International Conference on “Advances In Computing, Communication And Intelligence” ICACCI-2014 At Mauritius
Special Issue of International Journal of Electronics, Communication & Soft Computing Science and Engineering, ISSN: 2277-9477

Development of Reconfigurable Multiprocessor Systems

Archana Gomkar

Abstract- **Modern digital systems demand increasing electronic resources, so the multiprocessor platforms are a suitable solution for them. This approach provides better results in terms of area, speed, and power consumption compared to traditional uniprocessor digital systems. Reconfigurable multiprocessor systems are a particular type of embedded system, implemented using reconfigurable hardware. This paper presents a review of this emerging research area. A number of state-of-the-art systems published in this field are presented and classified. Design methods and challenges are also discussed. Advances in FPGA technology are leading to more powerful systems in terms of processing and flexibility. Flexibility is one of the strong points of this kind of system, and multiprocessor systems can even be reconfigured at run time, allowing hardware to be adjusted to the demands of the application**.

I. INTRODUCTION

Multiprocessor Systems-on-Chip (MPSoC) represent an important trend in digital embedded electronic systems. MPSoC are systems-on-chip with more than one processor. New applications in modern embedded systems require complex multiprocessor designs to reach Real-Time (RT) deadlines while overcoming other critical constraints such as power consumption and low area. MPSoC seem to be the solution for such complex systems. A lot of applications such as networking, multimedia, and control benefit from this type of system. The perfect example of this is a cell phone. Current models must offer low power consumption and integrate a large number of functions such as audio and video encoding, image processing, and Internet access. MPSoC offer better performance with lower energy consumption in this kind of complex systems compared to uniprocessor embedded systems. Traditionally, the trend in uniprocessor systems was to improve performance by increasing clock frequency; now the trend is to work in parallel with lower frequencies, in order to reduce energy consumption [1–3].

In the field of MPSoC, the reconfigurable or FPGA-based multiprocessor is a new and increasingly important trend. It facilitates rapid prototyping and allows research into new architectures and communications techniques without the problems of MPSoC ASIC production. The number of papers published over the last three years has increased significantly. Figure 1 shows the number of publications in the Inspec database using “multiprocessor” and “FPGA” as search keywords.

Reconfigurable Multiprocessor Systems, also known as Multiprocessor-on-Programmable Chip (MPoPC) (or Soft Multiprocessor), are normally presented as a way of making prototype systems for subsequent implementation on an ASIC. Now, not only prototypes are implemented using FPGAs, but final designs too. The growth in FPGA capacity allows designers to implement a complete multiprocessor system in a single FPGA. The main FPGA companies offer the possibility of using softcore processors specially designed to fit well in the FPGA; also, some FPGAs allow the use of hard-core processors. Furthermore, FPGAs are equipped with on-chip memory blocks, peripherals, and interconnection circuitry. Run-time reconfigurability is one of the strong points of FPGA-based multiprocessors systems. This feature allows multiprocessor systems to be adapted to a particular application, gaining flexibility in the designed system.

In the following section, we discuss the viability of FPGA-based Multiprocessors. In Section 3, we provide some examples of FPGA-based multiprocessors implemented by the research community in recent years. In Section 4, we examine the challenges of MPoPC. After that, a number of different methods of design are presented. In the final section of this paper, we highlight a number of important aspects relating to MPoPCs.

II. Viability of FPGA-Based Multiprocessor Systems

The first question we must ask is whether there is any sense in implementing a multiprocessor system on an FPGA. The answer is that it depends. The main disadvantage of this kind of multiprocessor is reduced performance com-pared with ASIC multiprocessor systems. However, FPGA-Multiprocessor systems have a number of advantages that compensate for this in some way.

(i) Flexibility and reconfiguration. The number of softcore processors that can be included is limited only by the capacity of the FPGA. Also, it is possible to configure each processor independently adding cache, FPU modules, and so forth.

(ii) Less time-to-market. The design process does not include the manufacture of the IC, with a considerable reduction in design time.

(iii) Less cost. The process is cheaper. Nowadays a state-of-the-art FPGA is relatively cheap, enabling own design with a small work team. Furthermore, if there is an error in the system design, this is not decisive.

(iv) Scalability. FPGA-based multiprocessors systems can house an increasing number of microprocessors or peripherals if there are logic resources available in the FPGA.
Therefore, using FPGA is the best choice in certain cases.

(i) Low-volume, mission-critical designs (e.g., radar and military applications).
(ii) Rapid design of new, reconfigurable multiprocessor systems.
(iii) Research field. New architectures, memory hierarchies, interprocessor communication, and so forth, can be developed.
(iv) Naturally-grown systems. Systems that have to be able to grow in features depending on the stage of development [4].

The use of FPGA technology provides numerous benefits for the design of embedded systems. These benefits include the ability to fix design bugs in the FPGA hardware, upgrade a system in the field, or simply swap out hardware functionality without redesigning the physical board that contains the FPGA [5]. FPGAs already provide a compelling time-to-market advantage over ASICs, and advanced tools targeting the needs of high-end FPGA designers would help establish clear leadership [6].

Ravindran et al. discuss the viability of soft multiprocessors [7, 8]. According to them, it is necessary to answer these two questions: (a) Can soft multiprocessors achieve performance levels competitive with custom multiprocessor solutions? (b) How do we design efficient systems of soft multiprocessors for a target application? In both papers, they demonstrate the viability of soft multiprocessors.

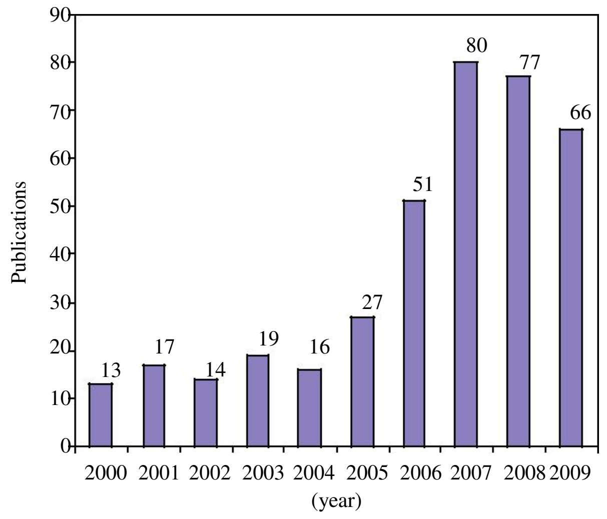

Figure 1: FPGA multiprocessor publications trend (2010 February).

