2012 15th Euromicro Conference on Digital System Design

The DeSyRe project: on-Demand System Reliability

I. Sourdis [1], C. Strydis[6], C.S. Bouganis[5], B. Falsafi[3], G.N. Gaydadjiev[1], A. Malek[1], R. Mariani[7], D. Pnevmatikatos[4], D.K. Pradhan[2], G. Rauwerda[8], K. Sunesen[8] and S. Tzilis[1]

[1]Chalmers University of Technology, Computer Science and Engineering Dept., Sweden
[2]University of Bristol, Computer Science Dept., UK
[3] École Polytechnique Fédérale de Lausanne, Computer and Communication Sciences, EPFL, Switzerland
[4] Foundation for Research and Technology – Hellas (FORTH), Institute of Computer Science (ICS), Greece
[5]Imperial College London, Electrical and Electronic Engineering Dept., UK
[6]Neurasmus B.V., The Netherlands
[7]Yogitech SpA, Italy
[8]Recore Systems B.V., The Netherlands

E-mail: sourdis@chalmers.se

The DeSyRe project builds on-demand adaptive and reliable Systems-on-Chips (SoCs). As fabrication technology scales down, chips are becoming less reliable, thereby incurring increased power and performance costs for fault tolerance. To make matters worse, power density is becoming a significant limiting factor in SoC design, in general. In the face of such changes in the technological landscape, current solutions for fault tolerance are expected to introduce excessive overheads in future systems. Moreover, attempting to design and manufacture a totally defect-/fault-free system, would impact heavily, even prohibitively, the design, manufacturing, and testing costs, as well as the system performance and power consumption. In this context, DeSyRe will deliver a new generation of systems that are reliable by design at well-balanced power, performance, and design costs.

***Keywords*— Fault-Tolerance, System-on-Chip, Reconfigurable Hardware, Medical Systems.**

I. Introduction

In the coming nanoscale era, chips are becoming less reliable, while manufacturing reliable chips is becoming increasingly more difficult and costly [1, 16]. Prominent causes for this are the shrinking device features, the sheer number of components on a given area of silicon, as well as the increasing complexity of current and future chips. It is expected that a significant number of devices will be defective already at manufacture time and many more will degrade and fail within their expected lifetime [2]. Furthermore, process variations as well as the increasing number of soft errors introduce additional sources of errors for future chips.

The ITRS targets a constant defect rate (1395 defects/m^2) in order to keep the chip yield constant [39]. Such a target is expected to substantially increase the chip manufacturing cost of future semiconductor technologies. Alternatively, chips need to be designed to tolerate an increasing number of defects in order to maintain a high yield. Apart from defects at manufacture time, aging effects are becoming more severe leading to more permanent and intermittent faults during the lifetime of a chip. Transistors degrade faster; while the degradation rate is further accelerated by the heavy testing processes (e.g. burn-in). Aging is expected to shorten SoC lifetime and to be a significant source of errors in technologies beyond 16-nm [1]. Process variations cause devices to operate differently than expected; such variations are random dopant fluctuations, heat flux, as well as lithography problems due to the shrinking geometries. Currently, on-chip clock frequency and total power consumption present variations up to 30% and 50%, respectively, across different parts of a single chip; it is projected that variations will only become more severe in the future and worst-case, deterministic design will be insufficient and unable to deliver reliable systems. Finally, as transistor count increases, the number of soft errors on a chip (i.e. transient faults) grows exponentially [2]. For example, by the 16-nm generation, the failure rate will be almost 100-fold higher that at 180-nm [2]; current fault-tolerance techniques such as simple check-pointing will, then, incur prohibitively high energy and performance costs.

As feature size continues to shrink and chips become less reliable, the cost for delivering reliable chips is expected to grow for future technology nodes. The price in system power and energy consumption, in performance degradation, and in extra resources, is getting higher in order to perform redundant computations in time or in space. However, it is a well-known fact that power consumption is becoming a severe problem, while performance no longer scales very well (mostly due to

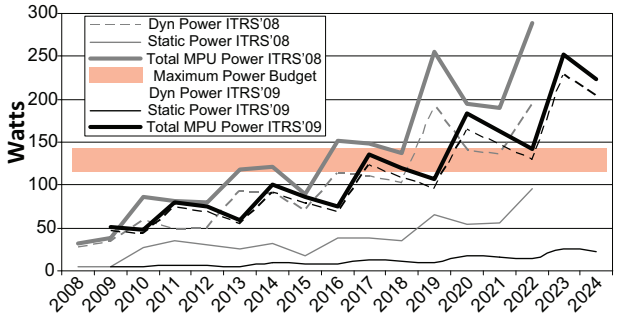

Fig. 1. Dynamic, Static and total power consumption of a Microprocessor chip vs. the maximum available power budget, based on the ITRS 2008 and 2009 projections.

