Review of Reconfigurable Architectures for the Next Generation of Mobile Device Telecommunications Systems

Ahmed O. El-Rayis*[1], Tughrul Arslan[2], Khalid Benkrid[3]

[1, 2, 3]School of Engineering, The University of Edinburgh, Edinburgh EH9 3JL, UK

[*1]A.El-Rayis@ed.ac.uk; [2]T.Arslan@ed.ac.uk; [3]K.Benkrid@ed.ac.uk

***Abstract*-The development of mobile devices has challenged hardware designers to come up with suitable architectures. Challenges such as power consumption, flexibility, processing power and area are likely to lead to the need for a reconfigurable architecture to cater for the growing demands made of mobile devices, and to suit the needs of the next generation of devices. Parallelism and multifunction in real-time will be the minimum required characteristics of the architectures of such devices. This chapter reviews the currently available reconfigurable architectures. The focus here is on coarse-grain reconfigurable architectures, with particular attention to those which support dynamic reconfiguration with low-power consumption. The capacity for dynamic reconfiguration will be a key factor in defining the most suitable architecture for future generations of mobile devices.**

This paper describes existing reconfigurable platforms. Their principles of operation, architectures and structures are discussed highlighting their advantages and disadvantages. Various coarse-grain reconfigurable architectures are discussed along with their improvement with time. Finally, the key characteristics which are required for a reconfigurable architecture to be suitable for telecommunication systems are identified. A comparison is given for the various architectures discussed in terms of suitability for telecommunications applications.

Keywords- Coarse Grain Reconfigurable Architecture; FPGA; ASIC; DSP; Dynamic Reconfiguration; Low Power Consumption

I. INTRODUCTION

The origins of reconfigurable computing date back to the 1960s when the concepts were proposed by Gerald Estrin [1]. The first FPGA (field programmable gate array) was introduced by Carter et al. [2]. Before the 1980s, software programmed microprocessors were the only available resource for providing flexibility. The emergence of the FPGA changed this situation. Configuration bits changes the hardware realisation in FPGA as software instructions is programming the processor. This led to another definition of reconfigurable computing, as a system incorporating programmable logic to customise existing hardware. Programmable logic is connected by flexible interconnects which can be changed periodically to execute different implementations on the same hardware, thus providing an ASIC (application-specific integrated circuit) solution with post-fabrication programmability.

The FPGA consists of two main components, which are logic blocks and interconnections or switches as illustrated in Fig. 1. The programmable logic blocks can be programmed along with the interconnections or switched array to perform a certain logic function. The programmable logic block of one of the leading FPGA manufacturers, Xilinx consists of a 3-input look-up table (LUT), a multiplexer and a flip-flop in its basic building block [3]. Nowadays, Xilinx's series seven includes more complicated FPGAs with embedded processors, large memory blocks and even transceivers along with various interface protocols [4]. The FPGA is a fine-grain reconfigurable architecture where it is based on a single bit operation. FPGAs require a high volume of configuration data, and the mapping functions are difficult and require specific skills.

The FPGA has various advantages, such as quick prototyping, speedy development and clear basic blocks. However, it is obvious that it has various drawbacks, such as being fine-granular. This property means that configuration data are complicated,

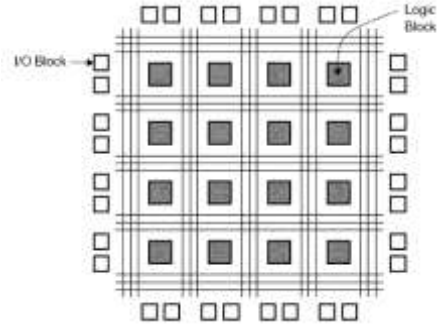

Fig. 1 FPGA building block - Xilinx

which as a consequence leads to the need for a large configuration memory. This in turn increases the power consumption, area and design complexity.

Architectures that use FPGA-based coprocessor along with a general purpose processor are capable in processing complex algorithms [5-[7]. However, this type of configuration has two main drawbacks. Firstly, a wider datapath requires a large area and longer delays occur due to the small width of the programmable logic block; secondly, FPGAs have lower logic density and are slower than a custom ASIC [8].

In course-grain architectures, the datapath ranges from 2 to 32 bits or more. The selection of the datapath width is a trade-off between flexibility, efficiency and programmability. The main advantage of a reconfigurable processor or functional unit is the ability to customize hardware for a specific algorithms or function's requirement.

II. GENERAL, DSP, FINE AND COARSE PROCESSORS

Multiprocessors or multi-core systems are increasingly introduced in personal computing and smart mobile devices. There have been several approaches to the enhancement of general-purpose architectures in order to increase performance, such as the SIMD (single instruction, multiple data), MIMD (multiple instructions, multiple data) and VLIW (very long instruction word) methods.

Reconfigurable computation can enable increased computational performance and lower energy consumption. Configurable computing combines the performance of application-specific hardware with the reprogrammability of general-purpose computers [9]. Fig. 2 presents a distribution of the computation architectures in relation to flexibility, area, power consumption and performance. It is clear that ASIC is the best in terms of performance, and having the lowest power consumption and area; while on the other hand, the general purpose processor has the highest flexibility but also suffers having the highest power consumption, lowest performance and largest area. The focus in this chart is the highlighted dotted rectangle at the bottom right-hand side, where low power consumption and area are the main characteristics. From the point of view of flexibility within this zone, then the coarse-grain reconfigurable architecture gives the best of both worlds in terms of power consumption, flexibility, area and function diversity. This would appear to be the privileged space for architectures that could satisfy the requirements of mobile communication systems.

From the point of view of functional diversity, microprocessors or general purpose processors can achieve more functions than FPGA and other reconfigurable architectures. On the other hand, a reconfigurable architecture can achieve higher performance than processors on highly repetitive computing tasks with limited functional diversity [5].

The rapid increase in demand for computation load has resulted in a number of accelerator styles. These can take the form of specialised extended instruction-specific processors, custom hardware, intense kernel codes, and reconfigurable computing. Such accelerators can be implemented as independent processors, co-processors or customised IP [10].

Wireless applications need processing modules that simultaneously demonstrate high computational performance, ultra-low-power consumption and a high degree of flexibility and adaptability. Reconfigurability is a necessity in the presence of multiple and evolving standards in dynamic conditions. The computing challenges for mobile devices are area, power and computing power efficiency.

To increase computing power, approaches such as larger processors, dedicated fabrics with application-specific cores, and reconfigurable computing have been considered. Most computationally complex applications spend 90% of their execution time on only 10% of their code [11].

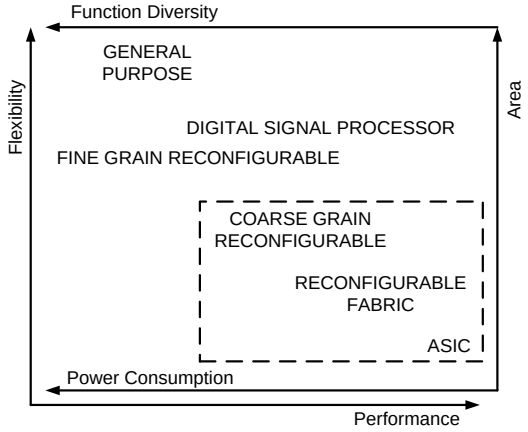

Fig. 2 Computation architecture characteristics

A. Reconfigurable Computing Classes

As mentioned earlier, a key interest in discussing reconfigurable architectures is their capacity for reconfiguration and their flexibility. From the literature, it is clear that there are different types of reconfiguration. In partial reconfiguration (PR), only a part of the reconfigurable fabric is reconfigured and there are two types, static and dynamic. The meaning of static partial reconfiguration (SPR) is clear from its name; however, dynamic partial reconfiguration (DPR) usually requires an external configuration control module. An improvement upon DPR is dynamic partial self-reconfiguration (DPSR), which does not require an external configuration control module. Various reconfigurable architectures are discussed in the following sections.

III. COARSE GRAIN RECONFIGURABLE ARCHITECTURES

Reconfigurable architectures are discussed in this section and their suitability for communication systems is considered.

A. CRISP: A Course-Grained Reconfigurable Instruction Set Processor

Francisco and et al. [12] presented a coarse-grain reconfigurable processor named CRISP, which consists of a processor and a reconfigurable logic. Thus it is based on the concept of a co-processor for the reconfiguration part or as an add-on functional unit. The architecture is illustrated in Fig. 3. The reconfiguration functional unit is activated through a special reconfigurable instruction from the main processor. The architecture targets multimedia applications, with performance claimed by the authors to be 2.5 times that of a RISC processor with an 18% energy overhead [12].

This architecture operates on 8 to 32 bits. The reconfigurable logic in this architecture consists of reconfigurable slices. Each slice contains several processing elements (PEs), a register file, a programmable interconnect and a small configuration memory. The internal elements of the single reconfigurable slice are shown in Fig. 3. Each PE can be an ALU (arithmetic logic unit), a shifter, multiplier or memory unit. The interconnection used in this architecture is a full crossbar. The slice internal structure is illustrated in Fig. 4.

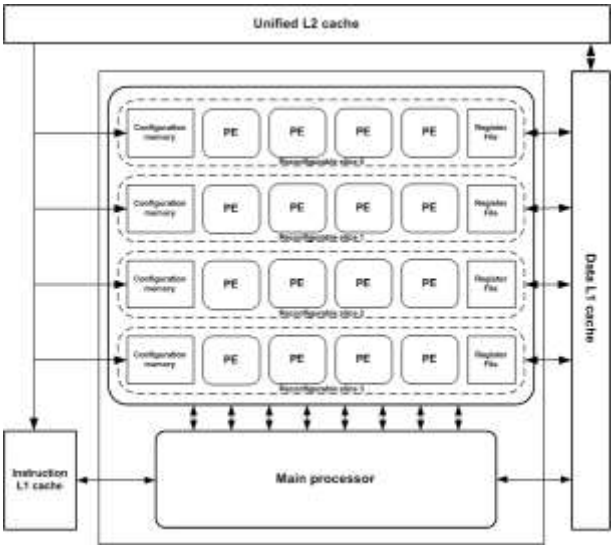

Fig. 3 CRISP architecture's construction of processor and a reconfigurable functional unit [12]

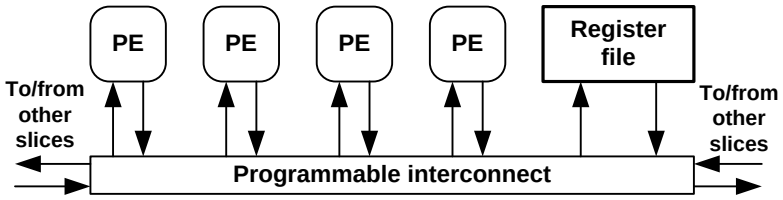

Fig. 4 Reconfigurable slice internal structure [12]

The CRISP architecture is novel in that its reconfigurable part is based on two levels or layers, the slice and PE layers. From one point of view, this appears to be incompatible to other mainstream reconfigurable architectures where usually the first layer is the PE and underneath it can be another layer. However, from a power savings point of view, it is an interesting concept since the inactive slice will be turned off completely along with all embedded PEs. From the programmability, mapping and computation distribution points of view, it appears that it is quite complex to realise functions on this architecture.

The slices appear to be integrated reconfigurable processors where, by default, the results of all PEs have to be written into the register file of the slice prior to interaction with the outer world such as other slices, memory or the main processor. The authors claim that the interconnections could be configured in such a way as to allow direct interactions between PEs within different slices [5] and [12]. The realisation of this will be complex from a mapping point of view and costly in terms of configuration time. In conclusion, the architecture is appealing for its power saving ability when unused slices are disabled and its ability to execute loops with multi-integrated configurations. The cost arising from the latter is degraded performance if the number of configurations would surpass configuration cache limitations.

