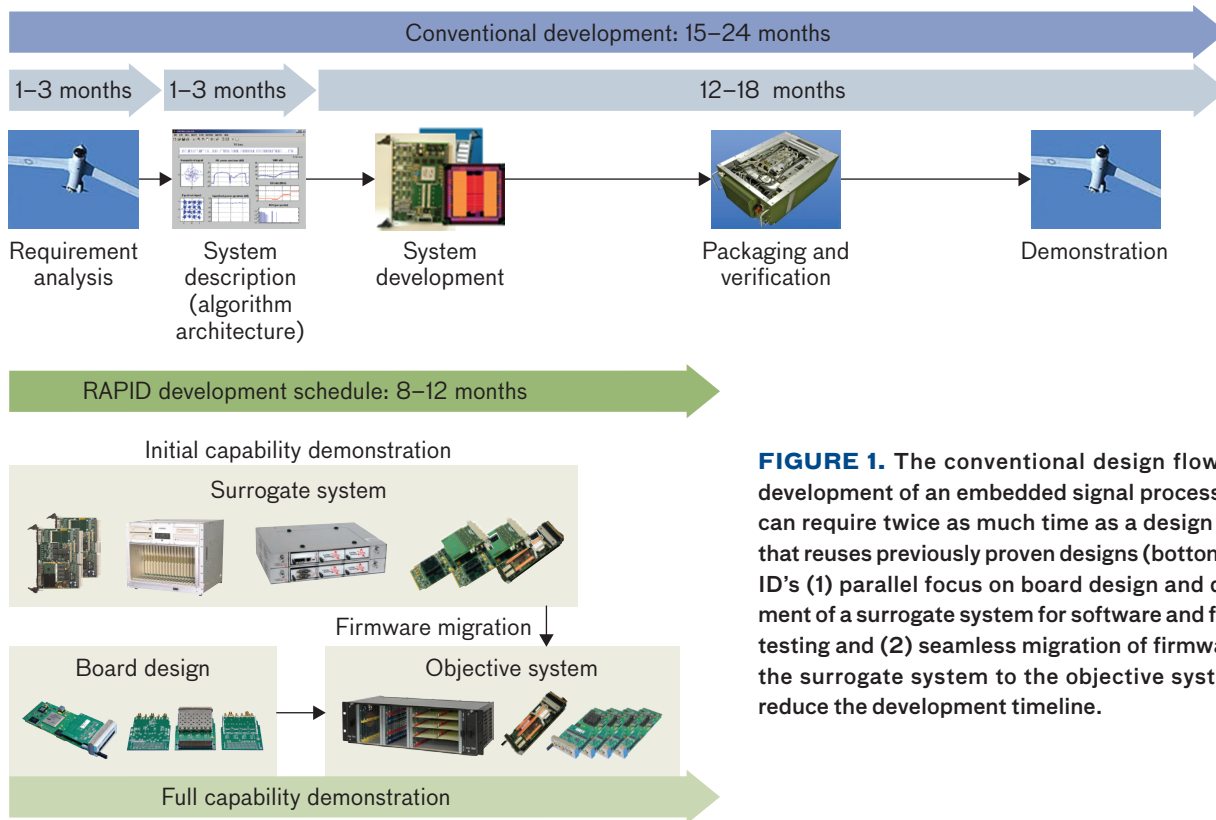


FIGURE 1. The conventional design flow for the development of an embedded signal processor (top) can require twice as much time as a design process that reuses previously proven designs (bottom). RAPID’s (1) parallel focus on board design and development of a surrogate system for software and firmware testing and (2) seamless migration of firmware from the surrogate system to the objective system help reduce the development timeline.

an easy process for a design to leverage Laboratory-wide expertise and experience, which are captured in a collection of documented, previously used good designs.

In addition, an efficient, integrated development environment that includes reference designs has been created to streamline the prototyping process. RAPID offers a field-programmable gate array (FPGA) common design environment, referred to as a *container*, that has open and stable interfaces. This container framework provides an enhanced controllability and observability of new designs, resulting in a significant productivity improvement in FPGA development, verification, and integration.

RAPID methodology mitigates risk factors associated with uncertainties in hardware and software performance, thereby increasing the probability of a first-pass success. Large savings in development time have been demonstrated in the prototyping of several high-performance, embedded processors for various sensor applications.

Prototyping

In many situations, especially those involving cutting-edge technology, new designs have unanticipated problems that are difficult to predict by modeling or simulations.

When the performance of a new device is uncertain, an early development of a prototype can be useful for testing key features of the design, exploring design alternatives, testing theories, and confirming performance prior to starting production. Prototyping is typically an iterative process, in which a series of products will be designed, constructed, and tested to progressively refine the final design. It is thus essential to minimize the latency of each prototyping cycle so that projects adhere to the original design schedules.

A typical prototyping flow of a high-performance embedded signal processor begins with a design phase in which the desired system capability is analyzed to determine hardware and software requirements. Next, in the implementation phase, the signal processing software and appropriate computational hardware are developed accordingly. Figure 1 depicts an example design flow of an embedded processor. In the conventional development flow, design steps are executed sequentially, and the entire process takes between 15 and 24 months. In the new RAPID design flow, a surrogate system is used for software and firmware development while the processor boards are being customized and fabricated for the objective system. The open-interface container allows firmware developed

and demonstrated in the surrogate system to be migrated seamlessly into the objective system. The complete system development time is reduced to 8 to 12 months.