The DeSyRe Project is supported by the European Commission Seventh Framework Programme, grant agreement n° 287611. www.desyre.eu

978-0-7695-4798-5/12 $26.00 © 2012 IEEE
DOI 10.1109/DSD.2012.127

power-density limitations). To reduce some of the above costs, *DeSyRe aims at reliable systems containing and tolerating unreliable components rather than targeting totally fault-free systems*. Our goal is to describe a new, more efficient design framework for SoCs which provides reliability at lower power and performance cost.

Although the above technology trends make the design of future SoCs harder, one of them can be turned to our advantage. As shown in Fig. 1, the increasing power density limits the gate density. In a few years, significant parts of a chip will be forced to remain powered-down in order to keep within the available power budget [49]. *In DeSyRe, we capitalize on this observation and propose to exploit the aforementioned unused resources to offer flexibility and reconfigurability on a chip.* Until now, reconfigurable hardware had a significant resource overhead; however, as explained above, this limitation no longer exists as on-chip resources are becoming "cheaper". A dynamically reconfigurable hardware-substrate can provide an excellent solution for defect tolerance; it can be used to adapt to faults on demand, isolate and correct defects, as well as to provide spare resources to substitute defective blocks. In the DeSyRe project, we intend to use such a reconfigurable substrate and combine it with system-level techniques to provide adaptive and on-demand reliable systems.

The remainder of the paper is organized as follows: in Section II, we give a brief overview of fault-tolerant systems and techniques, as well as of adaptive systems. In Section III, we describe the DeSyRe design framework. Sections IV and V summarize the DeSyRe fault-tolerance techniques and reconfigurable substrate. In Sections VI and VII, we present the medical applications and baseline SoCs used in DeSyRe. We conclude in Section VIII.

II. Background

In this section, we provide a very brief overview of current fault tolerance techniques and describe existing approaches for adaptive systems.

A. Fault-Tolerant Systems

A plethora of techniques have been proposed over the past 50 years to combat faults and provide reliable systems [3,4] starting even before the IBM mainframes [5] (S/360-50) in the 1960s until the recent Leon3FT [6] and Razor [7,8]. However, as indicated in the previous section the technological landscape changes, fault density increases, future chips will have an excess of resources which cannot all be active at the same time due to power issues. The above poses new challenges that will require radical changes in design methodologies when building reliable chips.

In general, techniques to increase reliability can be categorized to fault avoidance and fault tolerance techniques. *Fault avoidance* requires conservative design and is fault intolerant. For example, as illustrated in Fig. 2, implantable medical systems use significantly older and mature semiconductor technologies to manufacture reliable systems; a similar approach is also followed in other application domains such as the space domain. However, relying on older technology nodes has an excessive performance and energy

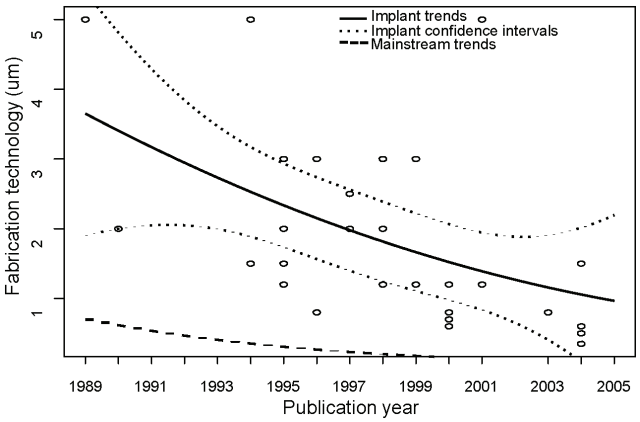

Fig. 2. Implantable Medical Systems currently use significantly older technology to guarantee reliability, resulting in excessive energy and performance overheads compared to using mainstream technology [9].

cost compared to using recent emerging technologies; such cost is often prohibitive for future embedded systems. In the other hand, *fault tolerance* requires redundancy (static or dynamic) to provide the information needed to negate the effects of a fault (detection, isolation, correction). However, redundancy, in time or space, introduces high power and performance overheads, too. DeSyRe attempts to reduce these overheads providing fault-tolerance on-demand based on fault types and densities, system constraints and application requirements.