B. Systolic Ring Architecture

The systolic ring architecture is a coarse-grained arithmetic block which includes a custom RISC (reduced instruction set computing) processor [13]. The role of the processor here is to dynamically configure the architecture and control dataflow at the operative layer. The architecture is divided into operation and configuration layers, as illustrated in Fig. 6. The operation layer is the reconfiguration part where the processing elements reside, while the configuration layer consists of RAM that holds the configuration information which resembles a FPGA. The RAM contents change every clock cycle.

The PE of this architecture is called the Dnode (data node). It consists of an ALU, datapath components and a few registers. It is configured using micro instruction code. Fig. 5 demonstrates the PE or Dnode architecture. Each Dnode has two execution modes, normal and standalone. In normal mode, the Dnode is in operation where it follows the micro-instruction code. The stand-alone mode allows the Dnode to take up to six clock cycles to process data or instructions located internally in its own seven registers.

This architecture is called a “ring” due to the fact that the Dnodes are arranged in a ring style or pipelined systolic structure. Each two adjacent Dnodes create a layer and can interact with neighbouring layers through the switches shown in Fig. 7. The length of the structure is the number of layers, and its width is the number of Dnodes per layer.

In this architecture the datapath (dataflow) is separated from data feedback or Dnode results. Data feedback passes through isolated dedicated pipelines through the switches as seen in Fig. 7. Its designers claim that this technique of separation dramatically reduces routing problems and supports the architecture scalability [13]. It is clear that this architecture imitates the FPGA in having operational CLBs and a configuration layer which is usually a large SRAM. This architecture is a clear step forward for coarse-grain reconfiguration, and its interconnection principle is interesting. However, mapping would be a complicated task given the complex ring structure used for the connections. Most importantly, the usage of RAM will clearly increase area and power consumption of the architecture. Also the architecture is clearly complex from an implementation point of view, due to the need for an embedded RISC processor within the architecture just to drive the configuration, while the whole architecture has to interface with an external processor in order to act as a co-processor as illustrated in Fig. 6.

C. MATRIX Architecture

The MATRIX (multiple ALU architecture with reconfigurable interconnect experiment) architecture is built according to an application-specific methodology, aiming to be suitable for general purpose applications [14]. The architecture is composed of an array of identical 8-bit basic functional units and a configuration network. The basic functional units (or processing elements) in this architecture include the following three main components as shown in Fig. 8: 256 x 8-bit memory; an 8-bit ALU and multiply unit, and a control logic.

A multiply operation takes two operating cycles. The architecture is by default pipelined due to the existence of a pipeline register at the input port of each function unit.

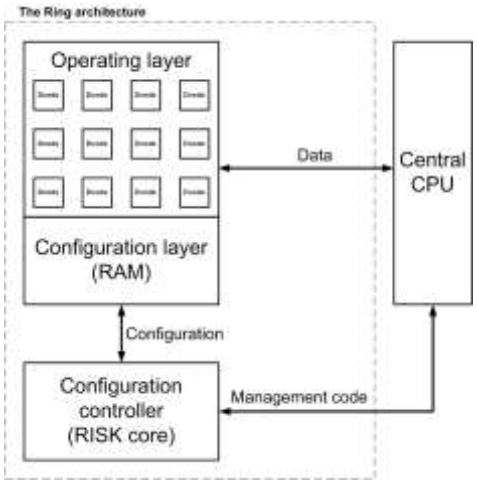

Fig. 6 The Ring architecture layout [13]

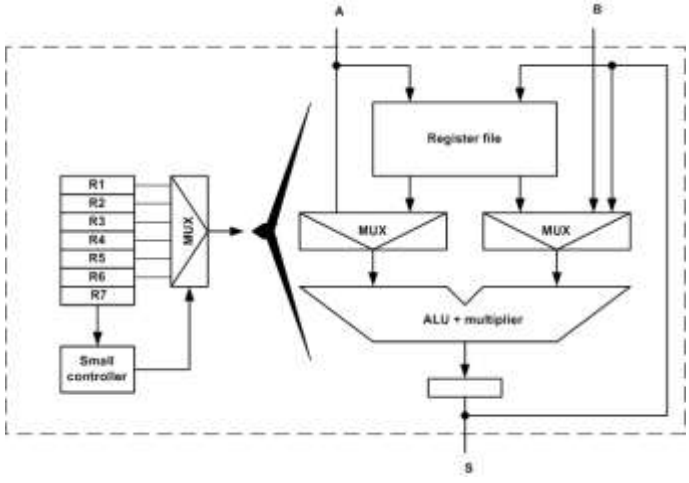

Fig. 5 The Dnode architecture [13]

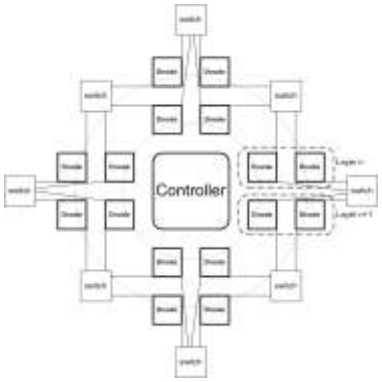

Fig. 7 Ring style for Dnode interconnections [13]

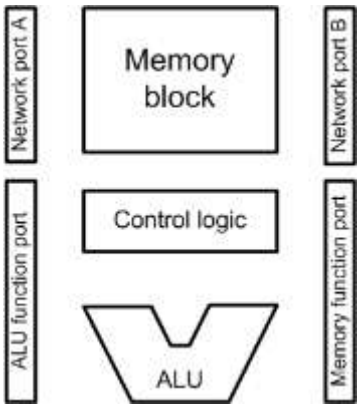

Fig. 8 MATRIX Processing element (functional unit) [14]

The interconnection used in this architecture is almost a crossbar style interconnection network. It has the capability of connecting nearest neighbours. Also it has four bypass connections and global lines. Global lines imply the usage of the four interconnect lines. Nearest neighbour interconnections can allow a single processing element to have direct connections with up to 12 neighbouring PEs as shown in Fig. 8.

One key aspect of this architecture is the port programmability of the basic function unit. The port configuration can be a holder of the input values of the ALU and this is termed static value mode. Meanwhile, in the static source mode, the word hold in the port is used to select the network bus from which data can be received. Another mode for the port configuration is the dynamic source mode where the port configuration word is ignored and the associated floating port controls the input source on a cycle-by-cycle basis.

Another attractive point in this architecture is its ability to be configured in order to operate VLIW, SIMD, MIMD, MSIMD or hybrids of these. Moreover, the architecture datapath can be wired up in an application specific manner.

The architecture's authors claimed that no specific applications are targeted and that the architecture can be a general purpose one [14].

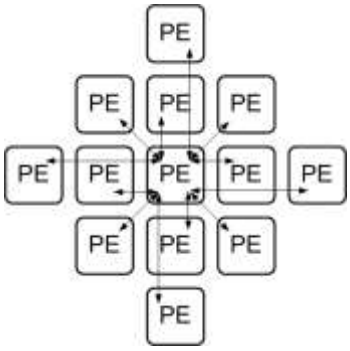

Fig. 9 MATRIX nearest neighbour interconnect

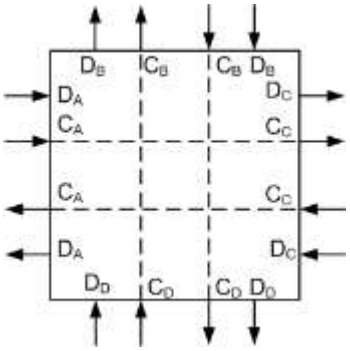

Fig. 10 PE structure for the Cell Matrix architecture [15]

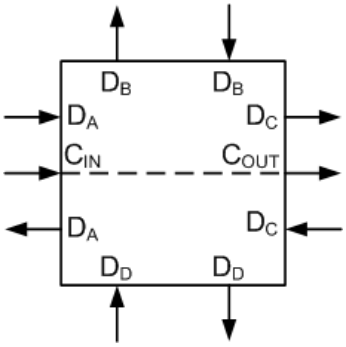

Fig. 11 PE structure for the vCell Matrix architecture [11]

D. Cell Matrix and vCell Matrix Architectures

The vCell Matrix architecture [11] is based on a commercially available architecture named the cell Matrix architecture [15]. The vCell architecture promises a simpler and faster reconfiguration mechanism compared with the cell matrix. Both architectures consist of two-dimensional homogeneous cell arrays.

The PE in the Cell matrix architecture is called a Cell and is illustrated in Fig. 10, while the PE of the vCell matrix architecture is called the vCell and its structure is illustrated in Fig. 11.

Each vCell has four input and four output data ports of 1-bit each distributed on its four sides (D_{A-D}). In addition, the vCell has two configuration control ports, one input and one output, named C_{IN} and C_{OUT}. When the input configuration control port is activated, it allows the vCell to store the configuration data through the data ports into its internal LUT. The output configuration control port allows the vCell to control the mode of operation of its neighbouring cells. Each cell is connected to its nearest neighbour to the north, south, east and west, as illustrated in Fig. 12. In the Cell matrix architecture, the

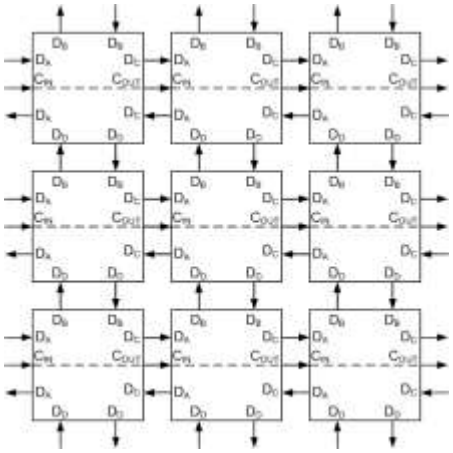

Fig. 12 Two-dimensional array structure for the vCell Matrix architecture[11]

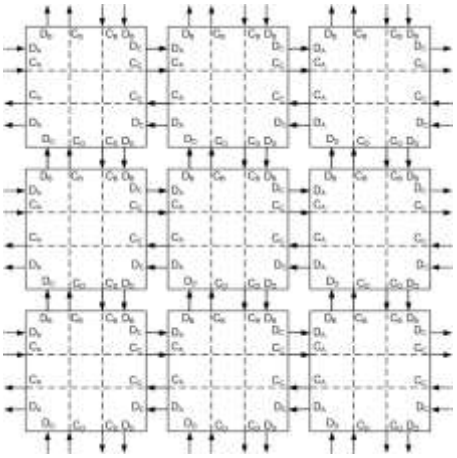

Fig. 13 Two-dimensional array structure for the Cell Matrix architecture

reconfiguration process is distributed, so that any cell can initiate the reconfiguration process by configuring its nearest neighbours; hence it supports dynamic partial self-reconfiguration (DPSR). The array structure of the Cell matrix architecture is illustrated in Fig. 13 where each cell is capable of configuring its nearest neighbour. The cell needs to be configured first as a data bus in order to pass configuration data to the furthest cell. In this architecture, the configuration mechanism allows great flexibility; however, it requires a complex and sophisticated configuration algorithm.

Conversely, in the vCell matrix architecture, each vCell can configure only its eastern neighbour, as shown in Fig. 12. In addition, the vCell cannot initiate the configuration process by itself. It has only two configuration ports, C_{IN} west and C_{OUT} east, whereas the Cell has eight configuration ports covering all the sides of the cell. The reduction in the number of configuration ports in the vCell significantly reduces the architecture's configuration flexibility as compared with that of the Cell architecture. Despite this drawback, the reduced number of configuration ports has the advantages of a simpler configuration mechanism and a smaller LUT within each individual vCell. The Cell matrix architecture is intended for a wide range of applications, being general purpose. The vCell matrix architecture, however, is suitable for applications that require a regular datapath and a simple control path, and thus mainly DSP applications.