III. FPGA-Based Multiprocessor Systems

In this section, we describe several multiprocessor systems implemented in FPGA. We have tried to include systems that represent the different trends in architecture and applications. First, we provide a little background information about MPSoC and FPGA-based multiprocessor systems. We also present the most widely accepted classification of MPSoCs: homogeneous and heterogeneous. Multiprocessor systems can be also classified as shared-memory systems or distributed-memory systems. We also highlight recent advances in run-time reconfigurable multiprocessor systems and give examples of these systems.

3.1. Main FPGA-Based Processors. In FPGA-based multipro-cessor systems, the most widely used FPGA soft processors are made by one of the two main FPGA companies: Xilinx or Altera. The other option is to use open-source soft processors.

Xilinx is the most widely used brand in MPoPC systems. It supplies three main processors: softcore MicroBlaze (MB), PicoBlaze (8 bits reduced soft processor), and hard-core PowerPC. The number of PowerPcs is limited by the FPGA model to a maximum of four. The number of MicroBlaze and PicoBlaze processors is limited only by logic resources.

The MicroBlaze is a 32-bit RISC softcore processor. It uses the Harvard memory architecture, that is, it has a separate instruction memory and data memory. The MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances. The shared-bus solution is the CoreConnect On-Chip Peripheral Bus (OPB), and every MicroBlaze has Fast Simplex Link (FSL) ports to make efficient point-to-point connections [9].

PicoBlaze is based on an 8-bit RISC architecture and can reach speeds of up to 100 MIPS in Virtex-4 family FPGAs. The processors have an 8-bit address and data port for access to a wide range of peripherals. The core license allows

Table 1: Main FPGA processor models.

Processor	Company	Type	Bits	Comments
MicroBlaze	Xilinx	RISC	32 bits	Harvard
PowerPC	Xilinx	RISC	32 bits	hard-core
PicoBlaze	Xilinx	RISC	8 bits	open-source
Nios	Altera	RISC	16 bits	obsolete
Nios II	Altera	RISC	32 bits	3 flavours
Leon 3	Gaisler	RISC	32 bits	open-source
OpenRisc 1200	OpenCores	RISC	32 bits	open-source
Mico32	Lattice	RISC	32 bits	open-source

them to be used freely, albeit only on Xilinx devices, and they come with development tools. The PicoBlaze design was originally named KCPSM which stands for "Constant (K) Coded Programmable State Machine" (formerly "Ken Chapman's PSM"). Ken Chapman was the Xilinx systems designer who devised and implemented the microcontroller.

The design flow provided by Xilinx for embedded systems is the Embedded Development Kit (EDK), which involves some limitations when implementing multiprocessor systems, even though it is widely used. These limitations are explained in Section 5.1.

Altera is the second most widely used processor utilized by researchers. It enables the use of Nios II (NII) processors, Avalon bus, and the EDA tool SOPC Builder, which is used as an aid in the development of multiprocessor systems.

Nios II is a 32-bit RISC embedded processor. Nios II is the evolution of the previous 16-bit Nios architecture. It is suitable for a wider range of embedded computing applications, from DSP to system control. Unlike Microblaze, Nios II is licensable for standard-cell ASICs through a third-party IP provider, synopsys designware. Through the designware license, designers can migrate Nios-based designs from an FPGA-platform to a mass production ASIC-device.

Nios II has three mostly unparameterized variations: Nios II/e, a small unpipelined 6-CPI processor with serial shifter and software multiplication, Nios II/s, a 5-stage pipeline with multiplier-based barrel shifter, hardware multiplication, and instruction cache, and Nios II/f, a large 6-stage pipeline with dynamic branch prediction, instruction and data caches, and optional hardware divider.

OpenRisc from OpenCores and Leon 3 from Gaisler are two very common open core soft processors. Leon 3 has more advanced functions than MicroBlaze and Nios II, but has the limitation of occupying a large amount of logic resources, so it is complicated to fit a large number of units in a single FPGA. We believe that it may be a good choice when larger FPGAs appear in the future.

Table 1 summarizes the main FPGA processor models (PowerPC is the only processor in the table that is not softcore).

3.2. Architecture Background. Normally the target application of the FPGA-based multiprocessor determines the architecture. There are three main system architectures: (1) Master-Slave, (2) Pipeline, and (3) Net. Also, it is possible to combine these: master-slave with pipeline, for instance, is very common.

(1) In master-slave systems, one or more processors act as the master processor, controlling the behaviour of the other slave processors.

(2) The pipeline approach is useful with stream applications; the architecture is composite with a chain of processors, every processor acting as a pipeline stage. The tasks are partitioned in time resulting in better performance if the application is adequate.

(3) Finally, net architecture refers to multiprocessor systems where there is no hierarchy between processors, all processors being able to communicate with each other when necessary. One example of this kind of system is the symmetric multiprocessor (SMP). A feature of SMPs is that all the processors are identical, so they are homogeneous multiprocessor systems.

Another important issue is the way the communica-tions connections are implemented physically. There are 3 approaches.

(1) Point-to-point, where the processors are connected directly. High bandwidth is an advantage because it is not necessary to share the communications channel, but when systems grow this is not area efficient.

(2) Shared-bus, the traditional approach that derives from uniprocessor systems. It is the best known mechanism to communicate cores, but it is not effective in terms of performance because the bus can only be used by one processor at a time.

(3) The most recent and promising approach is the network-on-chip (NoC). The basis of this method of interconnecting cores, is to apply network background to on-chip systems. When there are a lot of on-chip cores, it is the solution that best combines area and performance. The idea is to use small routers inside the chip to enable communications between all cores of the system with low latencies.

There are two possible methods for exchanging information between processors: shared-memory and message passing.