1) Dealing with Permanent Faults

The common way to cope with permanent faults is to either provide redundant identical components [10] (coarse-grain reconfiguration) or fine-grain reconfigurable hardware (FPGAs). Some works consider redundancy at the core-level [11,12,13,14], or at the pipeline-stage (StageNet[15]). As an alternative to coarse-grain reconfigurable hardware, a large number of works exploit FPGAs to built reliable systems. For example, currently, NASA and CHREC use FPGAs to build reliable Systems for Space Computing [17,18]. In the past years, many other research works proposed FPGAs for fault-tolerance, such as [19,20]. There is a tradeoff between the level of tolerance to permanent faults and the performance, power, and area overheads of reconfiguration. On the one hand, coarse-grain reconfiguration has reduced performance, power and area costs, but is less tolerant to faults. On the other hand, fine-grain reconfiguration is more flexible and thus more tolerant to defects, but suffers higher performance, power and area overheads.

In the DeSyRe project, we propose a mix of both coarse- and fine-grain reconfigurable substrate to tolerate permanent faults. This is expected to provide increased defect tolerance at lower performance, power, and area costs.

2) Dealing with Transient Faults

In order to tolerate transient faults, redundancy is required to detect and correct them. Redundancy can be introduced at the core-level [21], thread-level or functional-unit level using checkers [4]. Currently, for microprocessor architectures, there has been proposed a multitude of techniques for dealing with faults in either the control [22], the datapath [23] or both [24]. Other approaches suggest ISA extensions for data-dependency tracking and control-flow checking, such as RSE

[25]. Furthermore, the Leon3FT is an example of a current processor resilient to transient faults using error-correcting codes (ECC) [6].

In general, static and always-on redundancy suffices to detect faults; however, it is not power- and energy-efficient since it often doubles the hardware and power consumption for the same performance. In the past, lowering the reliability standards has been proposed in order to save energy [26]. However, for many applications this is not acceptable, especially in the embedded domain. Another interesting approach is to provide application-aware reliability [27]; that is to enable redundancy only when needed.

DeSyRe will develop fault-tolerance mechanisms which are adapted *on demand* to the application requirements and the system constraints (energy, resources, performance needs) as well as to faults; this will be achieved by the synergy of the logical layers and physical partitions of the DeSyRe framework described in the next Section.

B. Customizable and Adaptive Systems

Recently, researchers have started working on design techniques for self-adaptive systems. W. Luk described his vision for self-optimizing and self-verifying systems [28] aiming at better efficiency, re-usability and correctness. J. Henkel et al. proposed adaptive embedded systems based on an atom/molecule model to adapt at runtime on system requirements not foreseen at design or at compile time [29]. In all cases, FPGA devices have been considered as the hardware substrate due to their flexibility of on-demand dynamic reconfiguration, while fault tolerance is often not the primary objective.

DeSyRe will build a novel reconfigurable substrate to support adaptation and fault tolerance at reduced power and performance cost. In combination with runtime-system software routines, the reconfigurable substrate will provide fault tolerance and system adaptation.

1) Runtime System Optimizations

Several systems support dynamic power management, often through voltage scaling, however when combined with fault tolerance, power management becomes significantly more complex. That is simply because fault tolerance consumes more power, while power efficiency, i.e. voltage scaling, can be more fault prone. Optimizing power and reliability on dynamically adaptive systems is still an open problem [30]; adding resource management and real-time performance constraints is even harder. Some works attempt to solve part of the problem. For example, several techniques attempt to optimize execution time under a given power envelope [31], or conversely minimize energy consumption under a performance constraint [32]. Other tools such as Olay [33] and Maestro [34] focus only on reliability management targeting homogeneous MPSoCs such as StageNet. Existing systems do not consider simultaneous optimization of application requirements (performance, QoS), and system constraints (power, temperature), in conjunction with reliability. In DeSyRe, runtime optimizations will consider faults, system constraints (energy, resources) and application requirements (real-time performance, expected reliability-level, task priority, etc.). This task becomes more challenging considering the heterogeneity of the DeSyRe systems and the fact that runtime support should be lightweight in order not to overburden the system.

