Regular Article

A Survey on Reconfigurable System-on-Chips

Hung Kiem Nguyen, Thanh-Vu Le-Van, Xuan-Tu Tran

VNU Key Laboratory for Smart Integrated Systems (SISLAB), University of Engineering and Technology (VNU-UET), Vietnam National University, Hanoi (VNU), Vietnam

Correspondence: Hung Kiem Nguyen, kiemhung@vnu.edu.vn

Communication: received 21 November 2016, revised 21 August 2017, accepted 27 September 2017
Online publication: 7 March 2018, Digital Object Identifier: 10.21553/rev-jec.147
The associate editor coordinating the review of this article and recommending it for publication was Dr. Bui Trong Tu.

Abstract– **The requirements for high performance and low power consumption are becoming more and more inevitable when designing modern embedded systems, especially for the next generation of multi-mode multimedia or communication standards. Ultra large-scale integration reconfigurable System-on-Chips (SoCs) have been proposed to achieve not only better performance and lower energy consumption but also higher flexibility and versatility in comparison with the conventional architectures. The unique characteristic of such systems is the integration of many types of heterogeneous reconfigurable processing fabrics based on a Network-on-Chip. This paper analyzes and emphasizes the key research trends of reconfigurable System-on-Chips (SoCs). Firstly, the emerging hardware architecture of SoCs is highlighted. Afterwards, the key issues of designing reconfigurable SoCs are discussed, with the focus on the challenges when designing reconfigurable hardware fabrics and reconfigurable Network-on-Chips. Finally, some state-of-the-art reconfigurable SoCs are briefly discussed.**

Keywords– **Reconfigurable computing, Reconfigurable NoC, Network-on-Chip, System-on-Chip, reconfigurable processing fabrics.**

1 Introduction

Emerging System-on-Chips (SoCs), especially those for mobile systems, are typically battery-powered systems and have to support a wide range of applications such as multimedia processing or communication baseband processing. The high performance and low power consumption are becoming more and more inevitable requirements when designing such devices, especially for next generation multi-mode multimedia or communication standards. New chip architectures should be able to simultaneously support multiple applications with a high performance, while maintaining low power consumption, low area, low non-recurring engineering (NRE) cost, and shorter time-to-market, as well as offer the capability of hardware updating after systems have been deployed. In addition, these architectures should be able to resolve the appearance of defects or faults in the system in order to guarantee the correct operation of the system.

Reconfigurable systems-on-chips were proposed to meet the above requirements [1]. Such systems usually consist of several types of processing elements, such as software-programmable processors, application-specific IP cores, and especially reconfigurable processing fabrics. For connecting components of the system together, reconfigurable Network-on-chips (NoCs) [2, 3] have been proposed to achieve not only better performance and lower energy consumption but also higher flexibility in comparison with conventional on-chip bus architectures. Two key components that makes reconfigurability of a reconfigurable system are the reconfigurable processing fabrics and the reconfigurable NoC. The reconfigurable processing fabrics are organized into arrays of reconfigurable processing units (RPU) which is useful for implementing computation-intensive functions. The reconfigurable NoC is the on-chip interconnection fabric that can autonomously adapt their structure and behavior to various contexts at runtime. Reconfigurability allows the prefabricated integrated circuit of these components to physically alter the location or functionality of its infrastructure to become more than one kind of digital circuit. Reconfiguration can be carried out at either run-time (i.e. dynamic reconfiguration) or compile-time (i.e. static reconfiguration) [4]. Static reconfiguration is the common way for implementing applications with FPGA (Field-programmable Gate Arrays)-like reconfigurable logic fabric. Dynamic reconfiguration uses a schedule that can re-allocate hardware resources to various tasks or applications at runtime. Thanks to many functions are accelerated by being performed on the highly optimized processing and communication hardware fabrics during the operation of the system, the system's performance is increased. Moreover, the system's flexibility is also maintained while its functional density (i.e. number of applications mapped onto an area unit) is increased.

The reconfigurable SoC is a very wide topic having various other sub-domains, such as coarse-grained reconfigurable fabrics (e.g. [5–7] and [8]), reconfigurable on-chip communication fabrics (e.g. [9, 10] and [11]), fault tolerance design (e.g. [12, 13] and [14]), design methodologies ([15, 16] and [17]), operation systems,

programming models [18], mapping methods ([19–21]) and simulators, etc. However, this survey does not encompass any one of the mentioned domains because most of them already have been discussed in detail in the various past studies. The goal of this paper is to provide an interdisciplinary introduction to reconfigurable system-on-chips. The focus is on breadth to provide a unified view of this topic. The paper begins with an overview of the evolution of the system-on-chips. An introduction to the emerging architecture of reconfigurable SoCs is presented next, which is followed by a description of their two key components: reconfigurable processing fabric and reconfigurable NoC. Trends and future research directions in reconfigurable SoCs are also discussed. The paper is finished by listing various the state-of-the-art reconfigurable SoCs. The rest of the paper is organized as follows. The evolution of System-on-Chips is presented in Section 2. In Section 3, the emerging architecture of reconfigurable SoCs is introduced. Section 4 and Section 5 give the survey of reconfigurable processing fabrics and reconfigurable NoCs. Some state-of-the-art proposals for reconfigurable SoCs are described in Section 6. Finally, some conclusions are drawn in Section 7.

2 Evolution of System-on-Chips

Early SoCs often include only one processor in their structure. Unfortunately, a single processor no longer deals with the requirements of the applications those demand more and more parallelism and real-time constraints. To tackle that challenge, the modern embedded system is now designed with many processors together with other heterogeneous components integrated in a chip called Multiprocessor System-on-Chip (MPSoC).

Generally, MPSoCs offer a significantly better performance-to-cost ratio than those uniprocessor systems do. The basic reason is that the cost of processing element is a nonlinear function of performance [22]. The cost of a processor increases greatly as the clock frequency increases. On the other hand, the clock rate of the processor cannot be increased infinitely due to the reason of power efficiency and physical limitations. Therefore, partitioning the application so that it can be executed on several smaller processors in a parallel or pipelining fashion is more efficient. Moreover, using multiple processors can also make it possible to deal with real-time performance. It is often much easier to meet deadlines when those time-critical processes are mapped onto separate processors. Some research also shows that multiprocessors help to reduce power consumption. The reason is that several processors running at slow clock rates consume less power than a single large processor running at very high clock frequency [23]. However, the barrier to the development of MPSoCs is the on-chip communication. Strictly speaking, the on-chip communication nowadays defines the performance of MPSoCs [24]. Because of the limitations of bus-based architectures, bus-based SoCs cannot provide enough processing performance to keep up with this new technology trends. The Network-on-Chip paradigm [3] was proposed and considered as an outstanding alternative for the bus-based architecture. NoC architectures provide a higher communication performance and therefore it is a promising solution for integrating a large number of IP (Intellectual Property) cores for implementing SoCs with over one billion transistors in the future. NoCs have inherited the latest technologies of bus architectures in terms of advanced protocols and topology. Moreover, by adapting packet-based communication mechanism to the on-chip domain, NoCs can deal with many critical issues related to the interconnection fabric design. A NoC-based System-on-Chip is viewed as a micro-network of on chip components and must satisfy constraints on the implementation cost (i.e. power consumption, area, and efficient usage of resources) and quality-of-service (QoS) (i.e. reliability, throughput, and latency) under the conditions of intrinsically unreliable on-chip signal transmission.

On the other hand, the emerging embedded systems are often not only designed specifically for one application, but also designed to support multiple applications on the same hardware platform. In the other words, we desire that a System-on-Chip platform can be used in a wide range of different applications. Such sharing resources among several applications makes the system cheaper and more versatile. Besides high performance, low power consumption and cost, it also needs flexibility to easily upgrade without replacing the system.