(1) Shared-memory is used most frequently, one reason for this being that FPGAs have a limited amount of on-chip memory, so this method allows memory saving. Shared-memory systems, like SMPs, have the problem of synchronization and memory consistency. In FPGA-based designs, it is an important issue under research, because the most widely used soft processors do not have any solution to deal with these problems. Normally, shared-memory multiprocessor systems use a shared bus but there are also some systems with NoC interconnection.

(2) Message passing is mostly used in distributed memory systems [16] and consists of exchanging messages between processors. A message passing protocol is required.

Table 2 summarizes MPSoC architecture.

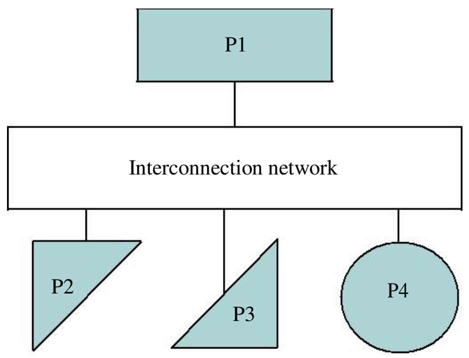

FIGURE 2: Heterogeneous multiprocessor systems.

3.3. Classification. The traditional classification of MPSoCs is heterogeneous, when there are different processors or even accelerators in the same system (see Figure 2) and homogeneous, when all the processors in the system are identical (same Instruction Set Architecture) (see Figure 3).

Normally, application-specific systems are heterogeneous. This is the common type of MPSoCs, as in FPGA-based multiprocessor systems. The reason is that MPSoC are usually implemented for embedded applications that behave intrinsically in a heterogeneous manner and require different kinds of processors.

Homogeneous MPSoCs are normally general-purpose systems, where all the processors are identical. In this kind of system, it is possible to increase the number of processors without changing the architecture (scalability property). It is easier to develop software for homogeneous systems.

Different taxonomies can be found in the literature. One is to classify systems in accordance with the memory architecture. There are two types: shared-memory (Figure 4) and distributed memory (Figure 5).

In shared-memory systems, all processors share the same memory resources; therefore, all changes made by a processor to a given memory location become visible to all the other processors in the system.

From an architecture design viewpoint, shared-memory machines are poorly scalable because of the limited bandwidth of the memory.

Shared-memory systems require synchronization mechanisms such as semaphores, barriers, and locks since no explicit communication exists. POSIX threads [17] and OpenMP [18] are two popular implementations of the thread model on shared-memory architectures.

In shared-memory architecture, different processes can easily exchange information through shared variables; however, it requires careful handling of synchronization and memory protection.

In distributed-memory systems, each processor has its own private memory; therefore, one processor cannot read directly in the memory of another processor. Data transfers are implemented using message-passing protocols.

Distributed-memory machines are more scalable since only the communication medium may be shared among processors.

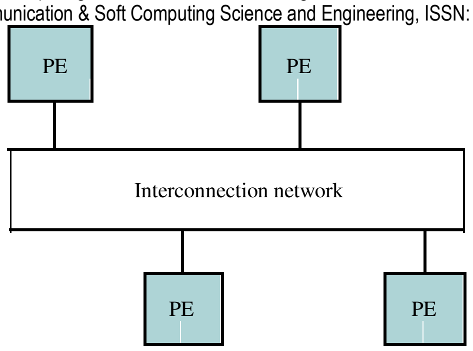

FIGURE 3: Homogeneous multiprocessor systems.

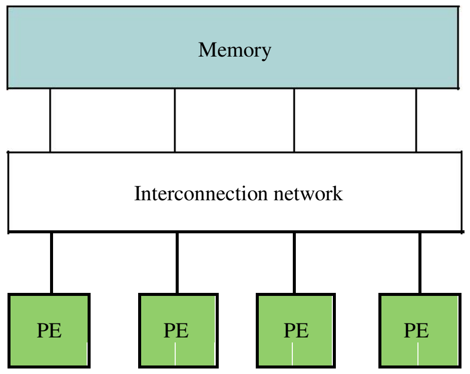

FIGURE 4: Shared memory multiprocessor systems.

TABLE 2: MPSoC architecture.

System arch	Comm arch	Comm Method
Master-slave	Point-to-point	Message passing
Pipeline	Shared bus	Shared memory
Net	NoC	

Distributed-memory systems require mechanisms for supporting explicit communications between processes. Usually a library of primitives that allow writing in communication channels is used. The Message Passing Interface (MPI) [19] is the most popular standard.

In distributed-memory architecture, a communications infrastructure is required in order to connect processing elements and their memories and allow the exchange of information.

We can also classify FPGA-MPSoC in (1) static reconfigurable multiprocessor systems and (2) run-time reconfigurable systems. Run-time reconfigurable systems repre-sent state-of-the-art reconfigurable multiprocessors systems. They use the dynamic reconfiguration FPGA feature to adapt hardware at run-time to a specific application. So, we have several dynamic modules which are loaded by an arbiter depending on the target application. In Figure 6, a scheme of a basic run-time reconfigurable system is depicted.

3.3.1. Heterogeneous FPGA-Based Systems. Most MPoPCs are application specific. In this case, the system is application dependent, so the architecture is designed to achieve the best

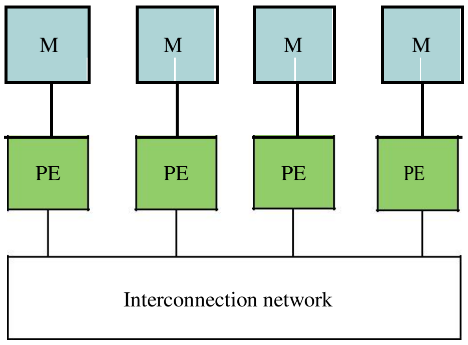

Figure 5: Distributed memory multiprocessor systems.

performance for the specific application. This architecture can be obtained by using design flow tools provided by FPGA vendors (known as hand-tune design) or using custom tools that obtain the architecture automatically from the specifications (known as automatic design) (see Section 5). There are systems that target different areas: multimedia, networking, control, and bioinformatics. First, we present application-specific FPGA-based multiprocessor systems (Table 3).