2) Runtime Hardware Synthesis and Customization

In the past, a significant body of work has been performed on hardware synthesis, trading computation quality (i.e. precision, accuracy, efficiency of algorithms) for performance (processing latency and throughput), for power consumption and for area/resources. Some of the recent works in the field have been performed by Imperial, such as [35], which nonetheless require significant computational power and are performed off-line. As opposed to previous works, DeSyRe hardware synthesis will be *generic* and will be performed *on the fly*. Generation of new system configurations will be based not only on pre-computed, but also on dynamically created configurations (for reconfigurable interconnects).

3) Self-correcting, Self-checking, Self-aware Components

This is a relatively new topic where DeSyRe will contribute with new generic techniques. One of the works closest to DeSyRe is the *faultRobust* technology [36] developed by Yogitech SpA, one of the DeSyRe industrial partners. Yogitech adds wrappers to the components of a SoC (CPU, on-chip memory, local bus, etc.) in order to check their correct functionality. In DeSyRe, we will build on these wrappers making them more advanced.

III. The DeSyRe Design Framework

In this section, we briefly describe the different parts of the DeSyRe framework. Systems-on-Chip comprise multiple design levels, ranging from top-level running software to hardware components and all the way down to the elementary technological (transistor) substrate. Although increasing design complexity has so far been kept in check by partitioning the design in horizontal levels (e.g. software, hardware, technology), this approach is rapidly becoming inefficient due to the increasing degree of cross-level sophistication required for modern embedded-system design. Knowledge of levels both above and below a specific level is becoming imperative for building functional systems. This implies a vertical, cross-cutting, level partitioning with designers having knowledge of what transpires above and below their respective levels of expertise.

To address this pragmatic need, the DeSyRe framework is partitioned across two orthogonal design dimensions: a *physical* and a *logical abstraction*. The physical partitioning is based on the different technological substrates (with different fault densities) used in the various parts of the framework and is mostly of interest to experts closer to the technology level. The logical partitioning considers the same framework from the viewpoint of functionality (i.e. which part of the system does what) and should mostly interest experts closer to the architecture and system level.

1) Physical Partitioning

Fig. 3 illustrates the physical partitioning of the DeSyRe SoC. The design area is physically divided into a fault-free (FF) section providing overall system management and a fault-prone (FP) section providing the actual system functionality. The motivation for this partitioning is reducing the chip cost: Designing a totally fault-free system is expensive, thus the FF section should be small and lightweight.

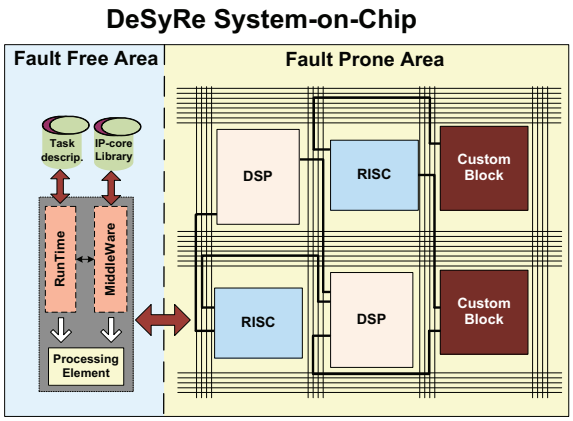

Fig. 3. DeSyRe SoC physical partitioning with a fault-free section for SoC management and a fault-prone section for SoC functionality.

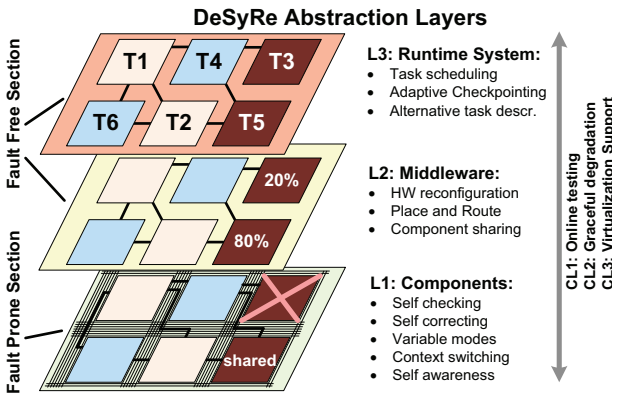

Fig. 4: Logical partitioning of a DeSyRe SoC: Components layer (SoC functionality), Middleware layer and Runtime-System layer (SoC management).