Figure 1 depicts several architectures with their variability in terms of flexibility, power efficiency and performance. The instruction-driven processors are very flexible and usually targets at the applications at where flexibility has a higher priority than implementation efficiency. Typical representatives of processors include general-purpose processors (GPP), digital signal processors (DSP), application specific instruction-set processor (ASIP), and embedded processors, etc. GPPs aim at the general-purpose applications, so they normally offer a very high flexibility at the expense of low performance, high power consumption, and high cost. Consequently, GPPs are unsuitable for embedded systems. ASIPs, DSPs, and embedded processors are an intermediate solution between GPPs and ASICs that target at a specific application domain. Therefore, the implementation of these processors focuses on power efficiency and high performance with good flexibility. To enhance the computing performance while reducing power consumption, many improvements in the execution model for the processors have been proposed such as superscalar, VLIW (Very Long Instruction Word), SIMD (Single Instruction - Multiple Data), MIMD (Multiple Instruction - Multiple Data), etc. The processors also have constantly been advanced either their operating frequency or the number of integrated cores. However, basically these processors still cannot solve the root problems in the instruction-driven execution method. In contrast, the dataflow-driven processing method of the ASICs (Application-Specific Integrated Circuits) usually offers the optimized performance and

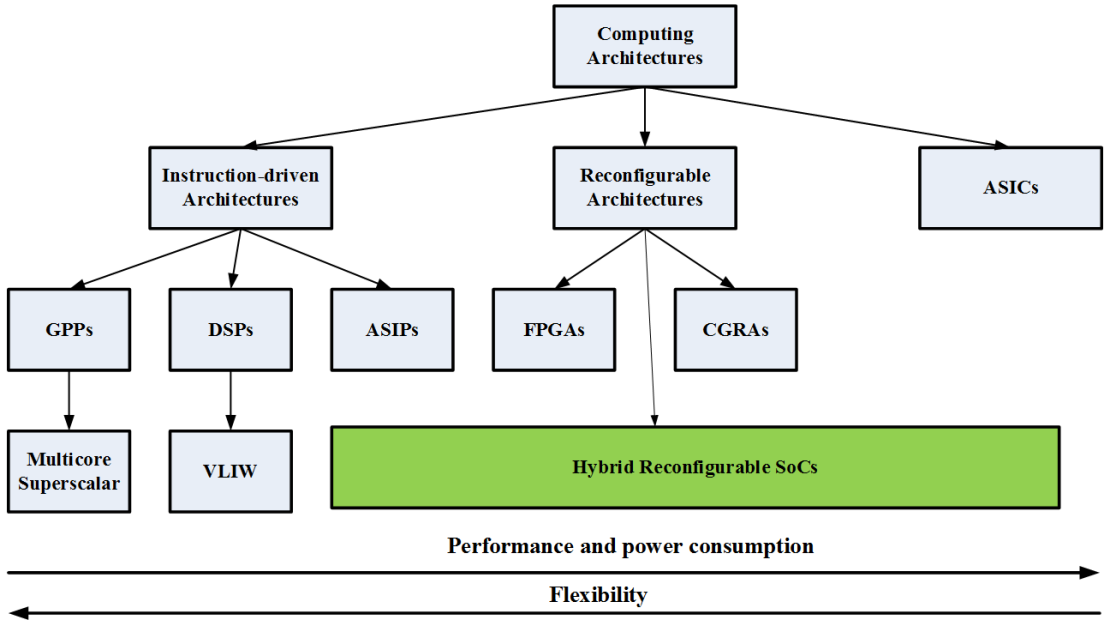


Figure 1. Comparison between computing architectures.

power consumption. However, the drawback of the ASICs is low flexibility. Moreover, the high price for designing and manufacturing the chip masks is becoming increasingly an important factor that limits the application scope of the ASICs. These all are becoming a very large barrier to the development of dataflow-driven architectures in the future. Field Programmable Gate Arrays (FPGAs) are more flexible platforms due to their reconfigurable possibilities, but are slower in comparison with the ASICs. In addition, the FPGAs consume a vast amount of energy for routing resources making them unsuitable for mobile devices. Unlike the FPGAs, Coarse-Grained Reconfigurable Arrays (CGRAs) often have a good balance between implementation efficiency and flexibility. However, the CGRA does not support bit-level computation and control-intensive function due to its coarse granularity.

Recently, the research trend has shifted toward the hybrid reconfigurable System-on-Chips that integrate all of the above architectures into a single system (e.g. [5, 6] and [7]). By mapping the kernel functions of an application onto the reconfigurable fabrics, these systems can achieve high performance approximately equivalent to that of ASIC while maintaining a degree of flexibility close to that of DSP processors (e.g. [5, 6, 25–27] and [7]). The tasks performed by the reconfigurable hardware are usually known as hardware tasks. By dynamically reconfiguring, these systems allow many hardware tasks to be mapped onto the same hardware platform, thus reducing the area and power consumption of the design [4]. In a dynamically reconfigurable system, the sequence of computation and configuration is just determined and handled at run-time. Run-time reconfiguration is a complex process that must deal with side-effect factors such as resource management and synchronization between the placed tasks. This process is done by the scheduler that may be implemented either as a part of the software running on the processor (e.g. [28]) or as a dedicated hardware (e.g. [29]). The scheduler is in charge of managing the tasks and deciding when a task will be executed. The scheduler then tries to place the task on the reconfigurable architecture by downloading from the configuration memory a set of corresponding configuration bits, so-called configuration context. The configuration context determines the functionality of resources on the reconfigurable hardware for executing the given task. Besides, since the same SoC platform is to be used for several applications that have very different requirements, the on-chip communication infrastructure must be very flexible to support a wide range of bandwidth and Quality-of-Service (QoS) requirements. Currently, such flexibility can be provided by a large packet-switched NoC with an over-engineered total bandwidth. Unfortunately, such a NoC would result in a significant area overhead because only a fraction of its bandwidth is really utilized for a given application ([17, 30, 31]). Another way is the reconfigurable NoC that enhanced the dynamical reconfigurability of the SoC platform; therefore, this also results in the further reduction in area and power consumption of the design ([17, 30]).

Several design methodologies have been proposed to deal with the hybrid reconfigurable SoCs and can be classified into two main categories: compile-time methodologies and run-time methodologies. Compile-time methodologies (e.g. [32, 33]) are generally aimed at designing the SoCs whose configuration contexts are generated by a cross-compiler and must be downloaded into the system after right power-on. Unfortunately, such SoCs are not flexible enough to support dynamic environments where the system's characteristics and behavior need to adapt strongly to various contexts at runtime. Recently, researchers have been paying a lot of attention to the development of run-time methodologies for self-aware reconfigurable SoCs (e.g. [7, 10]

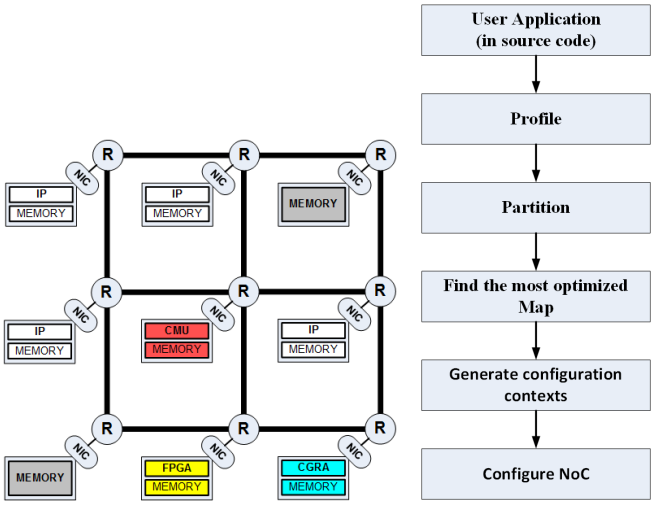
and [11]). These methodologies focus on the techniques that allow SoCs to monitor its own state as well as external environment and then autonomously adapt their structure and behavior during the period of their operation. The elements of the SoC that can be configured at run-time include both computation fabrics and interconnection fabrics.

3 Architecture of Reconfigurable SoCs

A reconfigurable SoC for modern embedded systems are usually integrated many heterogeneous processing resources such as software programmable processors, hardwired IP cores, reconfigurable infrastructure, etc. based on a flexible communication infrastructure.

Generally, such a system usually includes:

- **Computation Resources** provide the data processing and storage functionality. The computation resources can be either heterogeneous or homogeneous. However, according to International Technology Roadmap for Semiconductors (ITRS) report [34], heterogeneous integration will be the dominated trend in the future. The computation resource comprises of:
  - Instruction-driven processors such as GPPs, DSPs, ASIPs, etc.
  - Reconfigurable processing fabrics that are usually arrays of the repeated logic cells. These logic cells vary in complexity from very small and simple structures as the look-up tables in FPGAs to more complex ones as ALUs in CGRAs.
  - Fixed functional accelerators such as H.264 encoder, MP3 decoder, etc.
  - Memory subsystem includes storage units and direct memory access controllers.
- **Communication infrastructure** provides the communication medium between the computation resources. Most of existing reconfigurable SoCs (e.g. [5, 32, 33, 35] and [29]) are based on shared-bus or crossbar interconnection infrastructures. However, some NoC-based reconfigurable SoCs (e.g. [6] and [7]) are also emerged. While on-chip integration is more and more increasing, the Network-on-Chip will be the most dominated communication paradigm for Ultra Large-Scale Integration SoCs in future [34]. The communication among the cores on the NoC is carried out by sending the packets on a path composed of routers and inter-router links. The communication infrastructure in a NoC architecture comprises of:
  - Routers (or Switches) which direct the data according to the selected protocol,
  - Network adapter/interface (NI) which provides a bridge between the routers and the processing elements attached to them. Its main function is to separate computation part (e.g. PEs) from the communication part (e.g. routers),
  - Links which are physical channels for transferring data between network routers,
  - Topology is the paradigm in which network routers are organized and connected together.
- **Management mechanism** includes strategies for:
  - Data-flow and Context-flow control;
  - Dynamically reconfiguring NoC (e.g. placing, routing, and scheduling applications) on-demand at run-time.
- **Mapping Method** answers the question how to find an optimal map for an application onto the SoC platform.