E. Pleiades Architecture

The Pleiades processor architecture is based on the combination of a main processor coupled with an array of heterogeneous computational units of various granulates [16]. The PEs here are heterogeneous computational units and are named satellite processors. In addition to the satellite-processors, the architecture includes a reconfigurable interconnect network. The architecture's layout is presented in Fig. 14. The processor runs on data intensive loops called 'kernels.' Synchronisation between the satellite processors achieved is by a data-driven communication protocol in relation to the kernels.

The architecture operates direct memory read/write. The mesh structure has a two-level hierarchical reconfigurable interconnect network. The architecture address generator can handle addressing issues in addition to nested loops with loop counters. It controls the dataflow threads from initiation until end.

Because the system is realised on a data-driven principle, synchronisation between the processing elements employs a two-phase self-timed handshaking protocol consisting of request and acknowledge signals. This is realised in asynchronous fashion.

The architecture data-links consist of a 16-bit fixed-width data word in addition to 2-bit control signals, while the configuration bus is 32bit.

The hierarchical network architecture has sufficient connection flexibility for targeted applications, and in addition it cuts

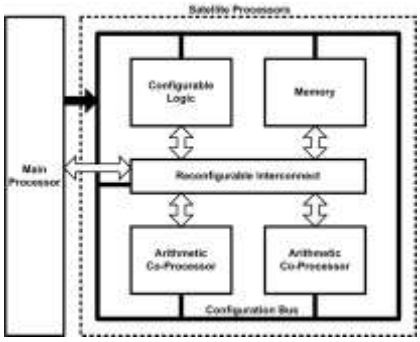

Fig. 14 Pleiades processor reconfigurable architecture layout [16]

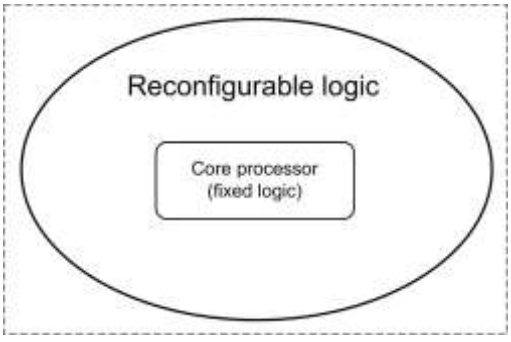

Fig. 15 OneChip architecture layout

the interconnect energy to a seventh of that of traditional crossbar network implementations [16]. This is achieved by having a universal switch box associated with each mesh level; in addition to cross-level interconnect switches so that only few buses are therefore required.

Targeted applications for this architecture are wireless devices and related baseband applications.

F. OneChip Architecture

The OneChip processor architecture is based on the combination of a fixed-logic processor core with large reconfigurable logic resources. This is illustrated in Fig. 15, and the idea is to offer large reconfigurable resources with the core processor. This is the classic processor and co-processor co-located approach. The main drawbacks of the processor and co-processor approach are the limitations on processor-coprocessor bandwidth and the rigidity in control and interaction of the coprocessor [5].

The OneChip architecture consists of the integration of a 32-bit core RISC processor surrounded by the reconfigurable logic resources which are tightly integrated into the processor pipeline. In this architecture there are two distinctive PEs, the basic functional unit (BFU) and the programmable functional unit (PFU). The BFU is responsible for basic functions, mainly arithmetic and logic operations; while PFU is more complex and can perform various functions in combinational or sequential form and in addition, it can work as glue logic whenever required.

It is worth mentioning that the OneChip architecture is an advanced version of the PRISC architecture [6]. The key difference is that the PRISC PFU only supports small combinational operations and is limited to one clock cycle operation. This leads applications of PRISC to be limited to bit-level applications. On the other hand, the OneChip architecture targets DSP applications.

G. Chimaera Architecture

The Chimaera architecture is based on the integration of reconfigurable logic into the host processor itself [7]. This allows direct access to the processor's register file and enables a set of new operands.

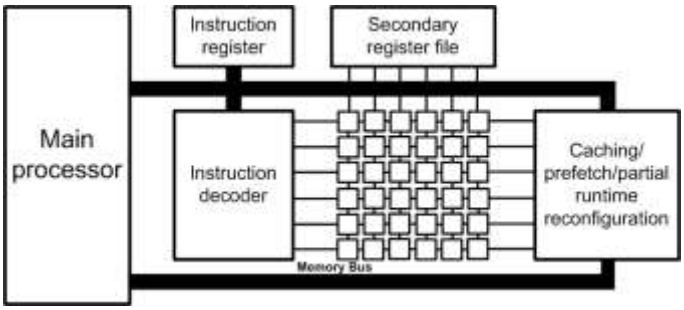

Fig. 16 Chimaera architecture layout

In addition, reconfigurable logic is always slower than the processor's own functional units when it comes to standard arithmetic computation. In the Chimaera architecture, the designers have integrated the advantages of the reconfigurable logic along with those of the processor. The processor is responsible for executing the bulk of the functionality while the most critical computation kernels are accelerated using the reconfigurable logic. Fig. 16 represents the Chimaera architecture layout with the reconfigurable logic at its heart. The architecture comprises a microprocessor with an embedded reconfigurable functional unit, which is described as a miniaturised FPGA array. The reconfigurable logic in this architecture is considered to be a cache for reconfigurable functional unit instructions. This architecture can be classified as fine-grained, due to its reconfigurable fabric being 1-bit based. An interesting aspect of this architecture is its instruction decoder, which supports multi-output functions and the efficient implementation of complex operations. Another interesting aspect is the availability of partial run-time reconfiguration, where the reconfigurable functional unit functions as an operation cache holding necessary instructions for the current operations. Many applications could potentially use the Chimaera, since the aim is for it to be a general purpose.

H. REMARC Architecture

REMARC stands for reconfigurable multimedia array coprocessor, and this architecture is a reconfigurable coprocessor coupled with a main RISC processor. The coprocessor includes a global control unit along with 64 programmable logic blocks or nano-processors, which are the PEs in this architecture [8].

The main processor has three coprocessors memory management and exception handling; a floating point processor; and the REMARC co-processor. Fig. 17 shows the architecture layout while Fig. 18 illustrates its internal construction of an 8x8 array of nano-processors and the global control unit. Each nano-processor has: 32 entry instruction RAM, a 16-bit ALU; 16-bit entry data RAM; 13x16-bit data registers; 4x16-bit data input registers; 1 instruction register; and a 16-bit data-out register.

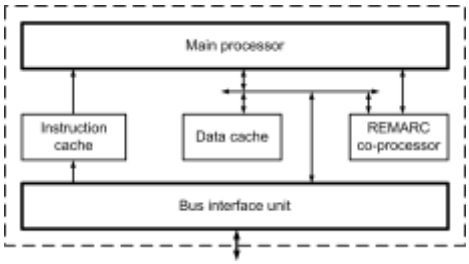

Fig. 17 REMARC Architecture layout

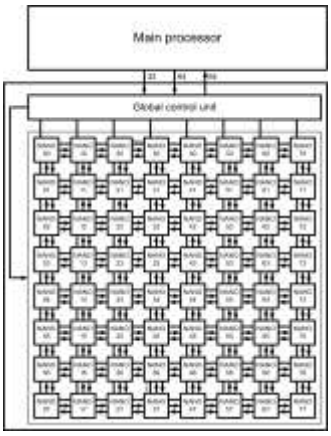

Fig. 18 REMARC Architecture Internal structure

Each nano-processor can communicate to the four adjacent nano-processors to the north, south, east and west. In addition, it can communicate with the processors in the same row and column through the 32-bit horizontal bus (HBUS) and the vertical bus (VBUS). The eight 32-bit VBUSes are also used for communication between the main the control unit and the nano-processors.

The nano-processors receive the program counter value from the global control unit, since it does not have its own program counter. Moreover, the function of the global control unit is to control the nano-processors and data transfer between the main processor and the nano-processors.

The REMARC architecture is a VLIW processor as its instructions consist of 64 operations. Its datapath is 16-bit and its targets multimedia applications such as image processing and video compression [8].

I. RaPiD Architecture

Reconfigurable Pipelined Datapath (RaPiD) is a course-grained FPGA architecture, which is designed for computing intensive, repetitive tasks where configuration supports computation pipelines [9].

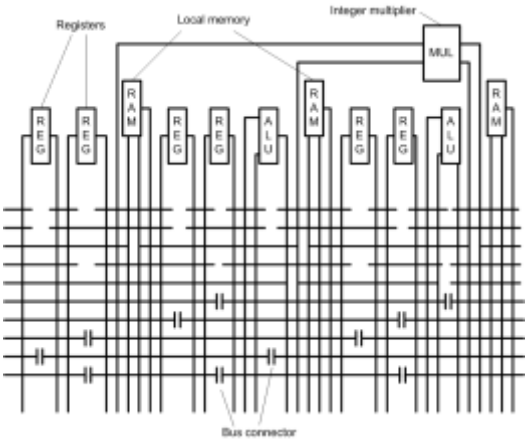

Fig. 19 RaPiD-I basic cell structure

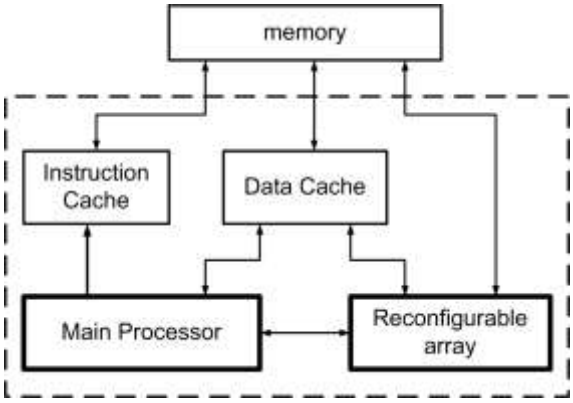

Fig. 20 Garp Architecture block diagram

The architecture consists of an application-specific datapath and the program for controlling it. The interconnections are based on a linear array, in nearest neighbour style. The processor's functional units are placed in a linear array and formed of identical cells.

RaPiD-I is a prototype developed by the University of Washington, in which the cells or PEs consist of an integer multiplier, two integer ALUs, six general-purpose registers and three small local memories. The cells are interconnected with a series of buses as demonstrated in Fig. 19. The inputs and outputs of the cells have multiplexers and de-multiplexers respectively to identify the specific buses to receive and send data.

RaPiD-I operates on 16-bit data. The datapath has registers which are used to store constant or intermediate values; however, these registers are costly in terms of area and utilisation. The control signals in this architecture are divided into static and dynamic signals. The former are used for initialisation and pipeline construction, while the latter are used for scheduling computation information and are changeable in every cycle.

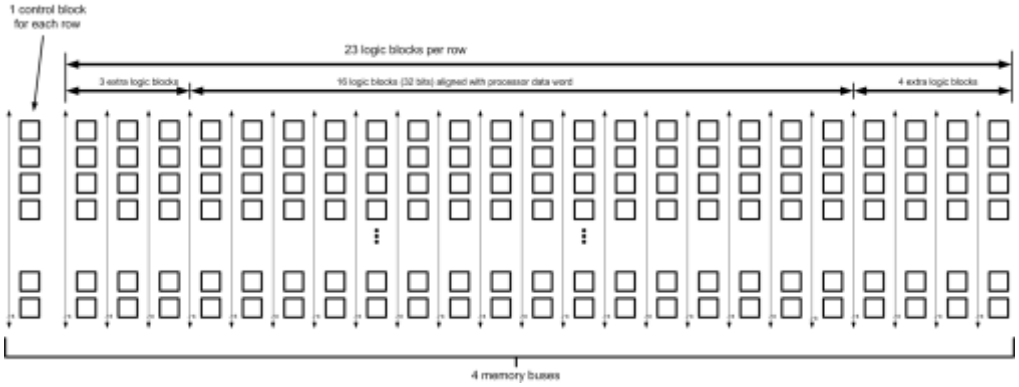

Fig. 21 Garp array organisation

Around 34% of the cell control signals in the RaPiD-I are dynamic while the rest are static signals, giving a total of 230 control signals per cell.