Network application is implemented in [7]. The solution proposed for IPV4 packet forwarding is a master-slave/pipeline approach. Communications among processors is point to point using MicroBlaze Fast Simplex Link (FSL) ports. There are different pipeline branches replicated in space to increase throughput. To demonstrate the viability of this solution, they compare the FPGA-based system proposed to an ASIC MPSoC with the same functionality. The FPGA-based system only loses a factor of 2.6X in performance normalized to area.

In [10], a master-slave/point-to-point multiprocessor system to control a laser-based transparency meter is presented. The authors are non-FPGA experts but they implemented the system successfully. According to them, the design tools made by the main vendors are easy to use. Tools abstract sufficient low-level details from the system design, allowing designers to implement successful MPSoC in a FPGA.

MPEG-4 Encoder is implemented in [11]. The system has master-slave architecture with support for message passing and shared SDRAM to interconnect NIOS processors. It uses a shared bus to connect instruction-shared memory and Heterogeneous IP Block Interconnection (HIBI) to connect data-shared memory by the plug and play method. It is an easy-to-scale computational system. Scalability is obtained through special parallelization: every image is divided into horizontal slices, and every slice is processed by 4 softcores in a master-slave configuration.

A stream chip multiprocessor (CMP) architecture for accelerating bioinformatic applications is presented in [12]. The architecture is master-slave/pipeline, and the memory hierarchy is customized for the target application. It is easy to increase the number of processors if there are enough logic resources in the FPGA. They compare the use of CPU, GPU, or FPGA for stream applications. GPU implementations achieve an order-of-magnitude speed increase compared to

Table 3: Application-specific FPGA-Based multiprocessor systems.

Ref	Application	Syst arch	Comm meth	Comm arch
[7]	Networking	M-S/pipeline	mess passing	point-to-point
[10]	Control	M-S	mess passing	point-to-point
[11]	MPEG-4	M-S	shared mem	shared bus/HIBI
[12]	Bio	M-S/pipeline	mess passing	point to point
[4]	Industrial	M-S	shared mem	shared bus
[13]	Automotive	Net	shared mem	shared bus
[14]	Automotive	pipeline	mess passing	point-to-point
[15]	RT control	pipeline	mess passing	point-to-point

optimized CPU implementations. Custom hardware implementations on FPGA of the test algorithm also exploit data parallelism to achieve speedups over CPUs [25]. They intend to make the prototype in FPGA and after migrate to GPU implementation.

In [4] a master-slave shared-bus/shared-memory architecture is used for industrial applications. They use Nios II softcore processors and an Avalon bus. In this paper, the authors discuss the advantages of using FPGA-based multiprocessor systems in industrial applications. Industrial production machines have to be highly flexible in order to satisfy changes deriving from the demand for new products.

The automotive sector is another area of application for FPGA-based multiprocessor systems. Tumeo et al. present a real-time solution [13]. This is a shared-bus/shared-memory multiprocessor system, but offers the possibility of exchanging small data packets using the message-passing method through a crossbar.

Another solution for the automotive sector is presented in [14], which proposes an NIOS II-based multiprocessor system. The architecture is totally built for the occasion using a pipeline approach, consisting of a chain of 15 processors connected point to point. They use a distributed-memory architecture where each processor executes independent tasks, instead of parallelizing a unique task between the different processors. The reason they give for not using shared-memory architecture is that each task execution time and access latency shared-memory are not the same and therefore the shared bus became a bottleneck. However, with the distributed-memory approach, each microprocessor has local memory and the latency time is lower.

Real-time control applications are the target of the system proposed by Ben Othman et al. [15]. The system has two or three processors connected directly using MicroBlaze Fast Simple Link ports.

A combination of processors and accelerators is proposed by Claus et al. [26]. These systems achieve better results regarding speed and area/power consumption compared to architectures with only processors and no accelerators.

3.3.2. Homogeneous FPGA-Based Systems. While most of today's MPSoC systems are heterogeneous (the same occurred in the case of FPGA-based multiprocessor systems) in order to meet the targeted application requirements, in the near future, homogeneous multiprocessor systems may

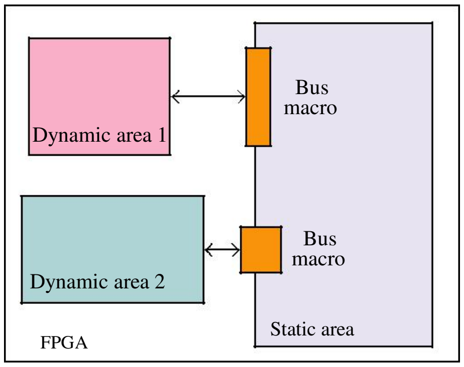

Figure 6: Architecture of a basic run-time reconfigurable system.

become a viable alternative, bringing other benefits such as run-time load balancing and task migration [16].

The homogeneous architectural style is used generally for data-parallel systems. Wireless base stations, in which the same algorithm is applied to several independent data streams, are one example; motion estimation, in which different parts of the image can be treated separately, is another. Normally, homogeneous multiprocessor systems are general purpose. In Table 4, a number of homogeneous MPoPC systems are summarized.

MPLEM [20] is a homogeneous, general-purpose 80-processor FPGA-based multiprocessor system.

Tseng and Chen [21] present a shared-bus/shared-memory multiprocessor system implemented on an Altera Cyclone. They use an instruction cache and Mutex core provided by Altera for synchronization.

A NoC-based homogeneous multiprocessor system is presented in [22]. This has 24 processors, is implemented in a Xilinx Virtex-4, and is external DDR2 constrained. To increase its compatibility and to facilitate the reutilization of the architecture, the IP Open Core Protocol (OCP-IP) standard is used to connect processor elements and NoC. Network interfaces (NIs) translate OCP to the NoC protocol. In this case, every MicroBlaze has an OCP adapter.

Almeida et al. utilizes 16 spartan3 Xilinx FPGAs to validate a homogeneous distributed-memory multiprocessor system [16].