(a) Top-level architecture (b) Flowchart of the management program

Figure 2. A typical NoC-based SoC.

To program such a system, an application is first represented intermediately as a series of tasks that depends on each other by Control and Data Flow Graph (CDFG) [36], and then partitioned and mapped onto the heterogeneous computational and routing resources. Especially, computation-intensive kernel functions of the application are mapped onto the reconfigurable hardware so that they can achieve high performance like that achieved when implementing these functions by a dedicated hardware. By dynamically reconfiguring, reconfigurable computing systems allow many hardware tasks to be mapped onto the same hardware platform. Moreover, reconfigurability also helps to resolve the appearance of defects/faults in the system in order to guarantee the correct operation of the system.

Figure 2 (a) depicts a typical reconfigurable NoC-based system in which each network node is a cluster of router, NI, and precessing element (e.g. IP core, memory, FPGA, CGRA). In this model, one general-purpose microprocessor plays the role of Central Managing Unit (CMU). The CMU is responsible for managing and controlling the operation of programmable/reconfigurable infrastructure on the NoC. The CMU performs the feasibility analysis, spatial mapping, temporal scheduling, generating configuration contexts and configuring the NoC before performing an application. However,

the CMU does not handle run-time scheduling for individual processes and communications in each node during execution. That is performed by the individual nodes and network routers. In other words, the NoC has two operation modes, reconfiguration and self-reconfiguration. The NoC is initialized in reconfiguration mode by the CMU. After that, NoC operates in self-reconfiguration mode that enables the NoC to adapt itself to the dynamically environment in real-time.

Figure 2 (b) shows the execution flow of the management program on the CMU. Firstly, the user application is fully executed on the CMU, and is profiled to estimate the computation complexity and memory access bandwidth of the application. Based on instruction profiling, the application is partitioned into several tasks that are suitable to available computational resources of the NoC. Next, the CMU performs run-time scheduling and maps these tasks and inter-tasks communication of the application to suitable processing elements and network links, respectively. It also tries to find the best configuration that is optimized in terms of either resources utilization or energy consumption or QoS, by using the optimization algorithms such as Particle Swarm Optimization (PSO), Evolution Algorithm (EA), Genetic Algorithm (GA), etc. Finally, the CMU generates configuration contexts and downloads them into the NoC.

Developing products and applications using reconfigurable NoC-based architectures usually offers many opportunities and challenges. The reconfigurable SoC exposed tremendous potential for enhancing the computing performance. Recently, the reconfigurable SoC has begun to be applied in some areas of high performance computing such as structural analysis, fluid dynamics, molecular modeling, bioinformatics, chemistry, geological and seismic hazards investigation, meteorology, cosmology, etc. However, there still exist many technical problems that must to be solved so that the reconfigurable SoC can be used popularly as GPP processors. In order to efficiently exploit such a reconfigurable computing system, many tools will be required for developing NoC architectures and reconfigurable architectures, as well as for mapping and scheduling of tasks into NoCs.

Reconfigurable architecture is often organized into arrays of configurable logic cells. This enables reconfigurable hardware to achieve excellent performance when implementing computation-intensive tasks, but relatively low performance when implementing control-intensive tasks. Consequently, to some extent, this limits the scope of applications of the reconfigurable hardware. New architectures should solve the problem in order to improve the performance when performing control-intensive tasks.

Mapping and scheduling tasks into NoCs must deal with many aspects of parallel computing and all its associated techniques such as computation and data partitioning, inter-process synchronization, interrupt management, and code generation, etc. (e.g. [19, 20, 37]). In the processor-based system, interrupt-driven mechanism is an effective method to communicate with peripherals. When an interrupt occurs, CPU (Central Processing Unit) has to save the context of the executing program on the stacks so that the CPU can return to the program that was interrupted. Unfortunately, this method is not suitable for reconfigurable processing systems because there are a lot of data, status information and configuration information that need to be stored and restored. Store all intermediate results and status can lead to huge time overhead and storage overhead. Therefore, the requirement on a new interrupt processing mechanism is inevitable when developing reconfigurable systems. Besides, reconfigurable systems also require the support of the operating system. Because the operating mechanism and the executing model of the reconfigurable hardware are not similar to the traditional processors, mechanism for memory management, tasks scheduling, thread/process management are all new problems must be solved.

Applying reconfigurable techniques to general-purpose computer is suitable with the development trend of computer structure, as well as with the technology trend in the integrated circuit design. This will be one of key trends of the new generation computer in the future. There are five main research issues in the field of reconfigurable SoCs design: (i) Methodology for designing reconfigurable SoCs (e.g. [15, 38] and [16]); (ii) Reconfigurable resource modeling and real-time operation system for the reconfigurable hardware (e.g. [28] and [39]); (iii) On-chip communication fabrics for reconfigurable SoC (e.g. [12–14, 30, 31, 40–64] and [65]); (iv) Fine-, and coarse-grained reconfigurable fabrics (e.g. [5, 6, 25–27, 32, 33, 35, 66–68] and [29]); and (v) Mapping method for reconfigurable architectures (e.g. [5, 36] and [69]).

Next sections will focus on survey and vision of on-chip communication fabrics, coarse-grained reconfigurable fabrics.

4 Reconfigurable Processing Farbics

The reconfigurable hardware architecture is generally classified into fine-grained reconfigurable architecture such as the Field Programmable Gate Array (FPGA) and coarse-grained dynamically reconfigurable architecture (CGRA). Some reconfigurable computing systems use a standard FPGA as a reconfigurable fabric (e.g. Zynq-7000 [32], Arria 10 [33], Warp [29] etc.), some adopt CGRA (e.g. MorphoSys [66], PACT XPP-III [67], ADRES [35], and REMUS [68] etc.), while others incorporate both (e.g. MORPHEUS [6] and [7]).

FPGAs support the fine-grained reconfigurable fabric that can operate and be configured at bit-level. FPGAs are extremely flexible due to their higher reconfigurable capability. However, the FPGAs consume more power and have more delay and area overhead due to greater quantity of routing required per configuration [1, 38]. This limits the capability to apply FPGA to mobile devices. CGRAs were proposed to overcome the limitation of conventional microprocessors and fine-grained reconfigurable devices in certain

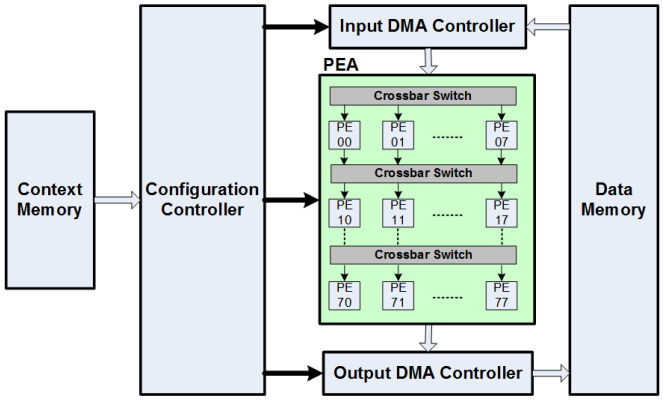


Figure 3. Basic architecture of reconfigurable hardware.

application domains such as multimedia and communication baseband processing [38]. In contrast to FPGAs, CGRAs aim at reconfiguring and manipulating on data at word-level. The CGRA can be a domain-specific system (e.g. ADRES [35], and REMUS [68] etc.) or an application-specific system (e.g. [70, 71]). The CGRA is proposed to exploit high Data-Level Parallelism (DLP), Instruction-Level Parallelism (ILP) and Task-Level Parallelism (TLP) of the computation-intensive loops of an application. The CGRA also supports the capability of dynamic reconfiguration by enabling the hardware fabrics to be reconfigured into different functions even while the system is running. By dynamically reconfiguring the hardware, many different functions are mapped to the same hardware structure, thus leading to a reduction in size, cost and power consumption of the system. Consequently, high flexibility and performance, and low power consumption of the CGRA make itself ideal to satisfy the design requirements of multimedia processing applications.