Fault-Free section: The FF section is required to provide centralized, system-wide control of the SoC, aiming to provide Quality of Service (QoS) attributes such as performance, low power consumption, resource utilization and fault tolerance. The various techniques through which this will be achieved involve an efficient combination of:

- Online fault tolerance.
- Runtime task scheduling, while being aware of task characteristics such as urgency and safety-criticality.
- Resource allocation, under varying availability of computational resources.
- Reconfiguration schemes to achieve flexible and defect-tolerant operation.

Fault-Prone section: The FP section is under the direct control of the FF section. It contains various components realized in a reconfigurable substrate (a mixture of fine- and coarse-grain reconfigurable logic). The components implement the main system functionality based on the target application (domain). They are required to exhibit, among others, self-checking and self-correcting properties, working in tight synergy with those of the FF section. To this end, the various components should be equipped with their own self-checking and -correcting mechanisms, albeit under (in)direct control of the FF section.

2) Logical Partitioning

The logical partitioning organizes the DeSyRe SoC in three main layers. Fig. 4 depicts the layers from bottom to top: components, middleware and runtime system. This subdivision is based on the abstraction level involved and the tasks handled by each layer.

Components: The bottom layer deals with fault-tolerance issues of each component (i.e. unit which delivers a specific functionality) in the FP section, individually. In other words, it is responsible for providing component-level (intrinsic) fault tolerance. The system is composed of multiple heterogeneous components located at the fault-prone section. The design of these components should take into account the requirements of the DeSyRe system. They will be able to autonomously detect faults that might appear in them and possibly correct a subset of these detected faults (in other words, to exhibit self-checking and self-correcting properties). To deal with the aforementioned faults, they should also have a certain degree of flexibility, by being able to switch context and support task migration –that is, receive a new task or transfer a running task elsewhere. Those features, of course, require interfacing with the above logical layers, to be able to send status information and receive commands concerning modifications in the mode of operation.

For the list of faults with which a component deals intrinsically, any detection and/or correction action from the component side has to be transparent to the upper layers. In case there is partial recovery - affecting functionality and QoS constraints - or no correction whatsoever, the upper layers have to be notified about the new status of the component. Component-level fault tolerance, as opposed to centralized techniques, should provide recovery schemes with the advantage of lower latency. On the other hand, these schemes will be, as a matter of fact, less efficient than the ones supported by upper logical layers.

Middleware: The second layer is responsible for the hardware synthesis and reconfiguration of the components in order to provide correctly functioning underlying hardware to the upper layer. In this layer we will develop mechanisms for the dynamic reconfiguration of hardware resources implemented in the (fine- and/or coarse-grain) reconfigurable FP section. The Middleware manages the components as black boxes which need to be interconnected, isolated, (re)placed in order to deliver a functioning hardware platform ready to be used for running the tasks that the runtime system dictates. To one extent, the Middleware makes all its actions transparent to the Runtime system in order to provide well-functioning hardware; that is in order to ensure the reconfiguration process is performed correctly and installs a correct new configuration.

Runtime System: The third layer deals with run-time issues of the system; its basic functionality is to schedule tasks to components, to ensure the best quality of service for the soft real-time portion of the applications, and to adapt the system in the presence of faults. To deal with faults, the Runtime System collects system health status information for all system components to establish a system health map. Using the health map together with a description of the application tasks and

TABLE I
DESYRE TECHNIQUES FOR TOLERATING VARIOUS TYPES OF FAULTS

Fault Type	DeSyRe Detection mechanisms	DeSyRe Correction mechanisms
Permanent	• Online testing • Self-checking components	• Coarse- or fine-grain Reconfiguration (via Middleware, and runtime optimizations) • Task migration (possibly using alternative task descriptions) • Component sharing • Graceful degradation (through the above mechanisms, and with the assistance of runtime system optimizations)
Transient	• Software error detection codes (used together with adaptive checkpoints) • Self-checking components	• Adaptive check-pointing (i.e. rollback and recover). o Application-aware o Adapted to fault density • Self-correcting mechanisms at the components (i.e. ECCs)
Intermittent	• Same as transient, with some additional software mechanism to distinguish them from the transient faults	• Task migration (possibly using alternative task descriptions) • System Reconfiguration and Runtime optimizations • Adaptive check-pointing (i.e. rollback and recover)

their requirements, the Runtime System identifies a global assignment of tasks to the various system resources that satisfies the application requirements and achieves the best possible performance from the (possibly faulty) processing cores in the fault prone area. The Runtime System functionality is realized in the FF section of the SoC (running on a GPP). The main challenge for the runtime system is to be adaptive in order to conform to the underlying reconfigurable hardware and to use it efficiently.