Symmetric Multiprocessors (SMPs) are the best known general purpose multiprocessor system. Huerta et al. [23] explores FPGA capabilities for building Symmetric Mul-tiprocessors systems. They implement a complete SMP system on FPGA using softcore processors. The system has centralized shared-memory architecture. The processors have no cache because of cache coherency problems. Hung et al. present a symmetric multiprocessor system with cache; they propose an HW/SW solution for the cache coherency problem [24]. An operating system with SMP functionalities is presented in [27].

Table 5 presents the area-performance relation of the main FPGA-based multiprocessor systems referenced previ-ously. It is depicted the FPGA model, the use of area (S: Slices; FF: Flip-Flops; LUT: Lookup Tables; LE: Logic Elements), the use of on-chip memory, and the maximum

Table 4: Homogeneous FPGA-Based multiprocessor systems.

Ref	# CPU	Syst Arch	Comm meth	Comm Arch
[20]	80 MB	Net	shared mem	shared-bus
[21]	4 NiosII	Net	shared mem	shared-bus
[22]	24 MB	Net	shared mem	NoC
[23]	4 MB	Net	shared mem	shared bus
[24]	*n* Nios	Net	shared mem	shared bus
[16]	16 NPU	Net	mess passing	NoC

3.3.3. Run-Time Reconfigurable Systems. The new evolution in reconfigurable multiprocessor systems is the run-time speed of the system. reconfigurable system. This type of systems adds the dynamic reconfigurability feature of FPGAs to the power of having multiple processors. It adds a new degree of freedom in the design of multiprocessor systems. This freedom allows designers to adjust system performance at run-time obtain-ing better efficiency in accordance with the application. Gohringer¨ et al. propose a new taxonomy for reconfigurable multiprocessor systems [29]. This is an extension of the taxonomy proposed and widely accepted by Flynn at 1966 [28] (see Figure 7). The new taxonomy is depicted in Figure 8. The superclass of this classification is the RAMPSoC [30]. Here, they propose a "meet-in-the-middle" design flow, which is to combine traditional top-down design with bottom-up approach. The bottom-up design is possible due to run-time reconfigurability. It allows hardware to be re-designed at run-time in order to obtain better efficiency in terms of speed, area, and power for a specific application.

Hubner et al. presents some pioneering work in [31], where run-time reconfigurability is proposed in multipro-cessors systems at an early stage. Depending on the context, a different algorithm will be re-configured. They utilize the Dynamic Partial reconfigurable FPGA feature.

IV. Design Challenges

The principal limitation of building a multiprocessor system on a FPGA is the amount of logic resources, specifically the amount of on-chip memory. Therefore, Shared-Memory architectures are widely used. It is possible to share data memory and/or instruction memory. The study by Kulmala et al. [32] examines the efficiency of sharing instruction memory.

Another solution is to use external memory to increase the amount of memory. In this case, the number of external memory blocks is limited by the package and pins of the FPGA [20, 22]. If the multiprocessor system does not fit in a single FPGA, it is possible to implement the system using a multi-FPGA approach [33].

An important issue in shared-memory multiprocessor systems is synchronization. The cores normally used in stan-dard FPGA tool chains do not support atomic instructions and rarely support advanced synchronization mechanisms. For instance, the memory sharing mechanism (Mutex Core) provided by Altera is not efficient for multiprocessor systems [21]. Researchers present a number of ad-hoc solutions to resolve this problem. The GALS approach is adopted by

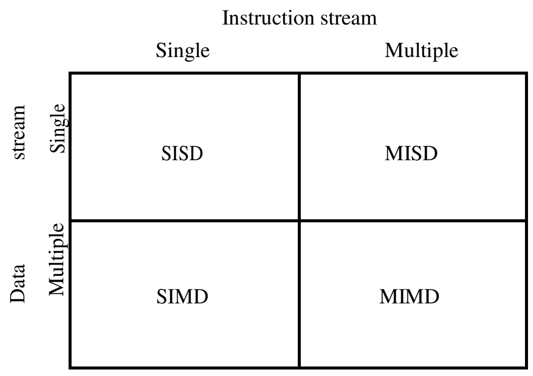

FIGURE 7: Flynn's taxonomy [28].

using the Bisynchronization method in [22]. Two hardware IP cores to perform lock and barrier functions are presented in [34]. Huerta et al. present an HW Mutex IP solution in [23].

Cache coherency is another critical point in FPGA-based multiprocessor design. Normally, the simple softcores used in such systems do not support any mechanism to guarantee cache coherency. Hung et al. propose an ad-hoc HW/SW solution [24].

Efficient on-chip communications between different cores is required to accommodate the increasing number of processors in FPGA-based multiprocessors. Traditional bus-share approach is not bandwidth efficient. Network-on-chip seems to be the best solution to interconnect cores [35, 36]. There are several papers that examine NoC in FPGA-based multiprocessor systems. The NoC approach is efficient compared to the shared-bus approach when the number of cores and data size increases [37, 38]. In [39], resource-efficient communications architecture is presented. Complete NoC-based multiprocessor systems implemented in a FPGA are presented in [22, 40]. Heterogeneous IP Block Interconnection (HIBI) is an approach designed and used by Salminen et al. [41]. A Multistage Interconnection Network (MIN) solution is presented in [42]. Gohringer¨ et al. present a thorough study of different reconfigurable communication infrastructures targeting FPGA-based mul-tiprocessors systems. They conclude that none of these fulfils all the goals required for use in FPGA-based systems for high-performance computing and present a new model: the star-wheels network on chip. This is a combination of the packet-switching and circuit-switching protocol [43].

Another important research area is parallel software efficiency. To maximize the potential of multiprocessors, it is necessary to develop parallel programming. In [44], Tumeo et al. present a tool to test and validate pipeline applications.

V. DESIGN METHODOLOGY

The main two ways of building multiprocessor systems in an FPGA are: (a) manually, using the design flow provided by FPGA companies or (b) automatically, mapping applications to a specific architecture.