The generic architecture of the reconfigurable hardware is composed of Processing Element Array (PEA), input/output DMA controllers, Context Memory, Data Memory, and Configuration Controller as shown in Figure 3.

Configuration controller takes charge of fetching a reconfiguration context from the Context Memory, and then decoding context to configuration information that establishes the function of reconfigurable fabrics. The time needed to configure the reconfigurable fabrics is called configuration time. Minimizing the configuration time is the main objective when designing context parser. Some techniques such as compressing context, parallelizing operation of context parser with execution of PEA, etc., can help to shorten the configuration time overhead.

PEA is often organized into a regular array of configurable blocks that can be either configurable logic cells (e.g. Look-up tables in FPGAs) or configurable processing elements (e.g. PACT XPP-III [67], REMUS [68] etc.). Configurable blocks are connected together by the configurable routing network that is based on either circuit-switching technique or packet-switching technique. The important parameters of one PEA include topology, granularity of PE (e.g. 4-bit, 8-bit, or 16-bit), homogeneous or heterogeneous PEs, configuration depth, and execution model, etc.

Memory is a key component in any processing system. The organization and capacity of memory directly affect the performance, power consumption and chip area of the target system. Especially for the computation-intensive tasks, which needs to perform a large number of computing in parallel, memory access throughput always become the bottleneck of the system. The centralized memory based on the traditional bus basically cannot satisfy the requirements on the data access bandwidth of the reconfigurable computing systems. A NoC-based distributed memory, which enables many PEs to perform simultaneously reading/writing accesses, is an outstanding solution for this problem. Adaptive bandwidth, memory subsystem architecture, memory access mechanisms are key issues when designing distributed on-chip memory.

The difference between a CGRA and a multi-core processor is that each core of multi-core processor is a complete processor including both control path and data path. In contrast, each PE of the CGRA only contains data path, and whole PEA is equipped one common control path. This helps to decrease the overhead for implementing the control path. The CGRA operation is driven by configuration contexts that keep the role like instructions of the processor. The context specifies particular operation of the PEA (i.e. operation of each Reconfigurable Cell (RC), interconnection between RCs, input source, output location, etc.) as well as the control parameters that control operation of the PEA. Similar to an instruction cycle of the processor, a context cycle is also composed of at least three phases including context fetching, context decoding, and executing. However, the difference is the CGRA just needs to be configured one time for executing multiple cycles. Once configured, CGRA operates as the hardware dedicated for the defined computation. The CGRA is just reconfigured when a different computation demand, which is equivalent to another context, appears. By contrast, the processor always has to perform all phases of an instruction cycle for every instruction even if the opcode of instructions does not change. In consequence, the CGRA performance is higher than that of the processor because the overhead time for performing two phases fetching and decoding is decreased.

5 Reconfigurable Network-on-Chips

5.1 Design Methodologies

The on-chip communication nowadays defines the performance of a MPSoC [24]. The on-chip communication medium should aim to provide high throughput, low latency, scalability, low power consumption, ensured QoS, reduced contention and less occupied area, etc. NoC paradigm has been proposed for SoCs design to meet the above requirements. Several design methodologies have been proposed to deal with NoCs and can be classified into two main categories, including design-time methodologies and run-time methodolo-

gies. Design-time methodologies (e.g. [12, 44], and [14]) are generally aimed at designing the NoC for a specific application. In other words, NoC is designed to function as the communication infrastructure in an ASIC or an application-specific SoC. Requirements of these designs are usually high throughput and low power. Therefore, all parameters, such as the on-chip interconnection architecture (i.e., topology, links), routing technique, switching schemes, and PE architecture etc., are defined at design time so that they are best optimized with the application. However, NoC should be scalable and adaptive to support various applications by selecting the most suitable parameters according to the requirements of the current application. Another problem is that the more and more complexity of SoCs makes them become very sensitive to permanent, transient, or intermittent faults [61]. These faults result in errors when passing data packet on the NoC, which affects the reliability of the SoC. To ensure the reliability of the system, these faults have to be detected by specific error detection blocks. Moreover, it is necessary to distinguish a permanent fault and a transient fault. If an error caused by a transient fault, the data can simply be retrieved by correcting or retransmitting. But if an error is caused by a permanent fault, this fault must be positioned, and then bypassed by either some adaptive control mechanisms or spare units. In this case, the failed elements must first be isolated from the communication system. Next, an adaptive routing algorithm allows packets to be re-routed in order to avoid faulty area. As a result, fault-tolerant router design (such as [12, 13], and [14]) is necessary to allow the system to continue operating in the presence of unexpected faults.

Unfortunately, the application-specific NoCs are not flexible enough to support dynamic environments where communication characteristics need to adapt strongly to various contexts at runtime. Recently, researchers have been paying a lot of attention to the development of run-time methodologies for reconfigurable NoCs. These methodologies focus on the techniques that allow NoCs to autonomously adapt its structure and their behavior during the period of their operation. Therefore, these NoCs can function as a communication infrastructure in reconfigurable SoC. These SoCs are often oriented to a certain application-domain instead of a specific application and therefore they need the reconfigurability of almost parts of the system. By this way, it is able to improve the flexibility of the design after fabricating. The parameters/elements of NoC that can be modified at run-time include reconfigurable topology (e.g. [50, 59], and [52]), reconfigurable links (e.g. Direction/Bandwidth of links [43]), reconfigurable PEs, and reconfigurable router (e.g. [45], and [55]). For example, a reconfigurable router can support adaptive routing techniques, adaptive switching techniques, adaptive number of VCs (virtual channels) and the buffer size per each VC etc., so that they can be dynamically adjusted based on the traffic load and network status.

Reconfiguration techniques allowing NoCs to meet the above requirement are as follows:

- **Low area and power consumption**: NoC is just the communication part in a SoC, therefore, its design must be simple enough so that area and power overhead resulted from the NoC is not too large compared with total area of the entire SoC. The resource is efficiently used by dynamically reconfiguring the NoC according to the requirements of user and/or system. For example, the links between two any nodes are often unidirectional channels for transmitting or receiving. If direction of channel is reconfigurable, it has potential to efficiently use of links, as well as increase bandwidth of links.
- **High performance** (including low latency and high throughput): Bandwidth requirement of the on-chip communication can vary depending on the user application, thus network with fixed communication parameters is not the optimal solution in terms of data throughput (and power). By dynamically changing link's direction or switching strategy so on, it has potential to improve the performance of the NoC.
- **QoS** (e.g. guaranteed-throughput, best-effort): implies reliably transferred data, reduced packet loss, etc. From this perspective, NoC architectures should be reconfigurable, for example, by rerouting at runtime traffic flow from congested area; or NoC elements must be able to change the routing path at runtime in order to efficiently avoid the faulty areas.
- **Scalability**: adding/removing a node is able to increase/decrease power consumption and latency. Reconfigurable technology is solution for this problem.
- **Reduced contention** (e.g. deadlock, live-lock, etc.): adaptive routing and flow control can help to reduce contention in the NoC.

5.2 Reconfigurable Parameters of NoC

The objective of NoC-based SoCs design is to find NoC instances that get the best trade-off between the cost criteria (e.g. area, power) and performance criteria (e.g. latency, throughput, and reliability). Pratomo *et al.* [57] had built scenarios for evaluating the impact of NoC parameters (e.g. packet rate, packet size, buffer size, routing algorithm) on the performance of SoC. Liu *et al.* [31] drew a comparison between circuit switched NoC and Packet Switched NoC. In real-life applications, almost most of parameters (i.e. communication load, buffer size, routing algorithm, switching diagram, packet rate, and topology, etc.) are determined in the NoC router. Thus, efficient and high-performance routers represent a critical issue when designing NoC. In reconfigurable SoC, reconfigurable router is needed to adapt to the immediate status of network so that the performance is not degraded. For example, in situation of faulty network, the reconfigurable router can adapt the NoC to the other operation mode by adjusting the topology, packet size and packet rate, or change

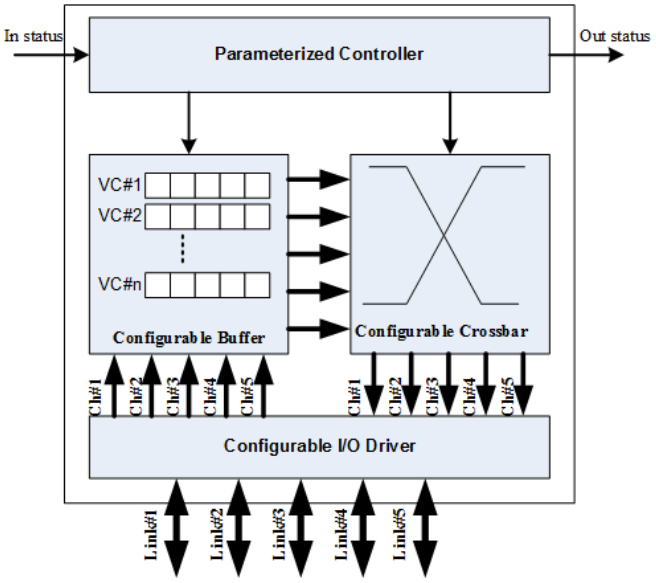


Figure 4. Block diagram of a reconfigurable router.