Application-specific technologies are often used to optimize the prototype performance so that it will meet real-time requirements. For example, many military applications (e.g., radars) have data rates as high as several billion samples per second. These signal-processing applications have very demanding computational requirements that are currently beyond the capability of programmable processors and require the use of application-specific integrated-circuit (ASIC) and FPGA technologies. ASICs can offer very high performance because they are designed and manufactured for a specific purpose. This advantage comes with the cost of an extremely high design complexity and a commitment to a chosen design. In contrast, an FPGA is a fully manufactured device that contains an array of configurable logic blocks, memory blocks, arithmetic blocks, and interconnects that are designer controllable. The reconfigurability of FPGAs renders them especially attractive in prototyping because, unlike ASICs, they allow changes. As these application-specific technologies allow a custom processor to be tailored specifically for the signal processing task at hand, the overhead of a general-purpose programmable processor is eliminated. However, these advantages are offset by a longer design time and reduced flexibility. As such, ASICs and FPGAs are typically only used to reduce the data volume to a rate within the capability of programmable processors, which complete the signal processing. A number of programmable technologies are available, such as general-purpose processors similar to those used in desktop computers, digital signal processors (i.e., general-purpose processors optimized for signal-processing tasks), and graphics processing units used for their capability of supporting many parallel tasks.

Given a specific application, the designer will mix and match different processing technologies to achieve the desired performance. Figure 2 depicts one such design flow for a heterogeneous signal processor, which includes programmable processors, FPGAs, ASICs, and, potentially, a graphics processing unit. The design space of this processor has four dimensions: algorithm or architecture, processing technology, processor board, and packaging. Some design considerations for each of these four dimensions are shown in Figure 3. Instead of searching for local

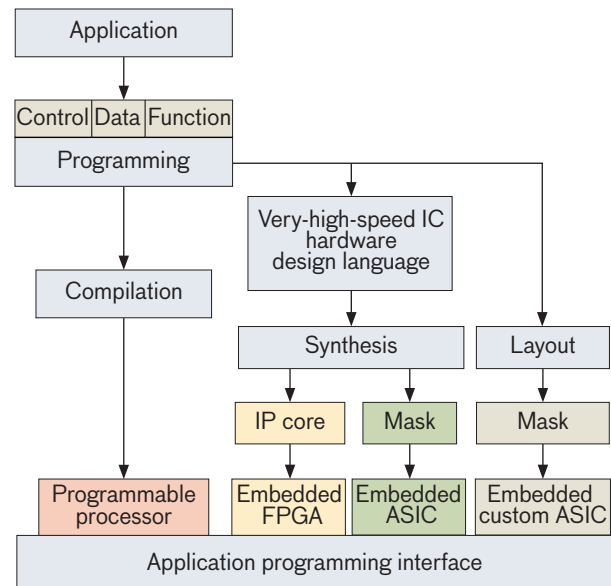


FIGURE 2. A design flow for a heterogeneous signal processor includes programmable processors, FPGAs, intellectual property (IP) cores, synthesized and custom ASICs, and, potentially, other hardware, such as a graphics processing unit (not shown).

optima on individual dimensions, the designer is better served by a global view that balances competing objectives, such as development cost, production cost, performance (operation speed and power consumption), time to market, and volume expectation.

The success of a design depends on the availability of performance benchmarks. However, realistic and scalable benchmarks are not widely available. Vendor-touted performance is often theoretical performance that is obtained under ideal conditions. Furthermore, benchmarking must be performed at both device and system levels to model multiple chip and board behaviors. Without an accurate benchmark at the system level, the same chip could perform differently when used in different boards.

Lincoln Laboratory has a long history in developing and building high-performance systems for military applications. The computing hardware can be a custom one-of-a-kind design (e.g., ASIC-based) or a commercial off-the-shelf (COTS) product. The COTS products have standard form factors so that they can be assembled easily. Also, COTS vendors often provide software and firmware libraries (also called *intellectual property cores* or “IP cores”) to facilitate the design process. COTS products are generally preferable in rapid

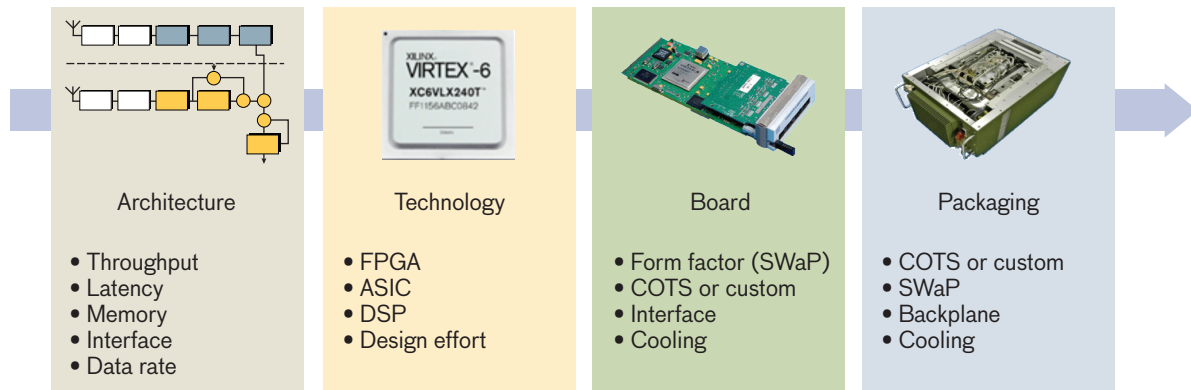


FIGURE 3. Design considerations are grouped across four design dimensions: architecture or algorithm, processing technology, processor board, and packaging.