5.1. Hand Tuned Design. The first and most common method is manual design using the design flow available from FPGA companies. The two most commonly-used tools are EDK from Xilinx and SOPC Builder from Altera. Their advantage is the familiarity designers have with these tools and their ease of use [10]. The problem is that it is a hand-tuned method so it is difficult to explore all the design space manually to get the best possible architecture for a specific application [8]. This method may be a good choice to build small systems where the architecture to be implemented is known, or where there are not so many possibilities for Design Space Exploration (DSE), so in these cases it is not difficult to perform experiments and then choose the best configuration.

Another important issue is that, with FPGA company design flows, it is not possible to explore all the possibilities for designing multiprocessor systems. Normally, these tools are made to design uniprocessor systems so they have a number of shortcomings when dealing with multiprocessor systems. The most important shortcomings are as follows.

(i) Limitations in architectures for interprocessor com-munication. The main design flows only allow shared-bus and point-to-point communications. Complex MPSoC may require a higher bandwidth than a bus can offer or may need to be more area-efficient than point-to-point connections. Bafumba-Lokilo et al. [38] present a generic cross-bar network-on-chip for FPGA MPSoCs, and suggest that FPGA manufacturers should integrate such technology in their standard development flow.

(ii) Lack of effective mechanisms to share resources. Mutex Core provided by FPGA vendors seems to be inefficient as a synchronization method for multipro-cessor systems [21].

(iii) Limitation in IP cores. The cores that can be included in one's design are restricted by the vendor library.

Researchers try to solve these limitations by creating *ad-hoc* IP blocks. These IP blocks act like add-ons to resolve this specific limitation. Some examples are

(i) Efficient Synchronization. Tumeo et al. [34] intro-duced two hardware synchronization modules for Xilinx MicroBlaze Systems. The modules can be completely integrated in the system using the EDK tool chain.

(ii) Cache Coherency Problem. A cache coherency mod-ule is presented in [24] as an IP Block. This paper assesses the solution and reports good results in performance.

(iii) An EDK-based tool is presented in [40]. It allows the designer to abstract low-level details of EDK and to create NoC-based systems rapidly.

(iv) Tumeo et al. also present a number of interesting modules to improve the performance of multipro-cessor systems on FPGA. An Interrupt controller is presented in [45], and a DMA mechanism in [46].

Table 5: Area-performance relation.

Ref	FPGA	#proc	Slices/LE	%	BRam	%	MHz
[7]	Virtex II Pro 50	14 MB	11250 S	48	454 KB	87	100
[10]	Virtex II Pro 30	2 MB	6434 FF 9675 LUT	23 35	80 KB	60	100
[11]	Statrix EP1S40	1 N, 3 NII	20000 LE	50	314 KB	75	70
[12]	Virtex II Pro 50	15 MB	22500 S	96	352 KB	67	80
[4]	Cyclone EP1C20	3 NII	9000 LE	45	—	—	50
[13]	Virtex II Pro 30	4 MB	—	—	—	—	50
[14]	Statrix II 2S60	15 NII	30000 LE	50	—	—	100
[15]	Virtex II Pro 30	3 MB	5268 S	38	108 KB	35	—
[22]	FX-140	24 MB	55266	87	384 KB	69	

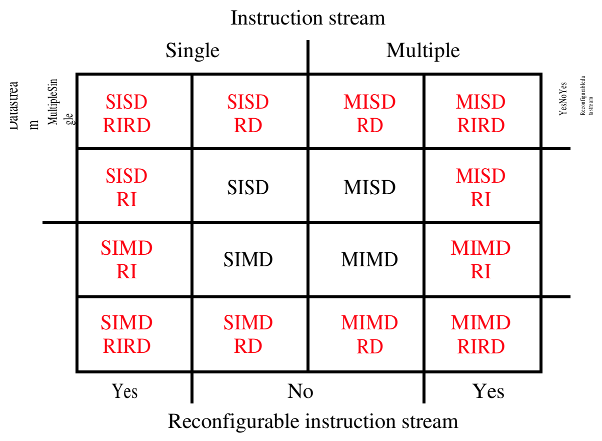

Figure 8: New taxonomy for reconfigurable FPGA-based systems proposed by Gohringer¨ et al. [29].

It may be concluded that Xilinx EDK is the most commonly used design flow [7, 10, 12, 13, 15, 20, 22, 23, 43], possibly due to the fact that Xilinx is the largest FPGA vendor. However, there are also several Multiprocessor systems using Altera SOPC Builder design flow [4, 11, 14, 21, 24].

5.2. Automatic Synthesis Design. Another trend consists of using automatic synthesis tools to map an application to an architecture. The input parameters are: the application and, sometimes, the platform for the architecture and the cores to be used in the design. The result is a synthesized system that fits in the FPGA target.

The problem with this method is that there are no standards. Some automatic tools have appeared in the research community, but they have not been standardized or commercialized. So, they are not widely used. Nevertheless, this is a very important research area, but is still on-going. Several papers present frameworks to perform this automation process; some of which we discuss here.

A promising possibility to increase designer productivity by automating the simultaneous design tasks of application mappings and IP selection is presented in [47]. Automation is possible because they have found a Solvable Integer Linear Programming (ILP) model that captures all the necessary design trade-off parameters of such systems.

Jin et al. use ILP [8] to solve the exploration problem. They propose an automated framework to assist the designer in exploring the design space (DSE) of soft multiprocessor microarchitectures. The objective is to identify the best multiprocessor on the FPGA for a target application and optimally map the application tasks and communications links to this micro-architecture. The framework proposed is evaluated improving the hand-tuned design presented in [7]. Therefore, DSE automation is important and becomes more critical when the system to be designed consists of many cores, due to in the fact that it offers greater possibilities for architecture configuration.

ESPAM [48] is a tool for automated design, programming, and implementation of multiprocessor systems on FPGAs implemented by Nikolov et al. While state-of-the-art development tools only support shared-bus architectures and point to point, ESPAM is general enough to implement multiprocessor systems with different communications topologies. ESPAM allows automated multiprocessor sys-tems to be programmed in a way which significantly reduces the design time. It uses Kahn Process Networks (KPN) to specify the application.

MAMPS [49] presents a framework similar to ESPAM but with significant improvements. The framework generates application-specific architecture from the description of the applications. This time Kumar et al. use SDF to specify the application. This is the first flow that allows mapping of multiple applications on a single platform. The flow allows the designers to traverse the design space quickly.