The RaPiD is not suited for non-highly repetitive algorithms or those whose control flow is strongly dependent on data, such as in error correction, image processing or data encoding.

J. Garp Architecture

The Garp architecture is based on a combination of reconfigurable hardware with a standard MIPS processor [17]. This means that the Garp is reconfigurable architecture as a co-processor for executing certain parts of the code which are slower when running on the MIPS. A block diagram is presented in Fig. 20.

The Garp's reconfigurable array is used to speed up functions, loops or subroutines that would be slower if run on the main processor. The reconfigurable array is fully controlled by the program running on the main processor.

The main processor instruction set has been extended for Garp. The processing element in the reconfigurable array is called a 'block.' The block is a logic unit which resembles the FPGA; however, at the start of the array row the first block is a control unit as illustrated in Fig. 21. The array has a restricted 24 columns, while the number of rows is application-specific with a minimum of 32. The architecture's datapath is 2 bits, and thus to give a 16-bit operations at least eight logic blocks will be required. Usually these blocks are combined in a linear style in the same row.

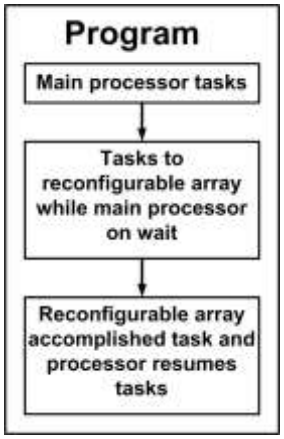

Fig. 22 Garp program flowchart [17]

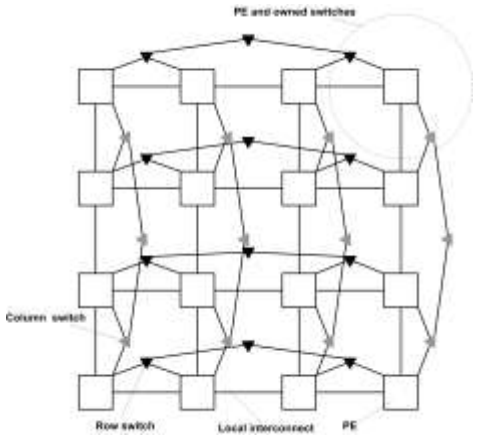

Fig. 23 SRGA architecture interconnect mesh [11]

The array has four vertical buses for interaction with the memory for the reception or transmission of data. In addition, these buses are used for array reconfiguration. There is an additional wiring network for data transfer between the different blocks.

An innovative feature within this architecture is the availability of cache units within the array blocks, which hold recently used configurations in order to minimise costly memory access and allow faster switching between reconfigurations.

The Garp architecture has two clocks, one for the main processor and another for the reconfigurable array. The control block at the end of each row works as the interface control between the array and the main processor or main memory, and it can even interrupt the main processor. Each block requires 64 configuration bits in order to be fully configured. The configuration of the whole array is a lengthy and costly process in terms of energy and waiting time without execution. The

latter is assumed by the architecture's authors to be 50μs in order to complete the configuration load. An interesting aspect of this architecture is the ability of the array to be partially reconfigured if only partial usage of the array is required. The architecture's minimum configuration is for a single row.

K. SRGA Architecture

The self-reconfigurable gate array (SRGA) architecture consists of an array of processing elements [11]. The processing element consists of a logic cell and memory block. The logic cell contains a 16-bit LUT and a flip-flop. The memory block can store one or several configuration contexts as well as data for the logic.

Processing elements are connected to their four nearest neighbours in addition to the mesh of the tree network. With this network, context switching and memory access operations can be performed in a single clock cycle. Each PE has two switches a row switch and a column switch, as demonstrated in Fig. 23.

Self-reconfiguration in this architecture is mainly used to allow each logic cell to modify its own configuration at run time. Therefore, instead of having an external or centralised reconfiguration controller, this architecture has a distributed reconfiguration capability integrated within each PE. This gives the device fast memory access and context switching.

To achieve self-configuration, a very complex interconnection network is required which consists of a logic interconnection network and a memory interconnection network, in addition to the switching network at each node.

L. CHESS Architecture

The CHESS architecture was developed by Hewlett-Packard (HP) laboratories targeting multimedia applications. This architecture is intended to be an ASIC IP or a unit of the processor datapath [10].

The PE in this architecture is a 4-bit ALU with a set of 16 primary instructions. The instructions can be constant and stored within the configuration word or dynamic and generated through an external circuitry feeding into the instruction input of the ALU or PE.

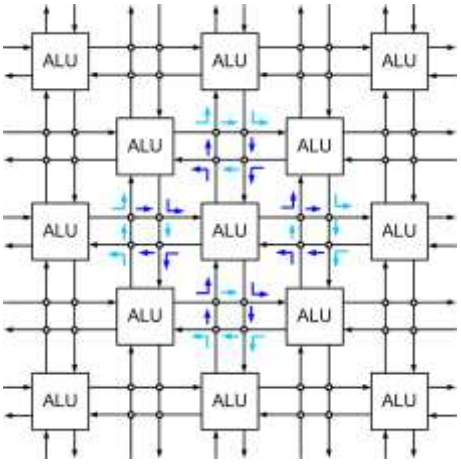

Fig. 24 CHESS architecture layout and neighbouring interconnections

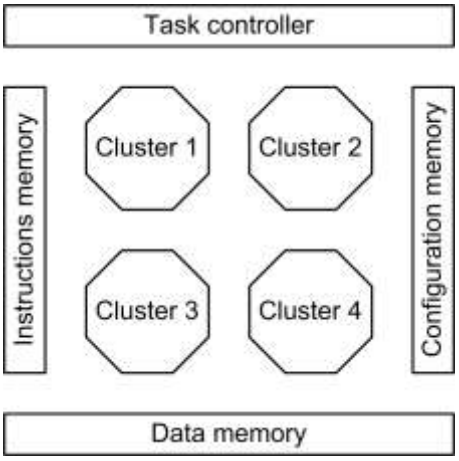

Fig. 25 DART system level architecture

The datapath of this architecture is 4-bit, allowing much flexibility and a wider range of datapath applications compared with 1-bit architectures (FPGA).

The architecture has a switch box which has a dual functions based on the mode of operation. First it may act as a cross-point switch, allowing 64 connections by connecting vertical and horizontal buses. Secondly, it may be a RAM of 16 words x 4 bits using the 64-bits configuration.

The architecture's layout has PEs arranged as a chessboard in a symmetrical fashion. This increases neighbourhood connectivity, and also reduces the routing network complexity due to the maximisation of the number of neighbours in close proximity to each other, as illustrated in Fig. 24. This proximity allows each ALU to have input and output buses on all four sides, enabling data transmission and reception from any of the eight surrounding ALUs. In addition, the architecture provides embedded evenly distributed block RAMs of 256W x 8-bits per 16 ALUs. The pipelining support increases throughput and efficiency of the architecture. There are two registers or buffers for each switchbox, and this is particularly helpful for long connections in order to avoid limiting the clock speeds of the entire architecture. Each PE has 100 configuration bits, which allows fast reconfiguration.

The architecture does not support partial reconfiguration; however, it has the usual offline reconfiguration in addition to the

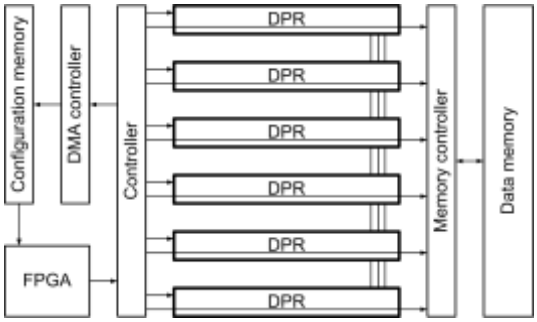

Fig. 26 DART's cluster construction

ability to alter changes in functionality at runtime through feeding instructions into the ALU in a cycle-by-cycle approach.

One of this architecture's advantages is the capability of the switch boxes to act as memory (RAM) of a reasonable size. In other architectures, the conventional technique is to use the PE's own configuration bit as a memory, and this is usually small and thus limits the architecture's capability. The PE or ALU can be combined with an adjacent switch box to provide a 16W x 4-bit memory. In this architecture, the routing scheme is large since it uses 50% of the array area; however, this is less than in most FPGAs.

M. DART Architecture

The DART architecture is intended to be a reconfigurable architecture for telecommunications applications. The authors claimed that the architecture can handle complex processing tasks of third generation telecommunications systems in an efficient and low-power manner [17]. The architecture can be broken down into independent processing units named clusters. Those clusters can work independently or in cooperation with other clusters, as illustrated in Fig. 25. The top level or cluster level architecture shows the main controller that is responsible for distributing tasks to specific clusters to execute. Then each cluster has its own embedded controller to manage the internal processing of the task allocated to it.

This architecture uses a hierarchical interconnect network, which is more suitable, smaller and less complex than a global interconnect network.

Fig. 26 shows the interesting feature of the DART architecture that it has two PE types inside each cluster; a reconfigurable datapath (DPR) and an FPGA core.

Each cluster consists of one FPGA core and six arithmetic processing primitives (DPR). Each DPR has four functional units of two ALUs and two multipliers. The functional units are dynamically reconfigurable. As illustrated in Fig. 26, there is an interconnection between the DPRs, so they can be configured to work together or work independently (in parallel).

In this architecture, there are three modes of reconfiguration. In dynamic reconfiguration mode, the interconnections within the cluster are reconfigured according to the calculation pattern. In hardware reconfiguration mode, this is the kernel configuration and it takes four cycles and requires a large amount of data; this is the regular ongoing configuration process. Software reconfiguration mode concerns only the functionality of operators and is used for irregular processing where the DPR configuration needs to be changed.

This is clearly a very interesting architecture on various levels: firstly in its combination of FPGA and reconfigurable processing elements; secondly for the reconfiguration modes available, including dynamic reconfiguration; and thirdly given that its area of application is in the same domain as that addressed in this work.

N. DReAM Architecture

The DReAM (dynamically reconfigurable hardware architecture for mobile communication systems) is a coarse-grained architecture dedicated for wireless communication applications [11]. This architecture has been designed to work within a system or system-on-chip, which means that the architecture would require a DSP, memory, controllers, and so on. The architecture has been developed at the Darmstadt University of Technology.

As illustrated in Fig. 27 the DReAM architecture consists of an array of reconfigurable processing units (RPUs) interconnected through local and global connections. In addition, there are dedicated input and output ports at the borders of the architecture for data and control interfaces. Within the array, each RPU is able to connect directly to its nearest four neighbours through the local communication network. Moreover, within the array each four processing units share one configuration memory unit.