prototyping activities because of their shorter implementation times. However, when the latest technology (e.g., the largest and fastest FPGAs) is required to meet the demands of rapid capability applications, these commercial products, which are designed to target a broad market, may not be tailored for the application at hand and alterations may not be ready in time. Furthermore, it can be easy to either overdesign (higher cost) or underdesign (failure) a system that uses new COTS products, as their performance in realistic environments often differs from vendor claims.

Developing custom processors is a viable alternative but not a panacea, as these systems still have similar problems to COTS products. Industry has many anecdotes of board development budget and schedule overruns. In addition to the cost of chips, hardware and software development costs are also significant. A typical system will need one or more printed circuit boards (PCBs), support components (e.g., memory), and hardware or software interfaces with other devices. It is especially challenging to integrate FPGAs, ASICs, and high-speed inputs and outputs on a complex PCB. For example, an FPGA can have more than 1000 pins, which cause a routing challenge that requires a high number of PCB layers. Signal paths have to be precisely matched in length to enable high-speed operations. An approach that optimizes the design at both system and chip levels should be taken, and much synergy is required between design team members to achieve such an integration.

Lincoln Laboratory has been developing embedded processors using a so-called “Lincoln off-the-shelf” (LOTS) approach that draws upon previous designs. When a new project begins, the reuse of a previously

proven custom processor board design is considered, as this board’s capability is well understood and could be adapted to meet the new program requirements.

Figure 4 displays examples of the LOTS approach, in which a base design was modified to support multiple programs. A custom, sophisticated radar-channelizing and adaptive-beamforming processor was developed in about two years for an intelligence, surveillance, and reconnaissance (ISR) application. This processor was later adapted to be used in a new space observation application after it was determined that there were no COTS products available to satisfy the requirements. Within eight months, the firmware was developed, integrated, and tested, and the system was fielded for this new application.

The baseline processor was also revamped to develop a real-time radar processor after it was determined that the use of COTS boards would present a high risk to the project schedule. Even though the circuit board had to be modified and manufactured to accommodate a data interface that operates four times faster, the new application was completed within a year. This radar processor was further adapted for a multifunctional phased-array radar and was developed in just 6 months.

The LOTS approach achieves a significantly faster turnaround time by leveraging previous nonrecurring engineering investments and team experience. As the baseline processor board has been thoroughly characterized, the chance of a first-pass success is improved. However, risks and issues similar to those of COTS still exist, and upgradability is a concern as new technologies, such as new FPGAs, must be incorporated as they become available to deliver the best performance possible. In addition, the LOTS approach still lacks the flexibility to meet the

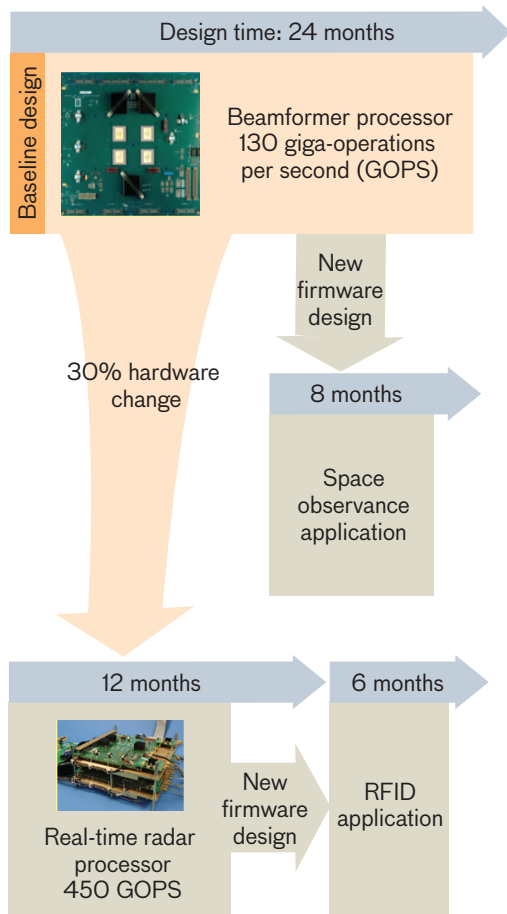


FIGURE 4. Three examples of Lincoln-off-the-shelf designs that each alter the base design to meet the demands of a new application (space observation, real-time radar, and radio frequency identification, or RFID).

quickly changing challenges in fighting an asymmetric warfare. To address these setbacks, the LOTS approach has been expanded into a RAPID prototyping methodology that systematically reuses previously proven hardware, firmware, and software designs to develop embedded processor systems.