the type of its routing algorithm. Figure 4 shows a typically architecture of the reconfigurable router that includes four main blocks: parameterized controller, configurable buffer, configurable crossbar, and configurable I/O driver. It is popular that each router consists of four I/O (Input/Output) channels for connecting to four neighbour routers and another channel for connecting to a processing element. The I/O channels are connected to physical links via the configurable driver that can establish direction and bandwidth of links assigned to each channel. The configurable buffer is composed of an array of registers or storage elements that can be flexibly organized into several VC with variable size depending on applications. The crossbar is responsible for physically transfer data from the input channel to the output channel. Routing path and switching diagram of the crossbar is able to be configured at run-time. The controller takes charge of monitoring and controlling operation of the other modules, for example allocating VC and switch to input data packet. Depending on the information received via the input status port, the controller makes decision on routing path or switching diagram in order to adapt the router to dynamic status of the NoC.

Many research efforts are spent for proposing reconfigurable router. The authors in [42, 59] proposed the router that can be configured with different number of local and communication ports to build multi-dimensional networks (i.e., 2D and 3D) with different topologies and number of nodes. Similarly, a physical circuit-switching router, which is used to alter the topology of the NoC depending on the application mapped onto the SoC, also is publicized by Stensgaard [50] and Modarressi [52]. Ahmad *et al.* [40] proposed a router that can dynamically change the switching diagram between circuit switching technique and packet switching technique depends on the required bandwidth. Nguyen *et al.* [10] proposed a hybrid switching router based on the combination of wormhole and virtual cut-through switching schemes. The router is dynamically reconfigurable to exchange between switching schemes at run-time, therefore, it achieves higher average performance than wormhole switching, while reducing the implementation cost in comparison with the virtual cut-through switching. Wolkotte *et al.* in [56] proposed a circuit-switched router with the physical link that is partitioned into multi-lanes. The purpose of lanes is similar to that of virtual channels in packet-switched router. Al Faruque *et al.* [43] presented a novel configurable link which can change its supported bandwidth on-demand at run-time for the adaptive on-chip communication architecture. In [55], Nicopoulos *et al.* introduced a novel unified buffer structure, called the dynamic Virtual Channel Regulator (ViChaR), which dynamically can allocate VC and buffer resources according to the network traffic conditions. Therefore, ViChaR can maximize throughput by dispensing a variable number of VCs on demand. The authors also implemented a cycle-accurate simulator for the proposed NoC. Al Faruque *et al.* [43] proposed novel adaptive routing algorithm (called weight-XY) and adaptive buffer allocation scheme. The proposed adaptive buffer allocation scheme can re-assign a certain number of buffer blocks to different virtual channel for each of output ports of a router on-demand. Also related to the buffer organization, Huang *et al.* [45] proposed an output buffer that is based on shared memory. The shared memory is partitioned into sections that correspond to available output ports. Depending on the payload on each output port, an overload buffer can borrow some memory locations from an underload buffer at run-time. Meanwhile, some works focused on researching and proposing new routing algorithms such as buffer-less routing algorithm by Lin *et al.* [48], fault-tolerant adaptive routing algorithm by Pratomo *et al.* [57], Valinataj *et al.* [63], and Kumaran *et al.* [65]. Besides, some interested topics on Reconfigurable Network-on-Chip have been recently emerging, including self-reconfiguration and self-optimization of NoCs proposed by Bakhouya *et al.* [53], Killian *et al.* [46], Neishaburi *et al.* [13], or ideas for applying bio-inspired techniques to reconfigurable NoCs by Moadeli *et al.* [49], Mishra *et al.* [51], and Bakhouya *et al.* [53].

5.3 Simulators

NoC proposals need to be validated by software simulators or hardware emulators before its implementation. These tools are used either to evaluate the characteristics, functions, power consumption and performances of the NoC design (e.g. [54, 60]), or to explore design space of reconfigurable NoCs (e.g. [30, 41, 62, 64]). Although simulator is slow and the results can be different from reality due to the use of models, but simulator is the cheapest way to validate the design. In contrast, emulator is faster and the results can be close to reality due to the use of physical hardware platform (e.g. FPGA). Generally, NoC simulator can be classified into high-level and low-level environment depending on the abstraction where the simulator is

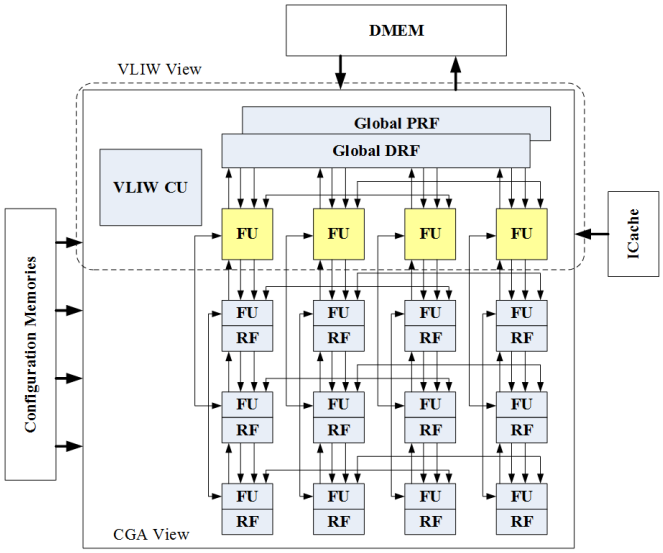


Figure 5. ADRES architecture template [35].

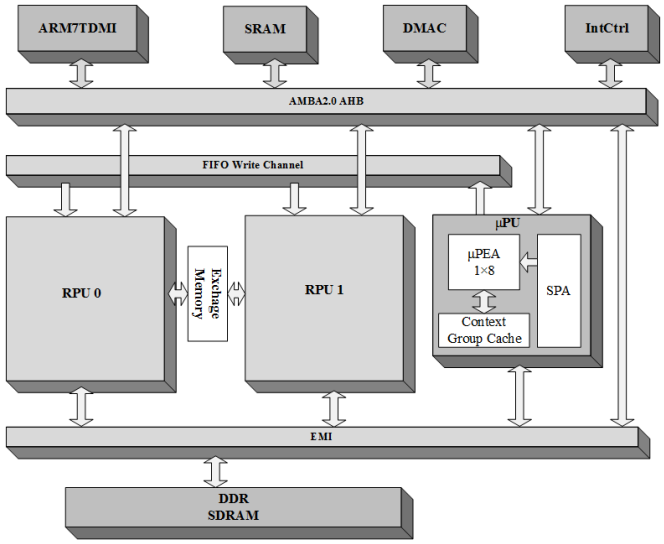


Figure 6. The overall architecture of REMUS-II [68].

modeled. High-level simulators are modeled at the behavioral and architectural levels thereby they are useful only in evaluating the functional correctness of a NoC proposal. By contrast, low-level simulators are required when the purpose of simulation is to aim at evaluating the performance or power consumption of NoC proposals. High-level simulators are usually written in either C^{++} language, such as VNOC, Booksim, and TOPAZ, etc. or java language, such as ATLAS and NoCsim. While Noxim, Nirgam and Nostrum, etc. can be classified in low-level simulators because of the SystemC language usage. Finally, the NoC proposals need to be implemented at RTL (Register Transfer Level) by describing the design in VHDL or Verilog HDL and prototyped on a FPGA platform (e.g. [47, 58, 59]).

6 State of the Art on Reconfigurable SoCs

6.1 ADRES

ADRES [35] (Architecture for Dynamically Reconfigurable Embedded System) is a reconfigurable system template (Figure 5), which tightly couples a VLIW processor and a coarse-grained reconfigurable matrix. The VLIW processor and the coarse-grained reconfigurable matrix are integrated into a single architecture but with two virtual functional views. The reconfigurable matrix has the capability to access directly to the register files, caches, and memories of the system. Besides, a methodology for mapping applications (described in C language) onto the ADRES template has been developed. The mapping methodology has the capability of exploiting loop-level parallelism of algorithms by using the modulo scheduling algorithm. The target domain of the ADRES is multimedia and loop-based applications.