The RPU is able to execute data-flow arithmetic, data manipulations and finite state machines for the control-flow. Each RPU contains two reconfigurable arithmetic processing units, one spreading datapath unit, two dual-port RAMs and one communication protocol controller. An interesting aspect of this architecture is the dedication of a specific unit 'the spreading

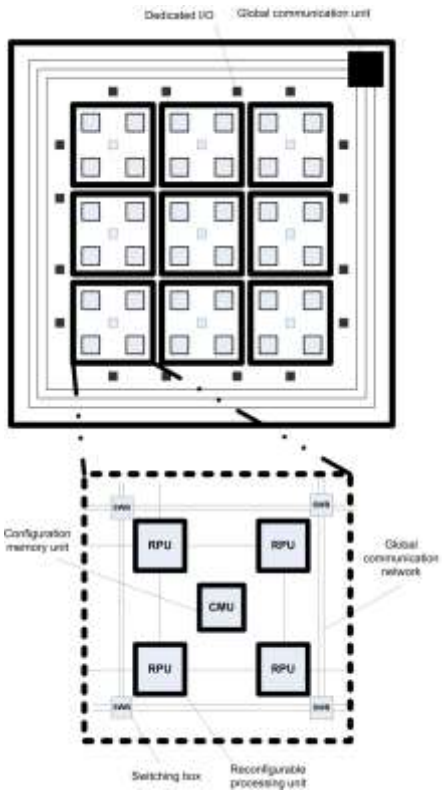

Fig. 27 DReAM architecture structure

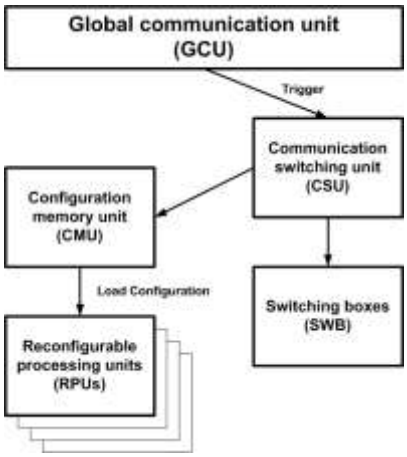

Fig. 28 DReAM hierarchy control for dynamic reconfiguration

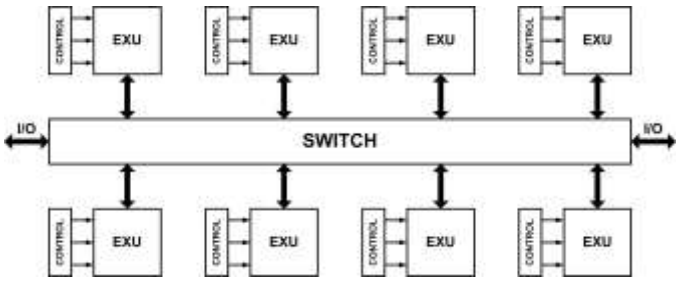

Fig. 29 PADDI architecture structure

datapath unit' to process correlation operations for the communication standards such as quadrature phase shift keying (QPSK), bi-phase shift keying (BPSK) and required for code division multiple access (CDMA) systems. The correlation process is based on generating PN codes within the unit, which require a PN-code generator. This makes the architecture suitable for the communication applications targeted. However, having such dedicated and fixed units in every RPU within the array, despite the fact that it may be neither used nor required, represents a waste of resources in terms of area and power.

The RPU is the processing element in this architecture and the architecture's datapath is 8-bit based. Despite this 8-bit limitation, the PRU unit is capable of addressing a higher number of operands through the manipulation of the RAM as a LUT.

The architecture is built on a hierarchical concept where the global communication unit is the controller for the whole architecture as illustrated in Fig. 28. The designers claimed that they used this approach as it offers a trade-off between area and configuration performance. However, the hierarchical concept not only adds complexity to both the system and its control, but it also increases its area and power consumption. It is possible for the dynamic reconfiguration in this architecture to occur in different scenarios, including during run-time by conducting partial reconfiguration.

O. PADDI Architecture

PADDI stands for programmable arithmetic devices for high speed digital signal processing architecture. The PADDI architecture was first introduced in 1990 [20]. The architecture targets the rapid prototyping of high-speed data paths for real-time digital signal processing applications.

The PADDI architecture contains a cluster of eight EXUs or processing elements [21]. The interconnection between the EXUs is a crossbar-based network termed a switch, which is dynamically reconfigurable, as shown in Fig. 29.

EXUs can be configured into two modes, 16 or 32-bit wide. Each EXU contains two register files each of which contains six registers. Registers are used for temporary data buffering, and the register files have dual ports for simultaneous read and write operations. Fig. 30 represents the internal architecture of the EXU or processing element.

Each EXU needs a 53-bits control word. The Nanostore holds words of 8 bits necessary for controlling the eight EXU units. A global controller is responsible for feeding or loading the instruction word into each Nano-unit. The programming for this processor takes place using a high-level data flow language, 'Silage' [22].

P. MorphoSys Architecture

MorphoSys is a reconfigurable architecture developed at the University of California targeting computation-intensive and high-throughput applications [23]. The architecture comprises a reconfigurable array, core processor and memory interface as

illustrated in Fig. 31. The reconfigurable array acts as a SIMD coprocessor and is responsible for exploiting the parallelism available in the application's algorithm.

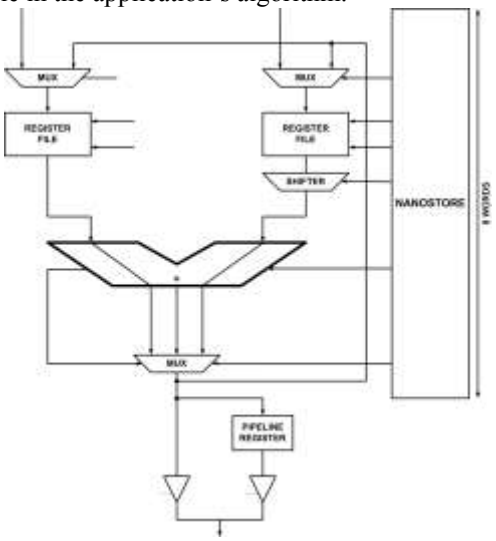

Fig. 30 PADDI's EXU's architecture

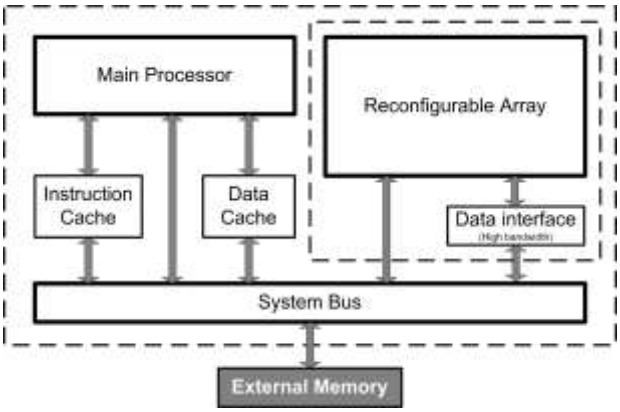

Fig. 31 MorphoSys architecture layout

The key component in this architecture is the reconfigurable cell array. The array consists of 8 x 8 reconfigurable cells (RCs), each of which consists of an ALU, a multiplier, a shifter and a register file and it is configured by a 32-bit context word stored within the array context memory. Each RC is connected to all of its neighbours in the same quadrant in both row and column directions in addition to the interconnection between the neighbouring quadrants as shown in Fig. 32. All eight RCs in the same column or row are configured by the same context word; however, each operates on different data.

Dynamic reconfiguration is supported in this architecture and takes place through having context data loaded into an inactive part of the context memory without interrupting the array operation. Interconnectivity here within the array is based on the use of a 2D mesh and hierarchical bus network.

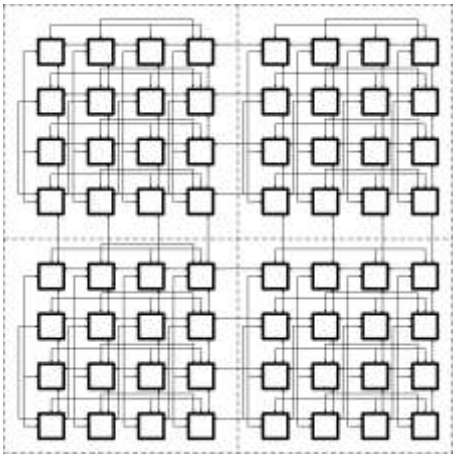

Fig. 32 MorphoSys 8x8 reconfigurable cells array and the row-column connectivity between each reconfigurable cell (RC)

The MorphoSys system can operate on 8 or 16-bit data, despite the fact that the architecture's RISC processor is 32-bit. This architecture has some clear features and advantages that have been highlighted earlier; however, there are also some drawbacks. The extensive use of multilevel memories has a significant effect on the processing times and power consumption.

Moreover, the processing element or RC is complex and sophisticated which implies a significant effect on the architecture's area and power consumption. In addition, all RC units in a single row or column have the same functionality. Although one of the reasons for this was to try to limit the interconnectivity overheads; however, this dramatically limits the architecture capabilities and flexibility and leads to a reduction in the range of applications that can be executed.

Q. PipeRench Architecture

The PipeRench architecture use pipeline reconfiguration as its main concept [24]. The architecture is composed of a set of pipeline stages, rows or stripes, as illustrated in Fig. 33. Each stripe consists of processing elements up to N and an interconnect network. Each processing element contains registers and ALUs. The ALU is based on a look-up-table, and. PEs can interact with each other within the same stripe but not adjacent ones.

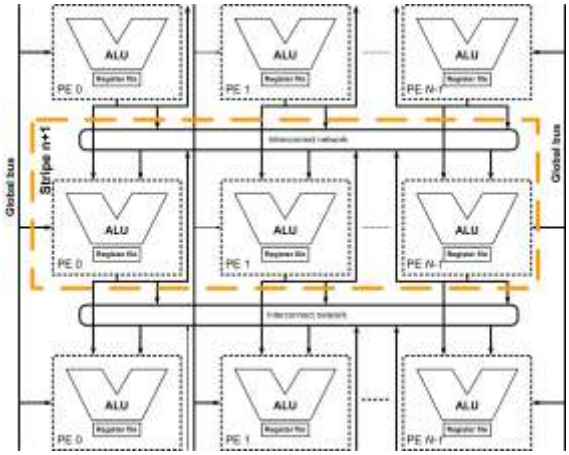

Fig. 33 PipeRench architecture layout

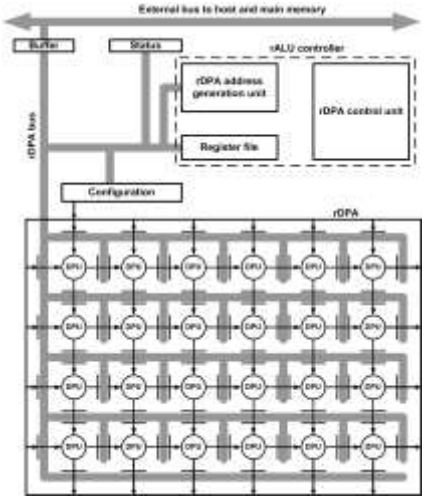

Fig. 34 rDPA architecture with ALU controller

This architecture has two types of interconnections: a local network where all PEs within the same stripe can share some data and have local transfer; and a global network where a PE in a stripe can read data from the output register of the above stripe. Each PE is 8-bits based while the whole stripe totals 128-bit in width. This means that there are 16 PEs per stripe.

The architecture's principle of operation in having two levels of configuration for the PE and the stripe is novel. However, it is too hardware-oriented and overly focused on the pipeline approach. This limits the applications that this architecture can handle. Moreover, it appears that the implementation of such an architecture will be costly in terms of area, power and performance.

R. rDPA Architecture

The processing elements in the reconfigurable datapath architecture (rDPA) are called datapath units (DPU) [25]-[27]. The rDPA is reconfigurable in-circuit and is scalable to large array sizes. The architecture has an added controller called the reconfigurable ALU (rALU) which allows the architecture to be data-driven. The rALU provides the architecture with the capability for the parallel and pipeline computation of complex expressions. The rDPA architecture consists of an array of reconfigurable processing elements or DPUs, as illustrated in Fig. 34. The number of DPUs within the array can be up to 128. The elements are connected using a mesh type interconnect network. The architecture has two interconnection levels: global interconnection through longer lines and local interconnection through shorter lines.

As illustrated in Fig. 34, the rALU consists of an rDPA control unit, an rDPA address generation unit and a register file. A data-driven reconfigurable ALU is the result of having rALU with rDPA within the architecture. The register file has 64 x 32-bit registers used for holding intermediate data in order to reduce the multiple reading of data from the memory. The execution of the whole architecture is data-driven, including the configuration process. The rDPA control unit holds the instructions sets and delivers instructions to the designated DPU within the rDPA. The architecture is uses a single I/O bus to connect all datapath units using multiplexing; however, the designers hinted that the architecture would benefit from two buses to speed up I/O operations [25].