RAPID Methodology

RAPID prototyping methodology’s key features include reusing previously proven designs, a highly productive design environment, and an inexpensive prototyping test bed. The design of the test bed allows the infusion of new technologies, while maintaining a stable user interface.

Design Reuse

Reusing previously proven designs saves development time and mitigates risk in time-critical projects. The flowchart in Figure 5 illustrates an example design process for a processor system having custom boards and FPGA firmware. The designer first searches the RAPID Wiki Design Reference Library for a match. If a previous design exists that satisfies the project’s needs, the designer downloads the relevant design database for building the board. If modifications are needed, the designer consults with the original board’s designer to gain insights, reducing the learning curve and potential for mistakes. Any new boards and associated firmware and software created with this process can be easily uploaded into the

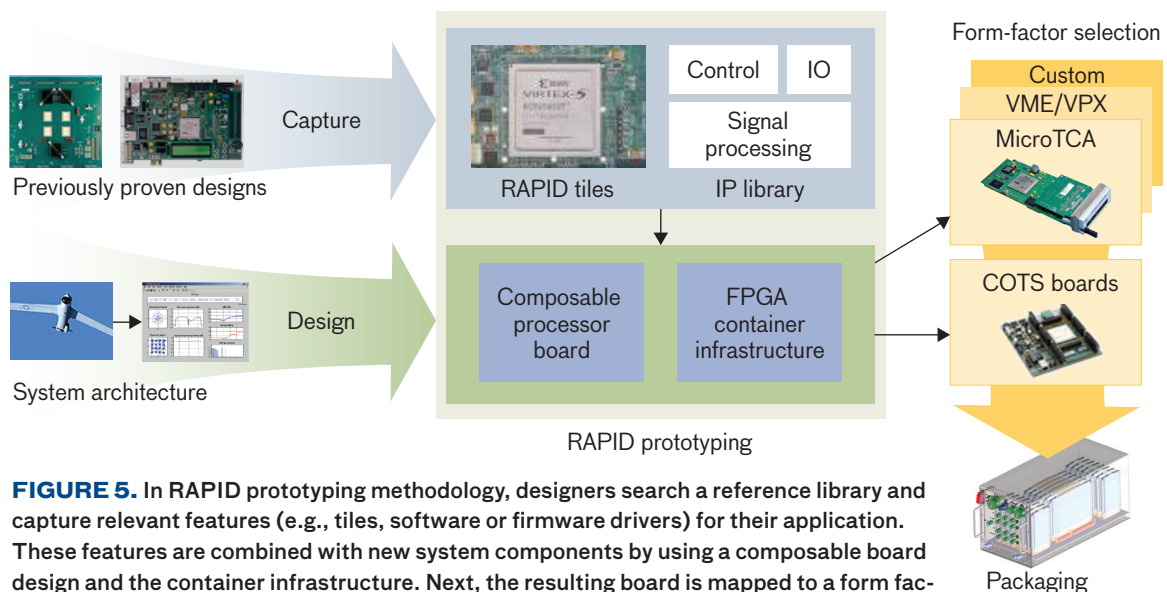


FIGURE 5. In RAPID prototyping methodology, designers search a reference library and capture relevant features (e.g., tiles, software or firmware drivers) for their application. These features are combined with new system components by using a composable board design and the container infrastructure. Next, the resulting board is mapped to a form factor (standard or custom) and packaged for use.

RAPID Prototyping at a Glance

The Rapid Advanced Processor In Development (RAPID) technology has been used successfully in several programs and is gaining support from the Lincoln Laboratory design community.^a Several groups have contributed reference designs to a RAPID Wiki Portal

a memory block and its interface to an FPGA (collectively called a “tile” in RAPID terminology) can be extracted and stored in the library for future use. This library of verified circuit board tiles and intellectual properties constitutes the first component of RAPID.

and tested. The open interface provided by the container significantly enhances the portability of the cores. Any cores developed on a surrogate platform can be ported over at a later time when the objective system is available. The result is a significant productivity improvement in FPGA development, verification, and integration.

The third component of RAPID is a heterogeneous processing test bed. Serving as a surrogate development platform, this test bed supports the early capability benchmarking and demonstration tasks in rapid prototyping programs.