6.2 REMUS-II

REMUS-II [68] is a coarse-grained dynamically reconfigurable heterogeneous computing SoC for multimedia and communication baseband processing. The REMUS-II consists of one or two coarse-grained dynamically reconfigurable processing units (RPUs) and an array of RISC processors (μPU) as shown Figure 6. The RPU is a powerful dynamically reconfigurable subsystem of four Reconfigurable Computing Arrays (RCAs) for speeding up computation-intensive tasks. In turn, each of RCAs is an array of 8×8 RCs (Reconfigurable Cells), and can run independently to accelerate the computing performance. The μPU consists of an array of RISC microprocessors (μPEA) and an array of stream processing elements (SPA). The microprocessors of the μPU communicate with each other and with the ARM processor by a simple mailbox mechanism. The SPA is a hardwired IP core for entropy decoding, whereas the μPEA is aimed at executing control-intensive parts and float-point operations of applications. By such hybrid, the REMUS-II can achieve the high performance approximately equivalent to that of ASIC while maintaining a degree of flexibility close to that of DSP processors.

6.3 MORPHEUS

MORPHEUS [6] is a dynamically reconfigurable heterogeneous SoC, which was developed by the European Union in the 6^{th} R&D Framework Program. MORPHEUS is integrated different reconfigurable technologies at several computation granularities that efficiently address the different requirement of the streaming applications. The overall architecture of MORPHEUS platform is shown in Figure 7. To target at different types of computation, MORPHEUS includes an ARM9 processing core and three heterogeneous reconfigurable computing engines that are connected together by a NoC architecture. ARM9 processor runs an embedded operating system and functions as the central controller for the whole platform. All of control, synchronization, and management functions are performed by an ARM9 processor. Three reconfigurable computing engines including XPP, DREAM, and M2000 have dif-

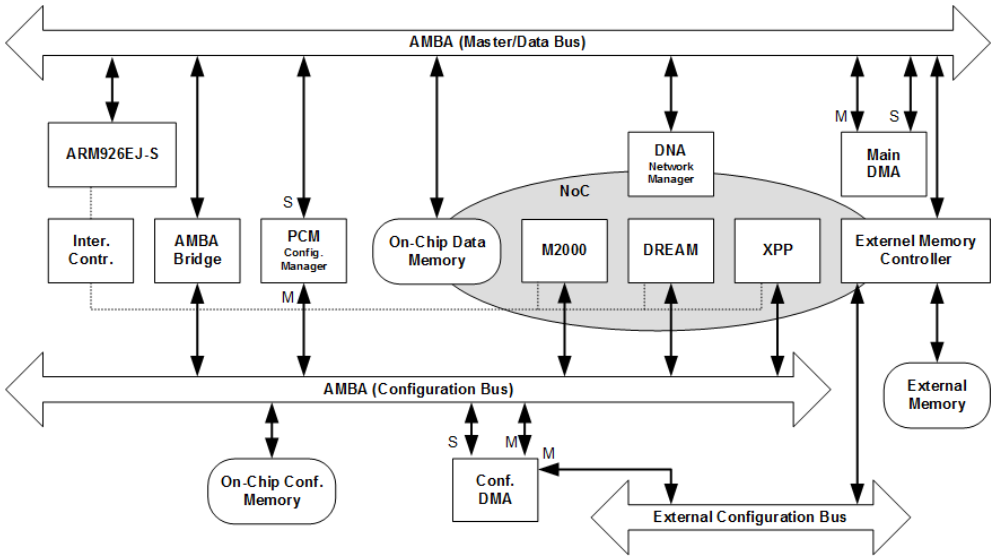


Figure 7. Simplified MORPHEUS platform architecture [6].

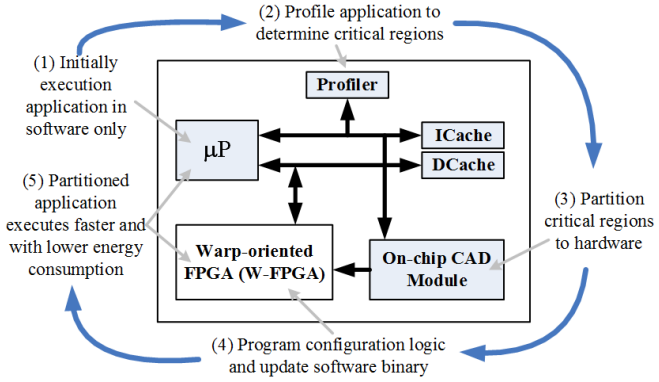


Figure 8. WARP processor architecture overview [29].

ferent configurable granularity. PACT XPP is a 16-bit coarse-grain reconfigurable array mainly targeting at computational-intensive algorithms. DREAM is based on a medium grain reconfigurable array consisting of 4-bit ALUs and 4-bit LUTs (Look-up Table). The architecture mainly targets instruction level parallelism of computation intensive algorithms that can run iteratively using only limited local memory resources. The M2000 is an embedded FPGA (eFPGA). In order to efficiently exploit the presented architecture, an integrated software toolset to map C applications to the different heterogeneous reconfigurable engines was also proposed. Thanks to the various types of reconfigurable technology, MORPHEUS can provide highly flexible environments to execute different kind of kernels and applications such as arithmetic computation, bit-level computation, control-intensive functions etc.

6.4 WARP

Researchers at the University of California have proposed a new processing architecture, known as a WARP processor that utilizes a FPGA to improve the performance and energy consumption of an application executing on a microprocessor [29]. This architecture consists of four main sections: a host processor, an efficient on-chip profiler, a WARP-oriented FPGA (W-FPGA), and an on-chip CAD (Computer as Design) module. Figure 8 shows the execution and dynamically mapping of a program on WARP. Unlike previous approaches that map application onto an FPGA using a cross compiler, the WARP processor dynamically maps an application onto its reconfigurable hardware by using the special hardware. The software's critical regions are dynamically detected by Profiler, and then synthesized to a custom hardware circuit in the FPGA by on-chip CAD. This paradigm allows users to exploit the FPGA to improve performance and reduce energy consumption with no required knowledge of the FPGA.

6.5 Cyberphysical system-on-chip (CPSoC)

Cyberphysical System-on-Chip (CPSoC) is an embedded computing platform that achieves self-awareness through a combination of cross-layer sensing, actuation, self-aware adaptations, and online learning [7]. The CPSoC platform is organized into several layers of abstraction including applications, operating system, network and bus communication, hardware, and the circuit/device layers as shown in Figure 9. The CPSoC chip consists of most features of the MPSoC based on an adaptive NoC and supported by on-chip sensing and actuation elements to enable the system to monitor its own state and behavior, as well as the external environment. In addition, an adaptive and reflective middleware with self-learning mechanisms take charge of managing and controlling both physical environment and characteristics of the chip. As a result, the system can adapt intelligently itself to various contexts of environment.

7 Conclusion

This paper illustrates the feasibility and potential of reconfigurable System-on-Chips. We provide an overall picture on reconfigurable SoCs with highlighting

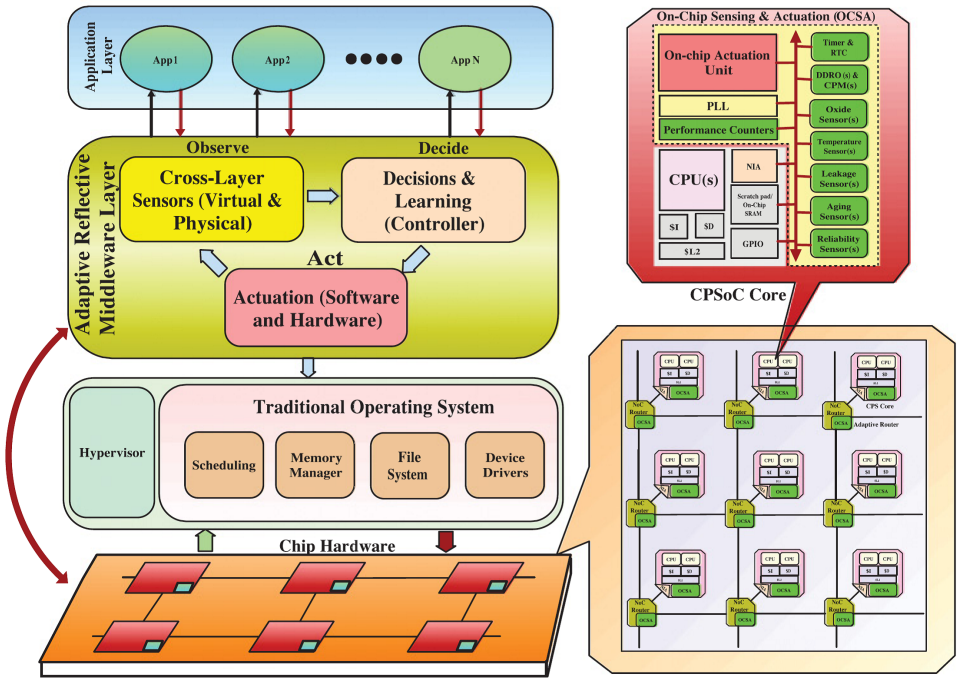


Figure 9. CPSoC architecture with adaptive Core, NoC, and the Observe-Decide-Act Loop as Adaptive, Reflexive Middleware [7].

main trends in the emerging architecture of reconfigurable SoCs, reconfigurable processing fabrics, and reconfigurable NoCs. We can foresee the possibility of reconfigurable computing becoming a feature in a variety of next generation high-performance embedded applications.