S. KressArray Architecture

The KressArray architecture or KressArray-III is a 32-bit-based coarse-grain architecture [25]-[27]. The processing elements of this architecture are called reconfigurable datapath units or rDPUs. Fig. 35 represents the structure of the architecture's layout with its interconnection network. Local interconnections are used to feed data directly to the designated PE or rDPU or to read the resulting data. In addition, they can be used to pass intermediate results from one rDPU to another. The hierarchical interconnects allows this flexibility. The architecture has nine PEs and 32-bit duplex connections in four directions, north, east, west and south (NEWS). The direction of the dataflow through the connection is programmable.

An interesting feature in this architecture is its capability to allow the rDPU to be reconfigurable as a router. Usually architectures have dedicated switches to reconfigure interconnections, and these switches act as routers. This is a compromise from the architectural design point of view, since having a complete rDPU acting like a router is a waste of valuable computation resources. However, the authors claimed that the rDPU can be split into a partial router (routing only limited number of connections; usually one) and a processing element. They suggest that, in partial mode, the processing capabilities remain intact and the PE can fully utilise them. Nevertheless; it is unclear weather or not what has been proposed by the authors is a full integration between the PE and the switch in a single unit called the rDPU. This may have implications for the area, power consumption and flexibility of the system, and could complicate the programming of the system as well.

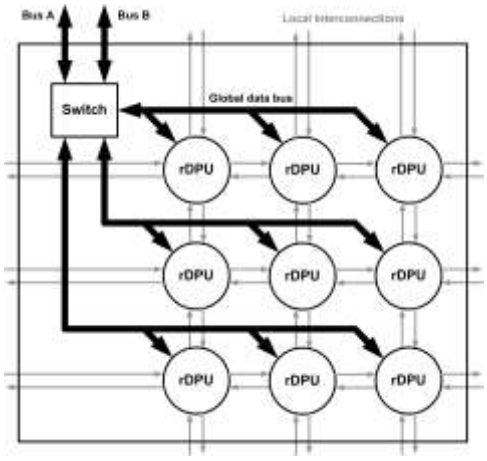

Fig. 35 KressArray architecture

Another interesting feature of this architecture is that the configuration memory consists of four independent layers. In addition, the register file within each rDPU has four configuration layers, as illustrated in Fig. 36, to hold the four complete configuration sets. All rDPUs within the architecture simultaneously change the actual configuration memory layer. The aim of these layers is to minimise or eliminate the reconfiguration time of the array, since there will be only one active configuration layer at any given time while the others are idle. The system can reprogram those layers, given that the configuration data and configuration control buses are independent. The elimination of reconfiguration time is a great step forward towards real-time reconfiguration.

A key drawback of having four layers is the need for four times the size of the configuration memory and registers, which will reflect negatively on the architecture area and, most importantly, power consumption. It is worth noting that the KressArray architecture is meant to be a co-processor or accelerator. Moreover, this architecture has a strong tool set that supports various optimisations for the algorithms implemented in seeking the best performance level.

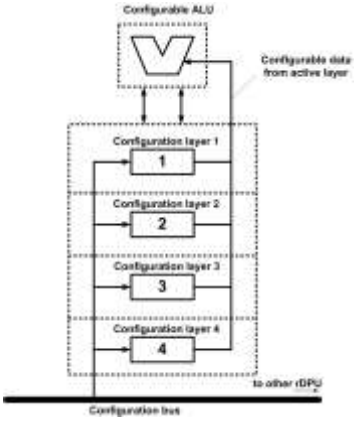

Fig. 36 rDPU's four configuration layers

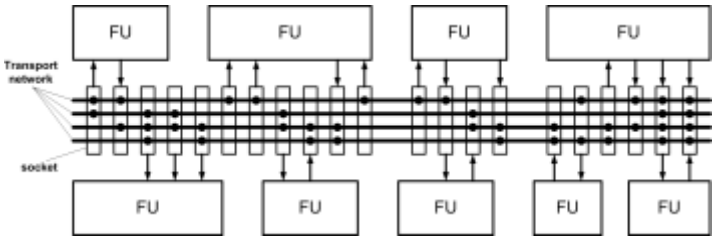

Fig. 37 Move architecture's structure

T. MOVE Architecture

Any thorough review of reconfigurable architectures would usually consider some or all of those discussed above. However, transport-triggered architectures (TTAs) are usually missing from such studies. However, this is such an important type of architecture that it must be taken into consideration. Arguably, the TTA may or may not be considered to be a truly reconfigurable architecture; however, from the point of view of performance and flexibility, it is significant and deserves to be considered among the reconfigurable architectures rather than as a general-purpose processor.

The transport-triggered programming paradigm was developed at the Delft University of Technology [28]-[30]. The paradigm was changed from 'operation triggered' to 'transport triggered', and the realisation of this paradigm is an architecture called the MOVE32INT. The key feature and main principle in this architecture is the reduction of the instruction set to only one operand, which is the 'MOVE' function; hence, the name of the architecture.

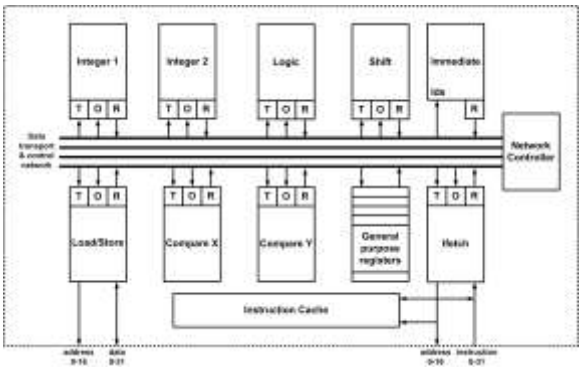

Fig. 38 MOVE32INT block diagram

TABLE 1 MAIN TYPES OF INSTRUCTION CELLS TYPES IN THE RICA ARCHITECTURE

Instruction Cell	Operations executed
ADD	Addition/subtraction
COMP	Compare two values
DIV	Signed/unsigned divisions
I/O REG	Register with access to I/O ports
JUMP	Branches/ step end
LOGIC	Logic operations (AND, OR, XOR, etc.)
MEM	Data memory READ/WRITE functions
MUL	Signed/unsigned multiplications
MUX	Multiplexer/simple branching
REG	Register
RRC	Reconfiguration rate controller
SHIFT	Shift logic operation

Fig. 37 shows the architecture's structure where the focus is the transport network and the processing elements or the functional units (FU) are distributed along the network. There are sockets that define the connection between FU and the network which can be either input or output from the function unit. The functions of the FU can range from being a single operation unit (operand), to a complete ALU, and it can also accommodate internal pipelining. Each FU has a register at its output named the result register. An interesting aspect here is that the FUs in general are heterogeneous and designed to fit targeted areas of application or can be narrowed down to suit a specific algorithm.

It is worth mentioning that there is a clear separation in this architecture between operations and transport.

As can be seen from Fig. 38, there are four types of registers within the architecture: 'O' is the operand register; 'T' is the trigger register, 'R' is the result register and finally 'r' is the general purpose register.

Operations within the FU will kick-start once the trigger register T is loaded. The cycle time of this architecture is determined by data transport.

The MOVE32INT is a 32-bit-based processor, which uses Harvard architecture with separate data and address paths to memory. The processor is capable of four concurrent data transports per clock.

It is clear that this architecture is conceptually interesting, allowing high enough flexibility for it to be a programmable processor. Despite all of its interesting features, however the architecture has a key drawback which is the complicated process of code compilation. When the MOVE was compared with VLIW in DSP applications [31], it was noted that MOVE has a much lower code density resulting in a larger code size. This is clearly one of the main drawbacks of the architecture.

Another upgraded processor has been developed to overcome some of these drawbacks, namely the MOVE-Pro [32]. This new processor is based on the TTA architectural concept; however, it is built with power savings as a major driver along with increasing code density.

Move-Pro promises significant dynamic power savings through the reduction of the number of accesses needed to the register file. Key changes from the earlier processor are the addition of jump and branch instructions to the instruction set. The authors concluded that the new processor achieves 80% savings in register file access and a total of 11% reduction in power consumption compared to the older version [32].

Despite the two processor versions, TTA has some clear advantages such as modularity, flexibility and scalability. TTA architectures may have a future as reconfigurable architectures; however, this would require more time and effort from the developers. Instead of trying to compare it with RISC processor, it may be worth looking further into moving the TTA architecture into the reconfigurable architecture arena.

U. RICA Architecture

The reconfigurable instruction cell array (RICA) architecture was developed at the University of Edinburgh [33]. This architecture is based on having an array of customizable instruction cells or processing elements. A unique feature is that the architecture's processing elements or instruction cells are heterogeneous. table 1 lists the different types of instruction cells.

The main concept behind the RICA is the processor, which is able to handle the control and dataflow aspects of applications. This handling is flexible, maximises utilisation, and supports parallel processing and low-power consumption. Thus, the RICA architecture is characterised by a high performance high parallelism and has a processor with low power consumption and small area. It is highly flexible being coarse-grain and scalable which allows it to be adapted to the application required by using the most suitable combination of types and numbers of instruction cells.

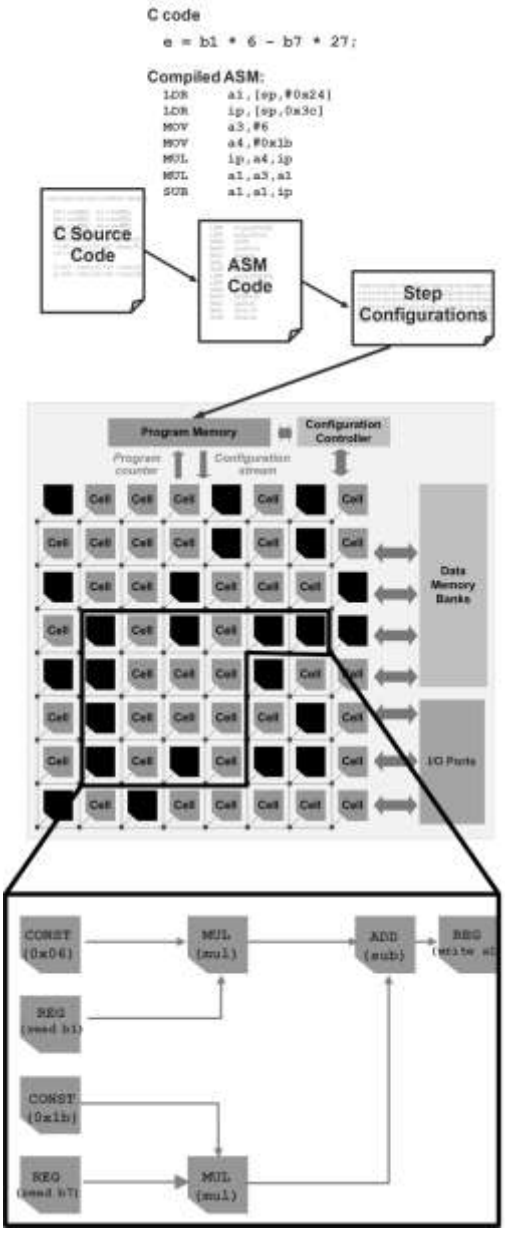

Fig. 39 Dynamic allocation of instruction cells into processing steps, scheduled within the GCC tool chain

From a programming point of view, this processor has the key advantage of DSP processors of being programmable using the ANSI-C programming language, and its tool flow can be designed using GNU C-compilers, which is familiar to programmers. Therefore the architecture will not need a skilled hardware engineer with experience in HDL languages, but on the contrary, C-programmers will be able to efficiently use and program this processor, thus saving time and resources. When the algorithm program is compiled, the resulting assembly code is then sliced into blocks of instructions, and each block is executed in a single step. The step size is defined by the resources available within the architecture and the number of read/write operations included. Therefore, the algorithm will be executed in steps, which allows the architecture to be dynamically adaptable to each step. Each step can have a different critical path, where the clock that controls the program

counter and memory is reconfigurable. The programmable clock allows the architecture to provide optimum performance, by providing the maximum performance level for each step, so that maximum performance or throughput for the whole application is guaranteed. The JUMP instruction is used as a trigger for the architecture to load the next configuration or step. If the step contains a full loop of instructions, this means that the processor will avoid any reconfigurations. Only the data will be loaded from the memory or registers, giving the processor a great advantage over other architectures. Fig. 39 illustrates how a single C-code instruction is compiled and mapped on the architecture cells. This allocation or mapping changes with every code or set of codes.