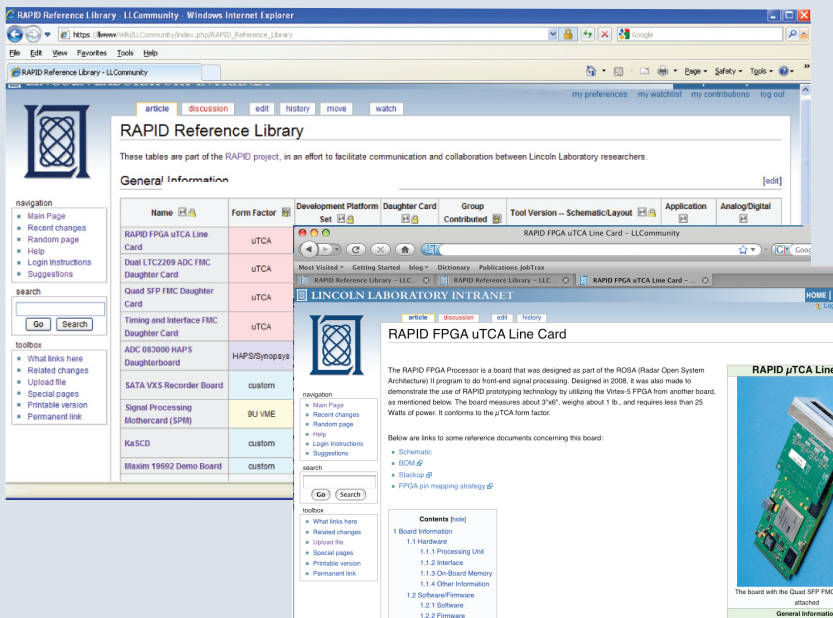


FIGURE A. Two screen shots of the Lincoln Laboratory RAPID Wiki Portal, which helps designers document, share, and reuse previously proven designs.

that is accessible from within Lincoln Laboratory. The wiki, shown in Figure A, was created to promote design reuse and sharing.

To help designers acquire expertise in new technologies and mitigate uncertainties, RAPID technology provides a process for leveraging Laboratory-wide experience and expertise, which are captured in a collection of documented previously proven designs. For example, the schematic and layout of

Another key component of RAPID is the container, a high-productivity FPGA design environment that is supported by a test bed.^b

The container provides enhanced controllability and observability of the application under development by enabling the designer to access the function cores from a host computer via a gigabit Ethernet connection. Each function core or group of cores can be individually addressed, configured, controlled,

REFERENCES

- a. H. Nguyen, M. Vai, A. Heckerling, M. Eskowitz, F. Ennis, T. Anderson, L. Retherford, and G. Lambert, “RAPID—A Rapid Prototyping Methodology for Embedded Systems,” *Proc. High Performance Embedded Computing Workshop*, 2009.
- b. A. Heckerling, T. Aderson, H. Nguyen, G. Proce, S. Siegal, and J. Thomas, “An Ethernet-Accessible Control Infrastructure for Rapid FPGA Development,” *Proc. High Performance Embedded Computing Workshop*, 2008.

RAPID Wiki Portal for future use by the Lincoln Laboratory community. The reference design library consists of schematics, layout, component data sheets, design reviews, and software and firmware drivers for previously proven designs. The most valuable benefit, though, is the venue for designers to discuss functional trade-offs and lessons learned in the design process. The availability of this expertise is crucial for reducing design uncertainty and increasing first-trial success.

The RAPID user, in consultation with the original designer, must decide what level of design reuse is appropriate for a specific project. For example, if there is a significant overlap in functionality, it may prove most advantageous to use the design as a starting point, delete superfluous items, and add new components. This is the usual previously proven board approach. When several pieces from various previously proven designs are to be integrated, a new method called Composable Board Design is used.

A user may extract elements of previous designs into tiles in computer-aided-design (CAD) format. Recently, a number of commercial PCB design tools are beginning to support the creation of a new circuit board by merging two or more previous designs and modifying the result. The resultant board layout is then mapped to a desired form factor. The design can be a standard size or a custom size to fit small and irregular enclosures, such as the payloads of miniature, unmanned aerial vehicles.

Note that RAPID prototyping methodology does not exclude the use of COTS boards, especially those successfully used in previous projects. In fact, a good source of library elements is the evaluation boards available from component vendors who routinely develop and sell evaluation boards that integrate their latest products (FPGAs, analog-to-digital converters), IP cores (interface), and other common peripheral devices (memory, Ethernet interface). These evaluation boards are excellent surrogates for developing firmware for specific applications while the custom circuit boards are being developed, thus converting a sequential design process into a parallel one. Furthermore, the schematics and layouts of evaluation circuit boards are often available and can be used to populate the reusable tile library. This approach provides an easy path to keep the library synchronized with state-of-the-art technologies.

High-Productivity Design Environment

As mentioned earlier, the reconfigurability of FPGAs motivates their use in many areas that require application-specific performance. This FPGA benefit will only be fully realized if a design environment that facilitates application development and debugging is available. Unfortunately, current FPGA design tools require the designer to write code to perform almost any debugging activities, such as setting and examining the internal values of an FPGA. This situation is reminiscent of the early days of computing when computers did not have an operating system. In addition, lacking a low-overhead, standardized control infrastructure for the FPGA is a huge barrier for other subsystems to interface with the FPGA.

The above limitations are addressed with the *container*, a small-footprint, computer-accessible control structure on the FPGA. As shown in Figure 6, the container provides an infrastructure on which a developer can build an application quickly. Through Ethernet connections, external software can observe and control the internal states of an application function core being developed. In fact, the container has enough functionality to serve as a computer-FPGA control interface for a real-time FPGA-based processor system.