Acknowledgment

This work has been supported by Vietnam National University, Hanoi under Project No. QG.16.33 and Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2013.17.

References

[1] C. Bobda, *Introduction to Reconfigurable Computing - Architectures, Algorithms, and Applications*. Springer Netherlands, 2007, pp. 181– 212.

[2] A. Agarwal, C. Iskander, and R. Shankar, "Survey of Network on Chip (NoC) Architectures & Contributions," *Engineering, Computing and Architecture*, vol. 3, 2009.

[3] N. E. Jerger and L.-S. Peh, "On-Chip Networks," in *Synthesis Lectures on Computer Architecture*, ser. 8. Morgan & Claypool, 2009.

[4] A. Shoa and S. Shirani, "Run-Time Reconfigurable Systems for Digital Signal Processing Applications: A Survey," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 39, no. 3, pp. 213–235, Mar. 2005.

[5] H. K. Nguyen, P. Cao, X.-X. Wang, J. Yang, L. Shi, M. Zhu, L. Liu, and S. Wei, "Hardware Software Co-design of H.264 Baseline Encoder on Coarse-Grained Dynamically Reconfigurable Computing System-on-Chip," *IEICE Transactions on Information and Systems*, vol. E96.D, pp. 601–615, 2013.

[6] N. S. Voros, M. Hübner, J. Becker, M. Kühnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schüler, H. Sahlbach, S. Whitty, R. Ernst, E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein, and W. Putzke-Röming, "MORPHEUS: A heterogeneous dynamically reconfigurable platform for designing highly complex embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 3, 2013.

[7] N. Dutt, A. Jantsch, and S. Sarma, "Toward Smart Embedded Systems: A Self-aware System-on-Chip (SoC) Perspective," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 2, pp. 22:1–22:27, Feb. 2016.

[8] V. Tehre and R. Kshirsagar, "Survey on Coarse Grained Reconfigurable Architectures," *International Journal of Computer Applications*, vol. 48, no. 16, 2012.

[9] G. Haiyun, "Survey of Dynamically Reconfgurable Network-on-Chip," in *Proceedings of the Future Computer Sciences and Application (ICFCSA)*, 2011.

[10] H. K. Nguyen and X.-T. Tran, "Design and implementation of a hybrid switching router for the reconfigurable Network-on-Chip," in *Proceedings of the International Conference on Advanced Technologies for Communications (ATC)*, Oct. 2016, pp. 328–333.

[11] T.-T. Nguyen, T.-V. Le-Van, H. K. Nguyen, and X.-T. Tran, "Routing-path tracking and updating mechanism in reconfigurable Network-on-Chips," in *Proceedings of the International Conference on IC Design and Technology (ICICDT)*, Jun. 2016, pp. 1–4.

[12] A. R. F. Majid Janidarmian and and V. S. Bokharaei, "Application-Specific Networks-on-Chips Design," *IAENG International Journal of Computer Science*, vol. 38, no. 1, pp. 16–25, Feb. 2011.

[13] M. H. Neishaburi and Z. Zilic, "A Fault Tolerant Hierarchical Network on Chip Router Architecture," *Journal of Electronic Testing*, vol. 29, no. 4, pp. 485–497, Aug. 2013.

[14] F. Refan, H. Alemzadeh, S. Safari, P. Prinetto, and Z. Navabi, "Reliability in Application Specific Mesh-Based NoC Architectures," in *Proceedings of the 14th IEEE International On-Line Testing Symposium*, Jul. 2008, pp. 207–212.

[15] K.Masselos and N.S.Voros, *System level design of Reconfigurable Systems-on-Chip*. Springer, 2005.

[16] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: architectures and design methods," *IEEE*

Proceedings - Computers and Digital Techniques, vol. 152, no. 2, pp. 193–207, Mar. 2005.

[17] A. Abbas, M. Ali, A. Fayyaz, A. Ghosh, A. Kalra, S. U. Khan, M. U. S. Khan, T. D. Menezes, S. Pattanayak, A. Sanyal, and S. Usman, "A survey on energy-efficient methodologies and architectures of network-on-chip," *Computers & Electrical Engineering*, vol. 40, no. 8, pp. 333 – 347, Nov. 2014.

[18] E. Fernandez-Alonso, D. Castells-Rufas, J. Joven, and J. Carrabina, "Survey of NoC and Programming Models Proposals for MPSoC," vol. 9, no. 2, pp. 22–32, 03 2012.

[19] R. Pop and S. Kumar, "A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems," School of Engineering, Jonkoping University, SWEDEN., Tech. Rep., 01 2013.

[20] P. K. Sahu and S. Chattopadhyay, "A Survey on Application Mapping Strategies for Network-on-Chip Design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[21] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proceedings of the 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–10.

[22] W. Wayne, J. Ahmed, and M. Grant, "Multiprocessor System-on-Chip (MPSoC) Technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[23] A. Todd, B. David, M. Scott, M. Trevor, C. Chaitail, and W. Wayne, "Mobile supercomputers," *Computer*, vol. 37, no. 5, pp. 81–83, May 2004.

[24] M. Duranton, K. D. Boschere, A. Cohen, J. Maebe, and H. Munk, "The HiPEAC Vision, HiPEAC Roadmap," Tech. Rep., 2015. [Online]. Available: http://www.hipeac.net/system/files/hipeacvision.pdf.

[25] M. K. A. Ganesan, S. Singh, F. May, and J. Becker, "H. 264 Decoder at HD Resolution on a Coarse Grain Dynamically Reconfigurable Architecture," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, Aug. 2007, pp. 467–471.

[26] S. Kim, Y. H. Park, J. Kim, M. Kim, W. Lee, and S. Lee, "Flexible video processing platform for 8K UHD TV," in *Proceedings of the IEEE Hot Chips 27 Symposium (HCS)*, Aug. 2015, pp. 1–1.

[27] L. Liu, D. Wang, M. Zhu, Y. Wang, S. Yin, P. Cao, J. Yang, and S. Wei, "An Energy-Efficient Coarse-Grained Reconfigurable Processing Unit for Multiple-Standard Video Decoding," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1706–1720, Oct. 2015.

[28] Y.-H. Chen and P.-A. Hsiung, "Hardware Task Scheduling and Placement in Operating Systems for Dynamically Reconfigurable SoC," *LNCS*, vol. 3824, pp. 489–498, 2005.

[29] R. Lysecky, G. Stitt, and F. Vahid, "Warp Processors," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 3, pp. 659–681, Jun. 2004.

[30] A. E. Kiasari, "Performance Analysis and Design Space Exploration of On-Chip Interconnection Networks," Ph.D. dissertation, Doctoral Thesis in Electronic and Computer Systems KTH Royal Institute of Technology, Sweden, 2013.

[31] S. T. Liu, A. Jantsch, and Z. H. Lu, "Comparison of Circuit Switched NoC with Packet Switched NoC," in *Proceedings of the Fifth Swedish Workshop on Multicore Computing*, 2012.

[32] XILINX, *Zynq-7000 All Programmable SoC*, 2013.

[33] Altera, "Intel Arria 10 SoC FPGA devices," Tech. Rep., 2014.

[34] "International Technology Roadmap for Semiconductor," www.itrs.net/reports.html, Tech. Rep.

[35] B. Mei, M. Berekovic, and J.-Y. Mignolet, *ADRES & DRESC: Architecture and Compiler for Coarse-Grain Reconfigurable Processors*. Springer Netherlands, 2007, pp. 255–297.

[36] J. M. Cardoso and P. C. Diniz, *Compilation Techniques for Reconfigurable Architectures*, 1st ed. Springer US, 2009.

[37] R. Dafali, J.-P. Diguet, and M. Sevaux, "Key Research Issues for Reconfgurable Network-on-Chip," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs*, 2008.

[38] S. Vassiliadis and D. Soudris, *Fine- and Coarse-Grain Reconfigurable Computing*. Springer Netherlands, 2007.

[39] T. Marconi, Y. Lu, K. Bertels, and G. Gaydadjiev, *Online Hardware Task Scheduling and Placement Algorithm on Partially Reconfigurable Devices*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 306–311.