Various applications have been implemented on the RICA and customised versions show high performance with significantly lower power consumption [34]-[37].

There are two features which the RICA and TTA architectures have in common, discussed earlier in section T. Firstly, both are architectures based on heterogeneous processing elements. The RICA has heterogeneous instruction cells while the TTA architecture has heterogeneous functional units. Secondly, both are claimed to be processors or processor-like architectures with Harvard-like structures. In other words, both can run independently without the need for an external processor to fetch instructions or synchronise operations. However, the drawbacks of the TTA discussed earlier, including its complex register file structure, do not apply to the RICA. Moreover, the RICA is designed from the ground up based on two key features, which are low power consumption and an architecture which is easy to program.

V. CDDS Variable Datapath Architecture

A control-flow driven data-flow switching (CDDS) variable data architecture has been introduced at by Hokkaido University, which is characterised by flexibility and low energy consumption as cited by the authors [38]. The authors achieved the balance between performance and power consumption was achieved by limiting the scope of dynamic reconfiguration. This architecture is aimed at control-intensive programs, and its datapath is divided into static and dynamic sections. Only the dynamic part is allowed to be dynamically reconfigurable at run time. Fig. 40 illustrates the split in dynamic reconfiguration the architecture datapath. The architecture is designed to work mainly as an accelerator beside the main processor.

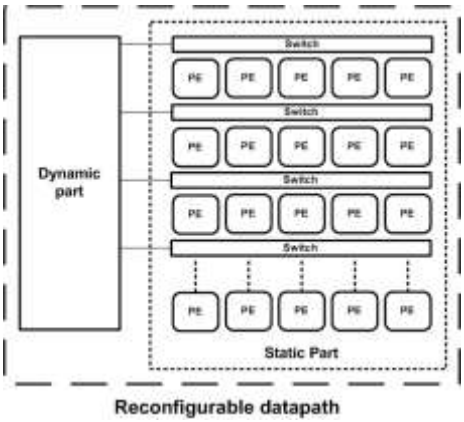

Fig. 40 CDDS architecture's reconfigurable datapath separated into static and dynamic parts [38]

The PEs in this architecture are ALU which are asymmetric to those of the main processor in the whole system, in order to streamline the architecture's programming and mapping.

This architecture is similar to RICA in that the key parameter for switching the dynamic reconfiguration is the branch, which in this case matches the RICA jump instruction.

The architecture has a clear novelty in restricting dynamic reconfiguration and memory access during execution for the static portions or PEs, while this is allowed for switching. This approach may have clear advantages in terms of power consumption; however, it has limitations in terms of being suitable for a restricted range of applications and needing a larger area in order to cater for the multiple branch options on the array so as to proceed with reconfiguration. This architecture is still in its early days, and may evolve when significant algorithms are implemented with it.

W. BilRC Architecture

The PEs of a execution-triggered coarse-grain reconfigurable architecture entitled BilRC [39] are inspired by the FPGA. The PEs in BilRC are of three types: ALU, memory and multiplier. The PEs are realised in columns of the same type as illustrated in Fig. 41 .

The applications this architecture is intended for span in a wide range, from signal processing to telecommunications. However it is clear that PEs the distribution of PEs will change depending on the application domain targeted. An interesting

aspect of this architecture is that the authors used the MUX instruction for transportation within the architecture, which is similar to the MOVE instruction in the MOVE32int architecture.

Key points of this architecture are that it is static and not dynamically reconfigured; moreover, it is programmed through a special language developed by the authors called LRC. Despite the high reported performance compared to FPGAs and dedicated DSPs, no comparison of power consumption was mentioned. This may be expected to be addressed in future publications.

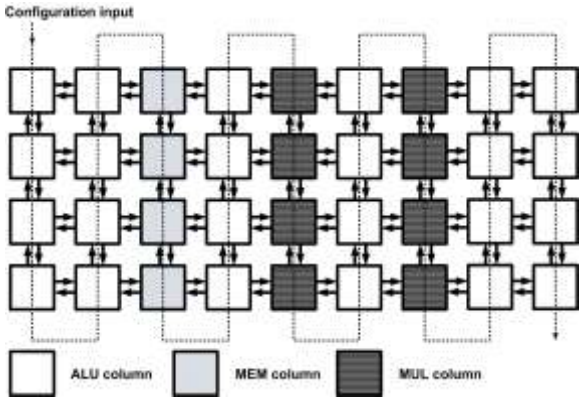

Fig. 41 PE column based structure in BilRC [39]

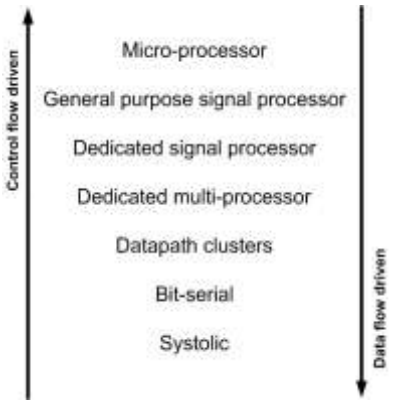

Fig. 42 Classification based on control/arithmetic ratio [40]

IV. COMPARISON AND DISCUSSION

Reconfigurable architectures can be classified based on various criteria, including datapath width, type of PE interconnection, reconfiguration model, programming language, placement and routing.

One classification can be based on the control/arithmetic ratio. Brodersen and et.al. [40] categorised architectures based on the amount of sharing operations on an arithmetic unit. As seen in Fig. 42, all operations in the micro-processor of the general purpose ALU are all operation time-multiplexed; hence, it is a control-driven architecture. On the other hand, a systolic array, for example, has each operation represented by separate hardware with minimal control. The best outcome would be the right balance between control and datapath for a given application and throughput.

For a reconfigurable architecture to be capable of carrying out the tasks of telecommunications system efficiently, various criteria would need to be satisfied. Datapath width or granularity is one of the key features of such architectures, and should be between 8 to 32 bits. This range would be suitable for telecommunications applications today and in the near future. It is anticipated that 64 bits would be desirable for future systems. Furthermore, it is desirable for reconfigurable architecture to have heterogeneous rather than homogenous PEs, for two reasons. Firstly, this will allow enough flexibility to accommodate the challenging functions of telecommunications systems. Secondly, it will most probably involve lower power consumption due to the high utilisation of the resources in the architecture. Another key aspect to consider is the ability of the reconfigurable architecture to work as a standalone unit and not just as a co-processor or an extra functional unit for a main processor. This is the key for telecommunications systems, from the point of view of efficiency, optimisation and power consumption.

Various efforts have been made to compare reconfigurable architectures and in particular the coarse-grain architectures [41]. The present study focuses on reconfigurable architectures suitable for telecommunications systems.

A comprehensive comparison of the various architectures is provided in table 2. Here the approach used with each processor can be clearly identified in terms of datapath width, and level of supported reconfiguration in terms of whether it supports dynamic reconfiguration or is limited to static reconfiguration. In addition, it should be noted whether the architecture is capable of being standalone or if it requires an external processor. Moreover, key differentiation among PEs in an architecture is whether they are homogeneous or heterogeneous. The architectures considered were built in targeting specific applications, which are also indicated in table 2.

Most of the reconfigurable architectures are designed to act as co-processors, except for the RICA, MOVE, BilRC and rDPA. Another important criterion is the nature of the reconfigurable cells or PEs. Most of the architectures are based on homogeneous PEs, except for the RICA, CRISP, Pleiades, OneChip, MOVE and BilRC architectures which have heterogeneous PEs. Dynamic configuration is another key feature that is most desirable for telecommunications systems, as highlighted earlier. Several applications are highlighted in table 2 which support dynamic reconfiguration, either partially or fully.

TABLE 2 COMPARISON OF STUDIES OF RECONFIGURABLE ARCHITECTURES

Architecture	PEs Homogeneous/ Heterogeneous	Datapath Width	Reconfiguration. Dynamic/Static	Application	Role
BilRC	Heterogeneous	16-bit	Static	General	Standalone
CDDS	Homogeneous	32-bit	Dynamic	DSP	Coprocessor
CHESS	Homogeneous	4-bit	Static	Multimedia.	Coprocessor
Chimaera	Homogeneous	16-bit	Static	DSP	Coprocessor
CRISP	Heterogeneous	8-bit	Static	Multimedia.	Coprocessor
DART	Homogeneous	8-bit	Dynamic	Telecomm.	Coprocessor
DREAM	Homogeneous	8-bit	Dynamic	Telecomm.	Coprocessor
Garp	Homogeneous	2-bit	Static	DSP	Coprocessor
KressArray	Homogeneous	32-bit	Static	General	Coprocessor
MATRIX	Homogeneous	8-bit	Dynamic	General	Coprocessor
Morphosys	Homogeneous	8-bit	Dynamic	General	Coprocessor
MOVE	Heterogeneous	32-bit	Static	DSP	Standalone
OneChip	Heterogeneous	32-bit	Static	DSP	Coprocessor
PADDI	Homogeneous	16-bit	Dynamic	DSP	Coprocessor
PipeRench	Homogeneous	8-bit	Static	DSP	Coprocessor
Pleiades	Homogeneous	8-bit	Static	Telecomm.	Coprocessor
RaPiD	Homogeneous	16-bit	Static	DSP	Coprocessor
rDPA	Homogeneous	32-bit	Dynamic	Telecom	Standalone
REMARC	Homogeneous	16-bit	Static	DSP	Coprocessor
RICA	Heterogeneous	32-bit	Dynamic	DSP	Standalone
SRGA	Homogeneous	2-bit	Dynamic	General	Coprocessor
Systolic Ring	Homogeneous	8-bit	Static	General	Coprocessor
vCell Matrix	Homogeneous	4-bit	Static	General	Coprocessor

V. CONCLUSION

A reconfigurable architecture which can meet the challenging requirement of communications systems has to have many crucial characteristics. Primarily it has to operate with low power consumption. In order to sustain this, the use of heterogeneous PEs is the best approach. PEs can be customised specifically to the system's needs, resulting in the highest utilisation, which will lead to lower power consumption and smaller area.

It appears from table 2 that the most suitable architectures for telecommunications systems are the MOVE, BilRC and RICA. BilRC is disregarded here since it uses a new nonstandard programming language.

The MOVE and RICA are very different architectures; however, they are similar in that the PEs used are being heterogeneous, both are standalone and do not need external processors for control, and are programmable using C-language.

However, the RICA appears to be superior in terms of power consumption, the processor has been built with low power use as its core principle. The MOVE designers only started to address power savings at a later stage, whereupon they deviated from having MOVE as the only instruction and added the two additional instructions - JUMP and BRANCH. In addition, the RICA is fully dynamically reconfigurable, which is a key feature lacking in MOVE. Hence, the RICA is the architecture chosen as the paradigm upon which the reconfigurable architecture in this work is built.

REFERENCES

[1] G. Estrin, B. Bussel, R. Turn, and J. Bibb, "Parallel processing in a restructurable computer system," *IEEE Transactions on Electronic Computers*, vol. EC-12, no. 6, pp. 747-755, Dec. 1963.

[2] W. Carter, K Duong, R H Freeman, H Hsieh, J Y Ja, J E Mahoney, L T Ngo, er al., "A user programmable reconfigurable gate array," in *Proc. CICC*, pp. 233-235, May 1986.

[3] "FPGA - Field Programmable Gate Array," available: http://www.fpgacentral.com/pld-types/fpga-field-programmable-gate-array, accessed on 11/9/2014.

[4] The Industry's Breakthrough 7 Series FPGA Families, available at http://www.xilinx.com/products/silicon-devices/fpga/index.htm, accessed on 11/9/2014.