The container is accessible through software calls from a host computer. A C++ software library allows the application software on the host computer to request reads and writes to the FPGA address space by handling the details of formatting one or more requested gigabit Ethernet (Gig-E) packets and interpreting the returned results. In this manner, the process is abstracted to simple

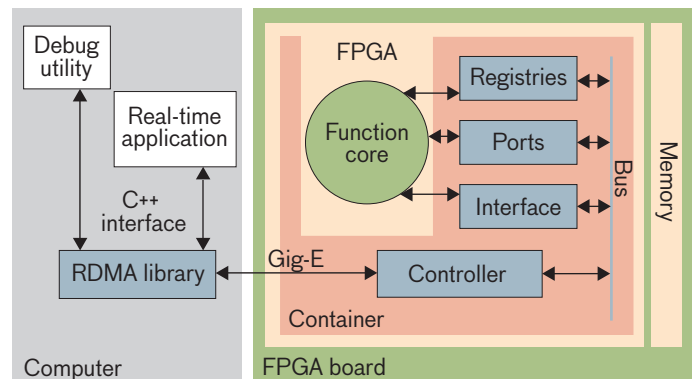


FIGURE 6. The computer-accessible *container* framework control structure for the FPGA provides an infrastructure on which a developer can build an application quickly.

remote direct-memory access (RDMA) calls. In the current version of the container, calls to the software library are implemented by sending control messages to the FPGA using the User Datagram Protocol (UDP), although other underlying protocols also may be used after minor changes to the calling application.

During the FPGA debug phase, interactive data probing is more desirable than running compiled programs. Therefore, a command-line interface may be used for loading data into an FPGA, initiating processing, and retrieving the output data and status. The command-line interface provides a similar functionality to the C++ software library. Using this command-line interface, commands can be entered interactively or issued with a prepared script file. Typically, a developer would first use the command line to verify FPGA operations, proceed to using scripts for automatic FPGA processor testing, and eventually create a C++ program to integrate the FPGA processor into the overall system. Figure 7 illustrates components on the FPGA side of the container structure: a UDP controller, a DMA controller, a Wishbone bus (an open-source hardware computer bus), and Wishbone peripherals.

The UDP controller receives packets from an Ethernet media access control (MAC) and decodes properly addressed and formatted UDP packets into commands for the DMA controller. UDP was chosen as a transport-layer protocol because it is efficient and more suitable for implementation in digital logic than a complicated protocol such as the transmission control protocol. The command-response protocol implemented on top of UDP was designed for simple translation into commands for the DMA controller.

The DMA controller translates the received commands into the required master read or write cycles on the Wishbone bus, providing a simple connection between the DMA controller and a variety of registers and peripherals that are useful for FPGA development. Once the command has been executed by the DMA controller, the resulting status and data responses are repackaged into UDP messages and reported back to the network address that made the request.

The Wishbone bus is an industry standard for memory-mapped, open-source buses that are used to connect devices on the same chip. In general, it connects one or more “master” devices that generate read or write cycles to

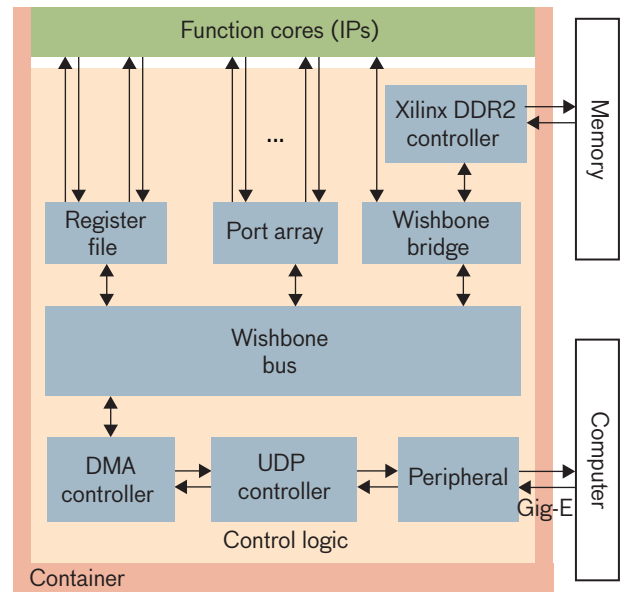


FIGURE 7. The FPGA side of the container structure includes a UDP controller, a DMA controller, a Wishbone bus, and Wishbone peripherals, some of which were developed at Lincoln Laboratory.

one or more “slave” devices that respond to read or write cycles within an assigned range of addresses.

“Register File” and “Port Array” are two Wishbone-compatible peripherals developed at Lincoln Laboratory. The Register File provides access to a set of registers for general control and monitoring of an FPGA application.

The Port Array provides a set of first-in, first-out (FIFO) ports, each of which has an address and can be written to or read from. The port array can be used for testing purposes to communicate with an FPGA processing core.

Another Wishbone peripheral developed at Lincoln Laboratory is a dual-port memory controller bridge for off-chip DDR2 SDRAM memory access. This bridge has one port that connects the DDR2 controller to the Wishbone bus and a second “pass-through” port for the processing application. This design allows high-speed processing logic to share memory with the lower-speed control and debugging logic. From the computer, an input pattern can be easily loaded into memory as a stimulus; processed results can be read back to the computer for application debugging.