[40] B. Ahmad, A. T. Erdogan, and S. Khawam, "Architecture of a Dynamically Reconfigurable NoC for Adaptive Reconfigurable MPSoC," in *Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, 2006, pp. 405–411.

[41] A. B. Achballah and S. B. Saoud, "A Survey of Network-On-Chip Tools," in *Proceedings of the International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 4, no. 9, 2013.

[42] A. F. Beldachi, M. Hosseinabady, and J. L. Nunez-Yanez, "Configurable router design for dynamically reconfigurable systems based on the SoCWire NoC," *International Journal of Reconfigurable and Embedded Systems*, vol. 2, no. 1, pp. 27–48, Mar. 2013.

[43] M. A. A. Faruque, T. Ebi, and J. Henkel, "Configurable links for runtime adaptive on-chip communication," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, April 2009, pp. 256–261.

[44] F. K. Koupaei, A. Khademzadeh, and M. Janidarmian, "Fault-Tolerant Application-Specific Network-on-Chip," in *Proceedings of the World Congress on Engineering and Computer Science (WCECS 2011)*, vol. II, San Francisco, USA, Oct. 2011.

[45] P.-T. Huang and W. Hwang, "2-level FIFO Architecture Design for Switch Fabrics in Network-on-Chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2006, pp. 4 pp.–4866.

[46] C. Killian, C. Tanougast, F. Monteiro, and A. Dandache, "A New Efficient and Reliable Dynamically Reconfigurable Network-on-Chip," *Journal of Electrical and Computer Engineering*, vol. 2012, 2012.

[47] Y. E. Krasteva, E. de la Torre, and T. Riesgo, "Reconfigurable Networks on Chip: DRNoC Architecture," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 293–302, Jul. 2010.

[48] J. Lin, X. Lin, and L. Tang, "Making-a-stop: A New Bufferless Routing Algorithm for On-chip Network," *Journal of Parallel and Distributed Computing*, vol. 72, no. 4, pp. 515–524, Apr. 2012.

[49] M. Moadeli, P. Maji, and W. Vanderbauwhede, "Quarc: A High-Efficiency Network on-Chip Architecture," in *Proceedings of the International Conference on Advanced Information Networking and Applications*, May 2009, pp. 98–105.

[50] M. B. Stensgaard and J. Spars, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology," in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008.

[51] P. Mishra, N. A, and J. K. Kishore, "Novel bio-inspired cobweb topology for highly scalable and cost efficient networks on chip," in *Proceedings of the IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Jan. 2014, pp. 1–6.

[52] M. Modarressi, H. Sarbazi-Azad, and M. Arjomand, "A hybrid packet-circuit switched on-chip network based on SDM," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, Apr. 2009, pp. 566–569.

[53] M. Bakhouya, "Towards a bio-inspired architecture for autonomic network-on-chip," in *Proceedings of the Inter-*

national Conference on High Performance Computing Simulation, Jun. 2010, pp. 491–497.
[54] N. Jiang, D. B. U., G. Michelogiannakis, J. D. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE Computer Society, 2013, pp. 86–96.
[55] C. A. Nicopoulos, D. Park, J. Kim, and C. R. D. N. Vijaykrishnan, Mazin S. Yousif, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, 2006.
[56] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, Apr. 2005.
[57] I. Pratomo and S. Pillement, "Impact of design parameters on performance of adaptive Network-on-Chips," in *Proceedings of the International Conference on High Performance Computing Simulation (HPCS)*, Jul. 2012, pp. 724–725.
[58] V. Rana, D. Atienza, M. D. Santambrogio, D. Sciuto, and G. De Micheli, "A Reconfigurable Network-on-Chip Architecture for Optimal Multi-Processor SoC Communication," in *Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration - System on a Chip*, C. Piguet, R. Reis, and D. Soudris, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Oct. 2008.
[59] R. Vancayseele, B. A. Farisi, W. Heirman, K. Bruneel, and D. Stroobandt, "RecoNoC: A reconfigurable network-on-chip," in *Proceedings of the 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, Jun. 2011, pp. 1–2.
[60] S. E. Lee and N. Bagherzadeh, "A high level power model for Network-on-Chip (NoC) router," *Computers & Electrical Engineering*, vol. 35, no. 6, pp. 837–845, Nov. 2009.
[61] Q. Yu and P. Ampadu, *Transient and Permanent Error Control for Networks-on-Chip*. Springer, 2012.
[62] W.-C. Tsai, Y.-C. Lan, Y.-H. Hu, and S.-J. Chen, "Networks on Chips: Structure and Design Methodologies," *JECE*, vol. 2012, pp. 2:2–2:2, Jan. 2012.
[63] M. Valinataj, S. Mohammadi, and S. Safari, "Fault-aware and Reconfigurable Routing Algorithms for Networks-on-Chip," *IETE Journal of Research*, vol. 57, no. 3, pp. 215–223, 2011.
[64] Z. Qian, D. C. Juan, P. Bogdan, C. Y. Tsui, D. Marculescu, and R. Marculescu, "A comprehensive and accurate latency model for Network-on-Chip performance analysis," in *Proceedings of the 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2014, pp. 323–328.
[65] G. Kumaran and S. Gokila, "Dynamic Router Design for Reliable Communication," in *Proceedings of the International Conference on Global Innovations In Computing Technology*, 2014.
[66] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, and N. Bagherzadeh, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Transactions on Computers*, vol. 49, no. 5, pp. 465–481, 2000.
[67] X. Technologies, "XPP-III Processor Overview," White Paper, Tech. Rep., Jul. 2013.
[68] X. Liu, C. Mei, P. Cao, M. Zhu, and L. Shi, "Date Flow Optimization of Dynamically Coarse Grain Reconfigurable Architecture for Multimedia Applications," *IEICE Transactions*, vol. 95-D, no. 2, pp. 374–382, 2012.
[69] H.-L. Chao, S.-Y. Tung, and P.-A. Hsiung, "Dynamic Task Mapping with Congestion Speculation for Reconfigurable Network-on-Chip," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 10, no. 1, pp. 3:1–3:25, Sep. 2016.
[70] A. Nieto, D. L. Vilarino, and V. M. Brea, "PRECISION: A Reconfigurable SIMD/MIMD Coprocessor for Computer Vision Systems-on-Chip," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2548–2561, Aug. 2016.
[71] F. An, X. Zhang, L. Chen, and H. J. Mattausch, "Dynamically reconfigurable system for LVQ-based on-chip learning and recognition," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 1338–1341.

Hung Kiem Nguyen received the B.S. and M.S. degrees in electronic engineering from Vietnam National University, Hanoi, Vietnam, in 2003 and 2005, respectively. He received the Ph.D. degree in electronic engineering from Southeast University, Nanjing, China in 2013.

He is currently a researcher at VNU Key Laboratory for Smart Integrated Systems, VNU University of Engineering and Technology. His research interests mainly include multimedia processing, reconfigurable computing, and SoC designs.

Thanh-Vu Le-Van was born in Hue, Vietnam. He received a Bachelor of Science degree in Physics from Hue University of Sciences – a member university of Hue University and a Master of Science degree from Vietnam National University Hanoi (VNU) in Electronics Engineering and Communications.

Currently, he is pursuing his Ph.D degree at VNU University of Engineering and Technology (VNU-UET), a member university of VNU. He joined the Key Laboratory for Smart Integrated Systems (SIS Lab) of VNU-UET in 2010. His research interests include Network-on-Chip (NoC), High-level modeling methods, Reconfigurable architecture.

He is a member of the IEEE (SSCS) and the Radio Electronics Association of Vietnam (REV).

Xuan-Tu Tran received a Ph.D. degree in 2008 from Grenoble INP (in collaboration with the CEA-LETI), France, in Micro Nano Electronics. He is currently an Associate Professor at the Faculty of Electronics and Telecommunications, VNU University of Engineering and Technology (VNU-UET), a member university of Vietnam National University, Hanoi (VNU). He is also Adjunct Professor at the University of Technology Sydney (UTS), Australia.

He is currently Director of VNU Key Laboratory for Smart Integrated Systems (SISLAB). He is in charge for CoMoSy, VENGME, ReSoNoC, ADEN4IOT projects for embedded systems and multimedia applications. His research interests include design and test of systems-on-chips, networks-on-chips, design-for-testability, asynchronous/synchronous VLSI design, low power techniques, and hardware architectures for multimedia applications.

He is a Senior Member of the IEEE, IEEE Circuits and Systems (CAS), IEEE Solid-State Circuits Society (SSCS), member of IEICE, and the Executive Board of the Radio Electronics Association of Vietnam (REV).