Both ESPAM and MAPS are limited by FPGA hardware resources. The amount of on-chip memory is the main limiting factor. It limits the size of the multiprocessor system in ESPAM and the number of applications that can be implemented in MAMPS.

VI. Conclusion

This paper surveys recent FPGA-based multiprocessor sys-tems appearing in the literature. There has been a significant increase in publications in this area over the last three years.

After revising the literature, we can draw the conclusion that one of the most important issues with regard to designing multiprocessor systems in FPGA is the use of block RAM. The amount of on-chip RAM is fixed, and it limits the number of processors that can be included in one's design more than logic resources. Therefore, it is important to design memory-efficient systems. When building shared-memory systems, cache coherency and synchronization are critical aspects because the simple softcore processors offered by FPGA vendors have certain shortcomings in these areas. NoC is a weak point in design flows provided by vendors, since they do not include this approach in their tools. In our opinion, it is only a matter of time before they do so. Other important challenges in multiprocessor design are parallel software and automation of the design process. There are a number of important advances in this area but no standards have been established as yet. Run-time reconfigurability uses the dynamic reconfiguration feature of FPGAs to obtain a new degree of freedom in the design of multiprocessor systems, making these systems more flexible to target different applications using the same hardware.

REFERENCES

[1] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 10, pp. 1701–1713, 2008.

[2] W. Wolf, "The future of multiprocessor systems-on-chips," in Proceedings of the 41st Design Automation Conference, pp. 681–685, ACM, New York, NY, USA, June 2004.

[3] G. Martin, "Overview of the MPSoC design challenge," in Proceedings of the 43rd ACM/IEEE Design Automation Conference, pp. 274–279, 2006.

[4] R. Joost and R. Salomon, "Advantages of FPGA-based mul-tiprocessor systems in industrial applications," in Proceedings of the 31st Annual Conference of the IEEE Industrial Electronics Society (IECON '05), pp. 4451–4506, November 2005.

[5] C. Wright and M. Arens, "Fpga-based system-on-module approach cuts time to market, avoids obsolescence," FPGA and Programmable Logic Journal, vol. 6, no. 6, 2005.

[6] S. Raje, "Catching the FPGA productivity wave," Electronic Design, vol. 52, no. 24, p. 20, 2004.

[7] K. Ravindran, N. Satish, Y. Jin, and K. Keutzer, "An FPGA-based soft multiprocessor system for IPV4 packet forward-ing," in Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05), pp. 487–492, August 2005.

[8] Y. Jin, N. Satish, K. Ravindran, and K. Keutzer, "An automated exploration framework for FPGA-based soft multiprocessor systems," in Proceedings of the 3rd IEEE/ACM/IFIP Interna-tional Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS '05), pp. 273–278, ACM, Jersey City, NJ, USA, September 2005.

[9] P. Huerta, J. Castillo, J. I. Martinez,´ and V. Lopez,´ "A microblaze based multiprocessor SoC," WSEAS Transactions on Circuits and Systems, vol. 4, no. 5, pp. 423–430, 2005.

[10] J. Dykes, P. Chan, G. Chapman, and L. Shannon, "A multiprocessor system-on-chip implementation of a laser-based transparency meter on an FPGA," in Proceedings of the International Conference on Field Programmable Technology (ICFPT '07), pp. 373–376, December 2007.

[11] O. Lehtoranta, E. Salminen, A. Kulmala, M. Hannikainen,¨¨ and T. D. Ham¨al¨ainen,¨ "A parallel MPEG-4 encoder for FPGA based multiprocessor SOC," in Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05), pp. 380–385, August 2005.

[12] R. K. Karanam, A. Ravindran, and A. Mukherjee, "A stream chipmultiprocessor for bioinformatics," SIGARCH Computer Architecture News, vol. 36, no. 2, pp. 2–9, 2008.

[13] A. Tumeo, M. Branca, L. Camerini et al., "A dual-priority real-time multiprocessor system on FPGA for automotive applications," in Proceedings of the Conference on Design, Automation and Test in Europe (DATE '08), pp. 1039–1044, ACM, New York, NY, USA, March 2008.

[14] J. Khan, S. Niar, A. Menhaj, Y. Elhillali, and J. L. Dekeyser, "An MPSoC architecture for the multiple target tracking application in driver assistant system," in Proceedings of the 19th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '08), pp. 126–131, July 2008.

[15] S. Ben Othman, A. K. Ben Salem, and S. Ben Saoud, "MPSoC design of RT control applications based on FPGA SoftCore processors," in Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems (ICECS '08), pp. 404–409, September 2008.

[16] G. M. Almeida, G. Sassatelli, and P. Benoit, "An adaptive message passing mpsoc framework," International Journal of Reconfigurable Computing, vol. 2009, Article ID 242981, 20 pages, 2009.

[17] B. Nichols, D. Buttlar, and J. P. Farrell, Pthreads Programming, O'Reilly & Associates, Sebastopol, Calif, USA, 1996.

[18] "The openmp api specification for parallel progremming," http://openmp.org/wp.

[19] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable Parallel Programming with the Message Passing Interface, MIT Press, Cambridge, Mass, USA, 1994.

[20] G.-G. Mplemenos and I. Papaefstathiou, "MPLEM: an 80-processor FPGA based multiprocessor system," in Proceedings of the 16th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '08), pp. 273–274, April 2008.

[21] C.-Y. Tseng and Y.-C. Chen, "Design and implementation of multiprocessor system on a chip (mpsoc) based on fpga," in Proceedings of the International Computer Symposium (ICS '09), 2009.

[22] Z. Wang and O. Hammami, "External DDR2-constrained NOC-based 24-processors MPSOC design and implementa-tion on single FPGA," in Proceedings of the 3rd International Design and Test Workshop (IDT '08), pp. 193–197, December 2008.

[23] P. Huerta, J. Castillo, J. I. Mart´ınez, and C. Pedraza, "Exploring FPGA capabilities for building symmetric multiprocessor systems," in Proceedings of the 3rd Southern Conference on Programmable Logic (SPL '07), pp. 113–118, February 2007.