RAPID’s controller infrastructure was implemented and tested on the Xilinx Virtex-5 family. The resource usage or overhead of this infrastructure on the Virtex-5 95SXT is between 7 and 12% and is summarized in Table 1. The software library has been tested under Windows XP/Cygwin and VxWorks. The highest communica-

ARCHITECTURE	LOOKUP TABLES	FLIP-FLOPS	BLOCK RAM (KBYTES)	CLOCK RATE (MHZ)
Controller core functions	3172	3853	83.25	125
Register file	132	200	0	0
Port array	96	1	0	125
DDR2 bridge / memory controller	2309	2275	31.5	125 200
Total	6009	6859	114.75	
(Total as percentage)	(10.2%)	(11.6%)	(10.5%)	

TABLE 1. RAPID's controller infrastructure was implemented and tested on the Xilinx Virtex-5 95SXT. The resource usage or overhead of this infrastructure is between 7 and 12%.

tion rate with the computer, as supported by the current software library, reached 13 MB/s. On the FPGA side, the control infrastructure is expected to support data rates of gigabit Ethernet speeds or higher. The achievable data rate will depend on the specifications and operating conditions of the FPGA.

RAPID Test Bed: A Surrogate Development Platform

A RAPID heterogeneous processing test bed has been implemented as a surrogate development platform to support early capability benchmarking and demonstration tasks in rapid prototyping programs. This test bed is equipped with stable interfaces and appropriate software/firmware support to improve application development productivity. In addition, this test bed can be readily replicated at low cost to support multiple programs at the same time.

A basic configuration of the test bed has a MicroTCA chassis (a standard form factor) that contains one single-board computer and one or more FPGA boards. The MicroTCA design environment provides a gigabit Ethernet hub connecting all payload slots in the system via a high-speed backplane connection that supports the RAPID container development framework. The costs for a complete MicroTCA development system start between \$2000 and \$10,000, which is equivalent to the price of a single processor board available from a defense industry vendor.

Multiple general processing unit nodes and PCI Express expansion capabilities can be added to the test bed. In order to support new hardware and communication protocols, the container infrastructure is being augmented with additional capabilities. For example, new

communication protocols such as Serial RapidIO and PCI Express are being evaluated.

Applications

RAPID prototyping methodology has been successfully employed in the development of a number of new, challenging designs. Three applications of the RAPID methodology have been selected as examples: a four-channel adaptive beamformer radar processor, a twenty-channel vehicle-mounted laser vibrometer signal-processing system, and an FPGA front-end processor for an airborne synthetic aperture radar (SAR) imaging system.

Adaptive Beamformer Radar Processor

RAPID methodology was used in the development of a front-end processor for the Radar Open Systems Architecture II (ROSA II) project, in which a common infrastructure for modular hardware and software enables radar systems to be implemented and upgraded with minimal overhead. New enhancements for ROSA II included a four-channel adaptive digital beamformer, which enables airborne systems with higher pulse rates, and a publish-subscribe capability through thin communication layers for even more flexibility in system data and message passing.

One of the key challenges of this project was the high level of concurrent development. The front-end processor was planned to be a critical subsystem of a ROSA II system demonstration, and its development was underway while the specifications for ROSA II system were still being finalized. However, by using the RAPID container framework, the design team was able to commence devel-

opment of the signal processing portion while the control portion was still evolving. As shown in Figure 8, some of the signal processing included analog-to-digital conversion, digital in-phase and quadrature processing, adaptive beamforming, and data packetizing.

The processor included several boards and modules, such as a high-performance FPGA processor board in MicroTCA form factor and a number of FPGA Mezzanine Card boards created with RAPID methodology. The FPGA board leverages the design of an evaluation board chosen from the repertoire of an FPGA vendor. Based on a Virtex-5 FPGA operating at a peak frequency of 550 MHz, the processor provides a throughput of 100 to 200 gigaflops per second and consumes 25 W. The board also hosts 1 gigabyte of RAM operating at 3.2 gigabytes per second (GB/s) and 8 megabytes of SRAM at 0.8 GB/s. The external input and output data rates are 5 GB/s.

Although this was the pilot test run of RAPID prototyping methodology and extra time was spent in tool configuration and verification, the FPGA processor was completed in five months. It is expected that an experienced design team could deliver a design of similar complexity in only three months.

The open-interface container approach allowed the design of the FPGA firmware to begin simultaneously with the processor board design. The firmware was verified on the test bed using a surrogate COTS processor with only a quarter of the required throughput, permitting a six-month head start on the development of FPGA firmware. When the target processor was completed, the team demonstrated a seamless migration of FPGA functionality from the surrogate system to the objective platform in just three weeks.

Laser Vibrometer Signal Processor System

RAPID methodology and test bed were also leveraged in the development of a vehicle-mounted laser vibrometer system. The signal processing subsystem involved the filtering and instantaneous frequency demodulation of 20