[5] R. D.Wittig and P. Chow, "One Chip: an FPGA processor with reconfigurable logic," *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 126-135, Apr. 1996.

[6] R. Razdan, M. D. Smith, "A high-performance microarchitecture with hardware-programmable functional units," *Proceedings of the 27th Annual International Symposium on Microarchitecture*, MICRO-27, pp. 172-180, 30 Nov-2 Dec. 1994.

[7] S. Hauck, T.W. Fry, M.M. Hosler, J. P. Kao, "The Chimaera reconfigurable functional unit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 206-217, Feb. 2004.

[8] Takashi Miyamori and Kunle Olukotun, "REMARC: Reconfigurable Multimedia Array Coprocessor," *IEICE Transactions on Information and Systems E82-D*, pp. 389-397, 1998.

[9] Carl Ebeling and DarrenC. Cronquist, and Paul Franklin, "RaPiD - Reconfigurable pipelined datapath," *Lecture Notes in Computer Science: Field-Programmable Logic Smart Applications, New Paradigms and Compilers*, vol. 1142, pp. 126-135, 1996.

[10] Alan Marshall, Tony Stansfield, Igor Kostarnov, Jean Vuillemin, Brad Hutchings, "A reconfigurable arithmetic array for multimedia applications," *Proceedings of ACM/SIGDA seventh international symposium on Field programmable gate arrays*, pp. 135-143, 1999.

[11] J. Becker, T. Pionteck, C. Habermann, M. Glesner, "Design and implementation of a coarse-grained dynamically reconfigurable hardware architecture," *IEEE Computer Society Workshop on VLSI*, pp. 41-46, May 2001.

[12] Francisco Barat, Murali Jayapala, Tom Vander Aa, et al. edited by Cheung, George Constantinides, "Low Power Coarse-Grained Reconfigurable Instruction Set Processor," *In Field Programmable Logic and Application*, vol. 2778, pp. 230-239, 2003.

[13] G. Sassatelli, G. Cambon, J. Galy, L. Torres, "A dynamically reconfigurable architecture for embedded systems," *12th International Workshop on Rapid System Prototyping*, pp. 32-37, 2001.

[14] E. Mirsky, A. DeHon, "MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources," *IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 157-166, 17-19, Apr. 1996.

[15] Lisa J K Durbeck and Nicholas J Macias, "The Cell Matrix: an architecture for nanocomputing," *Nanotechnology Journal*, vol. 12, no. 3, pp. 217-230, 2001.

[16] H. Zhang, V. Prabhu, V. George, M.Wan, M. Benes, A. Abnous, J.M. Rabaey, "A 1 V heterogeneous reconfigurable processor IC for baseband wireless applications," *IEEE International Solid-State Circuits Conference*, pp. 68-69, Feb. 2000.

[17] J. R. Hauser, J. Wawrzynek, "Garp: a MIPS processor with a reconfigurable coprocessor," *The 5th Annual IEEE Symposium Proceedings on Field-Programmable Custom Computing Machines*, pp. 12-21, 16-18 Apr. 1997.

[18] Reetinder P. S. Sidhu, Sameer Wadhwa, Alessandro Mei, Viktor K. Prasanna, "A Self-Reconfigurable Gate Array Architecture," *Proceedings of The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications*, Springer-Verlag, pp. 106-120, 2000.

[19] R. David, D. Chillet, S. Pillement, O. Sentieys, "DART: a dynamically reconfigurable architecture dealing with future mobile telecommunications constraints," *Proceedings Parallel and Distributed Processing International Symposium*, vol. 8, pp. 15-19, April 2001.

[20] D. C. Chen and J. M. Rabaey, "PADDI: Programmable Arithmatic Devices for Digital Signal Processing," *In VLSI Signal Processing IV*, pp. 240-249, IEEE Press, Nov. 1990.

[21] D.C. Chen and J.M. Rabaey, "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Real Time Data Paths," *IEEE Journal of Solid State Circuits*, vol. 27, no. 12, pp. 1895-1992.

[22] P. Hilfinger, "A high-level language and silicon compiler for digital signal processing," *in proceedings IEEE Custom Integrated Circuits Conferences*, pp. 240-243, May 1985.

[23] H. Singh, Lee Ming-Hau, Lu Guangming, F.J. Kurdahi, N. Bagherzadeh, E.M. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Transactions on Computers*, vol. 49, no. 5, pp. 465-481, May 2000.

[24] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R.R. Taylor, R. Laufer, "PipeRench: a coprocessor for streaming multimedia acceleration," *Proceedings of the 26th International Symposium on Computer Architecture*, pp. 28-39, 1999.

[25] R. Hartenstein, M. Herz, T. Hoffmann, U. Nageldinger, "On Reconfigurable Co-Processing Units," *Proceedings of Reconfigurable Architectures Workshop (RAW98), held in conjunction with 12th International Parallel Processing Symposium (IPPS-98) and 9th Symposium on Parallel and Distributed Processing (SPDP-98)*, Orlando, Florida, USA, March 30,1998.

[26] Reiner W. Hartenstein, Michael Herz, Thomas Hoffmann, Ulrich Nageldinger, "Using the KressArray for reconfigurable computing," *Proceeding of SPIE, Configurable Computing: Technology and Applications*, vol. 3526, Boston, USA, 1998.

[27] R. W. Hartenstein, R. Kress, "A datapath synthesis system for the reconfigurable datapath architecture," *Design Automation Conference, 1995. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95., IFIP International Conference on Hardware Description Languages. IFIP International Conference on Very Large Scal*, pp. 479-484, 29 Aug-1 Sep 1995.

[28] H. Corporaal, "Design of transport triggered architectures," *Proceedings of Fourth Great Lakes Symposium on Design Automation of High Performance VLSI Systems, GLSV '94*, pp. 130-135, 4-5 Mar 1994.

[29] J. Heikkinen, J. Sertamo, T. Rautiainen, J.Takala, "Design of transport triggered architecture processor for discrete cosine transform," *15th Annual IEEE International ASIC/SOC Conference*, pp. 87-91, 25-28 Sept. 2002.

[30] P. Hamalainen,; J. Heikkinen, M. Hannikainen, T.D. Hamalainen, "Design of transport triggered architecture processors for wireless encryption," *8th Euromicro Conference on Digital System Design Proceedings*, pp. 144-152, 30 Aug.-3 Sept. 2005.

[31] J. Heikkinen, J. Takala, A. Cilio, and H. Corporaal, "On Efficiency of Transport Triggered Architectures in DSP Applications," *in Advances in Systems Engineering, Signal Processing and Communications*, N. Mastorakis, Ed., pp. 25-29, WSES Press, New York, NY, USA, 2002.

[32] He Yifan, She Dongrui, B. Mesman, H. Corporaal, "MOVE-Pro: A low power and high code density TTA architecture," *International Conference on Embedded Computer Systems (SAMOS)*, pp. 294-301, 18-21 July 2011.

[33] S. Khawam, I. Nousias, M. Milward, Yi Ying, M. Muir, T. Arslan, "The Reconfigurable Instruction Cell Array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 75-85, Jan. 2008.

[34] A. Major, Yi Ying, I. Nousias, M. Milward, S. Khawam, T. Arslan, "H.264 Decoder Implementation on a Dynamically Reconfigurable Instruction Cell Based Architecture," *IEEE International SOC Conference*, pp. 49-52, 24-27 Sept. 2006.

[35] Z. Wang, A.T. Erdogan, T. Arslan, "A SDR Platform for Mobile Wi-Fi/3G UMTS System on a Dynamic Reconfigurable Architecture,"

2009 European Signal Processing Conference (EUSIPCO-2009), Glasgow, UK, August 24-28, 2009.

[36] I. Nousias, S. Khawam, M. Milward, M. Muir, T. Arslan, "A Multi-objective GA based Physical Placement Algorithm for Heterogeneous Dynamically Reconfigurable Arrays," *the 17th International Conference on Field Programmable Logic and Applications (FPL 2007)*, pp. 497-500, Amsterdam, Netherlands, 27-29 August 2007.

[37] Z. Wang, T. Arslan, A.T. Erdogan, "Implementation of Hardware Encryption Engine for Wireless Communication on a Reconfigurable Instruction Cell Architecture," *4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA 2008)*, pp. 148-152, 23-25 Jan. 2008.

[38] T. Hirao, Kim Dahoo, I. Hida, T. Asai, M. Motomura, "A restricted dynamically reconfigurable architecture for low power processors," *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pp. 1-7, Dec. 2013.

[39] O. Atak, A. Atalar, "BilRC: An Execution Triggered Coarse Grained Reconfigurable Architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no.7, pp. 1285-1298, July 2013.

[40] R.W. Brodersen, J.M. Rabaey, "Evolution of Microsystem Design," *Proceedings of the 15th European Solid-State Circuits Conference ESSCIRC '89*, pp. 208-217, 20-22 Sept. 1989.

[41] R. Hartenstein, "Coarse grain reconfigurable architectures," *In Proceedings of the 2001 Asia and South Pacific Design Automation Conference (ASP-DAC '01). ACM*, New York, USA, pp. 564-570, 2001.

Ahmed O. El-Rayis is a Visiting Research Fellow in the School of Engineering at the University of Edinburgh, UK. He is the founder of the Advanced Smart Antenna Technologies (ASAT) research group at the University of Edinburgh; in addition, he is co-Founder and Chief Operating Officer at Sofant Technologies Ltd a spin-out company from the University of Edinburgh.

He has PhD in Micro and Nano Systems from School of Engineering at The University of Edinburgh, UK; MEng in Computer Engineering from Lviv Polytechnic National University, Ukraine; Diploma in VLSI circuit design and Embedded Systems from Information Technology Institute, Egypt; BSc in Communication and Electrophysics EE from the University of Alexandria, Egypt. He worked in the space field for more than 5 years participating in satellites design and development especially in the payload, data control and acquisition in SSRE CONECS and YMZ in Ukraine. Before, he was working in Ellipsis Digital Systems for research and development department, also in Mentor Graphics as a development engineer in System design division for Signal Integrity issues. His research interests include wireless communication, Satellite development and space applications and Reconfigurable architectures.

Tughrul Arslan holds the Chair of Integrated Electronic Systems in the School of Engineering, University of Edinburgh, UK. He is a member of the Integrated Micro and Nano Systems (IMNS) Institute and leads the System Level Integration (SLI) Group in the University. His current research focuses on enhancing personal mobility and the design of high performance embedded wireless systems. He has supervised over 30 successful PhD theses and the author of more than over 300 articles and is the inventor of a number of patents in these areas.

He has led a number of successful projects in the design of low power embedded wireless systems. These have resulted in new patented technologies such as the Reconfigurable Instruction Cell Architecture (RICA), highly directional low power MEMS based antenna systems and embedded algorithms for exploitation of wireless signals for personal navigation indoors and areas of poor GPS signal visibility. Most of these technologies have been licensed to spin-out companies.

Professor Arslan was an Associate Editor for the IEEE Transactions on Circuits and systems I and IEEE Transactions on Circuits and Systems II. He is a member of the IEEE CAS Committee on VLSI Systems and Applications and is involved in the organization of numerous conferences (please see a selected recent few below). Recently he was the general chair for the IEEE NASA/ESA conference on Adaptive Hardware and Systems (AHS) and the ECSIS Bio-inspired, Learning, and Intelligent Systems for security Symposium (BLISS). He has been invited as a keynote speaker to a number of international conferences.

Khaled Benkrid is a Visiting Researcher in the School of Engineering at the University of Edinburgh; in addition, he is with ARM as World Wide University Program Manager. He is a member of the Integrated Micro and Nano Systems (IMNS) Institute and the System Level Integration group (SLIg). His research interests include: Custom hardware computing, Electronic Design Automation, Construction of optimised compilers, Image and video processing, High performance parallel computing with more than 50 Journal and Conference publications in these areas.