[24] A. Hung, W. Bishop, and A. Kennings, "Symmetric multi-processing on programmable chips made easy," in Proceedings of the Conference on Design, Automation and Test in Europe (DATE '05), vol. 1, pp. 240–245, March 2005.

[25] T. Oliver, B. Schmidt, D. Maskell, D. Nathan, and R. Clemens, "Multiple sequence alignment on an FPGA," in Proceedings of the 11th International Conference on Parallel and Distributed Systems Workshops (ICPADS '05), vol. 2, pp. 326–330, IEEE Computer Society, Washington, DC, USA, July 2005.

[26] C. Claus, W. Stechele, and A. Herkersdorf, "Autovision: a run-time reconfigurable mpsoc architecture for future driver assistance systems," Information Technology Journal, vol. 49, no. 3, pp. 181–187, 2007.

[27] P. Huerta, J. Castillo, C. Sanchez,´ and J. I. Mart´ınez, "Oper-ating system for symmetric multiprocessors on FPGA," in Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08), pp. 157–162, Decem-ber 2008.

[28] M. Flynn, "Very high-speed computing systems," Proceedings of the IEEE, vol. 54, no. 12, pp. 1901–1909, 1966.

[29] D. Gohringer,¨ T. Perschke, M. Hubner,¨ and J. Becker, "A taxonomy of reconfigurable single-/multiprocessor systems-on-chip," International Journal of Reconfigurable Computing, vol. 2009, Article ID 395018, 11 pages, 2009.

[30] D. Gohringer,¨ M. Hubner,¨ T. Perschke, and J. Becker, "New dimensions for multiprocessor architectures: on demand heterogeneity, infrastructure and performance through reconfigurability—rhe RAMPSoC approach," in Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '08), pp. 495–498, September 2008.

[31] M. Hubner, K. Paulsson, and J. Becker, "Parallel and flexi-ble multi-processor system-on-chip for adaptive automotive applications based on xilinx microblaze soft-cores," in Pro-ceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, p. 149a, April 2005.

[32] A. Kulmala, E. Salminen, and T. D. Ham¨al¨ainen,¨ "Instruction memory architecture evaluation on multiprocessor FPGA MPEG-4 encoder," in Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '07), pp. 1–6, April 2007.

[33] A. Kulmala, E. Salminen, and T. D. Ham¨al¨ainen,¨ "Evaluating large system-on-chip on multi-FPGA platform," in Proceed-ings of the 7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, vol. 4599 of Lecture Notes in Computer Science, pp. 179–189, 2007.

[34] A. Tumeo, C. Pilato, G. Palermo, F. Ferrandi, and D. Sciuto, "HW/SW methodologies for synchronization in FPGA," in Proceedings of the 7th ACM SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '09), pp. 265–268, ACM, New York, NY, USA, February 2009.

[35] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," Computer, vol. 35, no. 1, pp. 70–78, 2002.

[36] S. Kumar, A. Jantsch, M. Millberg et al., "A network on chip architecture and design methodology," in Proceedings of the IEEE Computer Society Annual Symposium on VLSI, p. 117, 2002.

[37] H. C. Freitas, D. M. Colombo, F. L. Kastensmidt, and P. O. A. Navaux, "Evaluating network-on-chip for homogeneous embedded multiprocessors in FPGAs," in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07), pp. 3776–3779, May 2007.

[38] D. Bafumba-Lokilo, Y. Savaria, and J.-P. David, "Generic crossbar network on chip for FPGA MPSoCs," in Proceedings of the Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA '08), pp. 269–272, June 2008.

[39] X. Wang and S. Thota, "Design and implementation of a resource-efficient communication architecture for multipro-cessors on FPGAs," in Proceedings of the International Confer-ence on Reconfigurable Computing and FPGAs (ReConFig '08), pp. 25–30, December 2008.

[40] S. Lukovic and L. Fiorin, "An automated design flow for NoC-based MPSoCs on FPGA," in Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping (RSP '08), pp. 58–64, June 2008.

[41] E. Salminen, A. Kulmala, and T. D. Ham¨al¨ainen,¨ "HIBI-based multiprocessor soc on FPGA," in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05), vol. 4, pp. 3351–3354, May 2005.

[42] B. Neji, Y. Aydi, R. Ben-atitallah, S. Meftaly, M. Abid, and J.-L. Dykeyser, "Multistage interconnection network for MPSoC: performances study and prototyping on FPGA," in Proceedings of the 3rd International Design and Test Workshop (IDT '08), pp. 11–16, December 2008.

[43] D. Gohringer,¨ B. Liu, M. Hubner,¨ and J. Becker, "Star-wheels network-on-chip featuring a self-adaptive mixed topology and a synergy of a circuit- and a packet-switching communication protocol," in Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09), pp. 320–325, September 2009.

[44] A. Tumeo, M. Branca, L. Camerini et al., "Prototyping pipelined applications on a heterogeneous FPGA multipro-cessor virtual platform," in Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '09), pp. 317– 322, January 2009.

[45] A. Tumeo, M. Branca, L. Camerini et al., "An interrupt controller for FPGA-based multiprocessors," in Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS '07), pp. 82–87, July 2007.

[46] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, "Lightweight DMA management mechanisms for multiprocessors on FPGA," in Proceedings of the 19th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '08), pp. 275–280, July 2008.

[47] H. Ishebabi, P. Mahr, C. Bobda, M. Gebser, and T. Schaub, "Answer set versus integer linear programming for automatic synthesis of multiprocessor systems from real-time parallel programs," International Journal of Reconfigurable Computing, vol. 2009, Article ID 863630, 11 pages, 2009.

[48] H. Nikolov, T. Stefanov, and E. Deprettere, "Efficient auto-mated synthesis, programing, and implementation of multi-processor platforms on FPGA chips," in Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '06), pp. 1–6, August 2006.

[49] A. Kumar, S. Fernando, Y. Ha, B. Mesman, and H. Corporaal, "Multiprocessor systems synthesis for multiple use-cases of multiple applications on FPGA," ACM Transactions on Design Automation of Electronic Systems, vol. 13, no. 3, pp. 1–27, 2008.