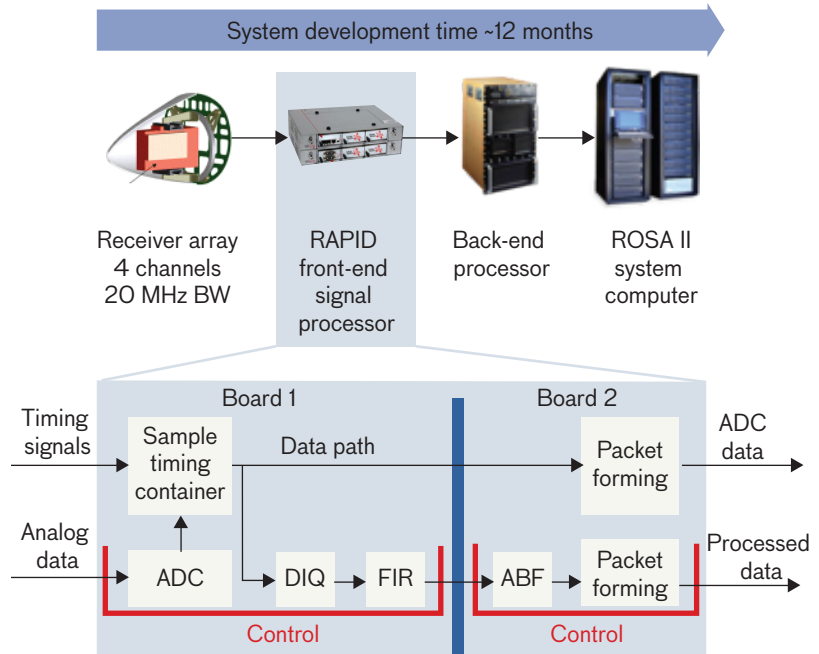


FIGURE 8. RAPID methodology was used for designing a processor for the ROSA II system, whose development was underway while the specifications were still being finalized. Some of the signal processing included analog-to-digital conversion (ADC), digital in-phase and quadrature (DIQ) processing, finite impulse response (FIR), adaptive beamforming (ABF), and data packetizing.

signal channels, all performed in real time.

Because of the schedule of the application, there was significant overlap between development and experimentation. For example, the signal processing flow was designed and evaluated while parameters such as processing block sizes, method of detection, etc., were still under investigation. In this situation, the collection of raw data for analysis was extremely valuable. After a minor modification to the RAPID test bed, a functional recording system was delivered in three weeks. This is a remarkable turnaround time when compared to the six- to eight-week window typically required for acquisition of an equivalent COTS system, plus a few additional weeks required to develop the desired operations.

While the algorithm and associated firmware were being developed using the RAPID test bed, the semi-ruggedized objective hardware was advancing in parallel. A single-channel real-time processor was successfully created for a proof-of-concept demonstration. A seamless firmware migration from the test bed to the objective hardware is expected.

Processor for Synthetic Aperture Radar Imaging

RAPID's high-productivity container framework was also used in the design of an FPGA front-end processor for an airborne SAR imaging system. The processor interfaces with analog-to-digital converters, performs spectrum processing, and packetizes data into multiple gigabit Ethernet links that are fed into a back-end multicore, real-time processor.

Pressed to meet a short development schedule, the design team concentrated its efforts on the back-end, real-time processor (a 128-core parallel processor). The required high volume of data transfer between the front-end and the back-end processors would not have been developed in time without the efficient gigabit Ethernet infrastructure available in the RAPID library.

Future Work

RAPID prototyping methodology has been extended into a self-sustaining infrastructure to serve all of Lincoln Laboratory. As the embedded processor design community continues to adopt RAPID methodology, more and more design tutorials, examples, and workshops are being added to the library through the Wiki portal.

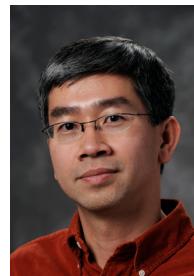
New strategic technologies are also being pursued, such as the development of a data-path container to augment the firmware development environment. This data-path container will support protocol standards, such as PCI Express and Serial RapidIO protocols, with the goal of incorporating the general-purpose, graphics processing technology into the RAPID test bed.

The grand vision for RAPID is to provide an integrated design environment for a heterogeneous embedded processor system that could easily be composed from different processing technologies along with their available intellectual properties. For example, a matrix computational function in the signal processing chain may be implemented in software for a proof-of-concept demonstration during the early phases of development. At a later phase of development, the software implementation can be retargeted to an FPGA for improved performance. This type of cross-technology migration will allow a new system to be quickly validated on a desktop computer, then migrated to a non-form-factor benchtop system for a real-time demonstration, and finally ported to an objective platform for field tests and deployment.

Acknowledgments

The authors would like to thank RAPID team members A. Heckling, T. Anderson, M. Eskowitz, F. Ennis, S. Siegal, A. Horst, L. Retherford, S. Chen, and T. Kortz for their contributions. Special thanks to R. Bond for his vision and guidance and to the Lincoln Laboratory Technology Office for funding support. ■

ABOUT THE AUTHORS



Huy Nguyen is a staff member of the Embedded Digital Systems Group. He has been involved with designing low-power high-performance signal processors for 15 years. He earned a bachelor's degree from the University of Delaware and a doctoral degree from the Georgia Institute of Technology, both in electrical engineering. Prior to pursuing his doctorate, he worked on real-time radar software at G.T.R.I. He joined Lincoln Laboratory in 1998.



Michael Vai is the assistant leader of the Embedded Digital Systems Group. He has worked in the area of high-performance embedded computing for over 20 years and has published extensively on the topics of very-large-scale integration, ASICs, FPGAs, design methodology, and embedded systems. He earned his bachelor's degree from the National Taiwan University, Taipei, Taiwan, and his master's and doctoral degrees from Michigan State University, all in electrical engineering. Prior to joining Lincoln Laboratory in 1999, he was on the faculty of Northeastern University.