In addition to the above logical layers, the project is involved with three *distributed tasks* that span across all three layers, since all three of them need to deal with and support them. Those tasks are online testing, graceful degradation and virtualization support. The functionality of all three cross-layer tasks needs to be implemented in a distributed manner across the layers of the framework.

Online testing: Dynamically scheduled online tests of SoCs to detect errors and diagnose faults. Online testing is a task that is partially treated by: (i) the Runtime system, by rescheduling tasks to idle components for on-line testing, (ii) the Middleware, by providing a spare component while the original is tested, and (iii) the Components, by providing observability and controllability required for testing.

Graceful degradation: Depending on the faults, the functionality of the system should be adjusted and/or degraded. Instead of letting the entire system crash, it is desired to allow performance to drop or functionality to be reduced. The Runtime System controls the graceful degradation based on priorities and constraints of the application. Critical tasks are to be supported at all cost, while low priority tasks may be dropped; thereby, emergency modes of operation will be introduced, when the original functionality cannot be supported.

Virtualization: DeSyRe targets multiprocessor SoCs (MPSoCs) composed of heterogeneous and dynamically reconfigurable components. It is essential for the efficiency of a DeSyRe-based system to allow a task to be ported in heterogeneous components or various configurations of a component. Consequently, virtualization support is an important issue and it is addressed in the proposed work. To do so, components will be designed to support virtualization; they should either translate task descriptions to match the underlying hardware or alternatively maintain the same functionality despite reconfiguration. An additional issue requirement for the system is to support migration of a task from one component-type/configuration to another, in order to increase system availability and defect tolerance.

IV. DESYRE MECHANISMS FOR FAULT TOLERANCE

The proposed framework is expected to cope with permanent, transient and intermittent faults using the above described logical layers and physical parts. Table 1 summarizes the DeSyRe mechanisms to detect and correct various fault types.

Permanent faults (defects), due to design, manufacturing or aging effects, are detected and diagnosed in DeSyRe through online testing scheduled centrally by the runtime system or distributed by self-checking mechanisms at each component. Hardware reconfiguration of the defective part is the first choice for correction; this involves isolation, and replacement of the defective part, performed by Middleware with possible high-level decisions made at the runtime system layer. When reconfiguration is not possible the tasks scheduled in the defective parts of the system are re-scheduled in other available components. To economize system resources, component sharing may be possible at the cost of performance degradation. The above will be achieved based on the graceful degradation policies of the system.

Transient faults (soft-errors) are detected in DeSyRe either centrally at the software-level using error detection (and correction) codes or distributed at the components using checkers. Check-pointing mechanisms will be used to rollback to a known good state and recover from faults. Check-pointing will be adaptive to the application requirements and the fault density (i.e. adaptive frequency and placement of checkpoints) for energy efficiency. At the component level self-correcting mechanisms may be able to correct a sub-set of faults.

In DeSyRe, intermittent faults (i.e. periodic transient faults) are distinguished from transient faults in order to treat these differently. To do so, we add to the transient-fault detection mechanisms extra functionality to determine whether a fault is repeated. Then, task migration to other available components or hardware reconfigurations can be used for correction, while adaptive check-pointing and context switching mechanisms will be required to rollback and recover or to migrate a task.

V. DESYRE RECONFIGURABLE SUBSTRATE

The DeSyRe framework relies significantly on a flexible and dynamically reconfigurable hardware substrate to isolate, replace and (when possible) correct design and manufacturing

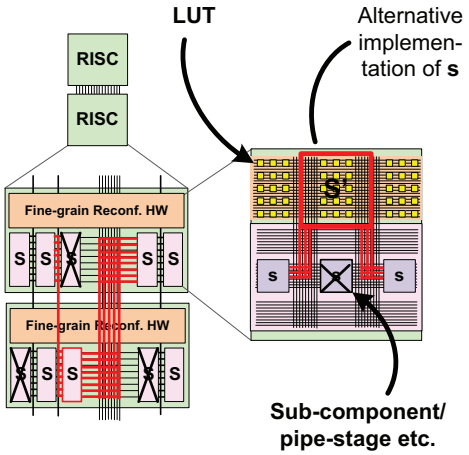

Fig. 5. The novel DeSyRe flexible/reconfigurable hardware substrate.

defects as well as defects due to aging. In previous works, the design choice was either coarse- or fine-grain granularity of substitutable units; these are units that can be replaced when defective. In the first case, the substitutable unit can be an entire component (e.g. a microprocessor), while in the latter case an FPGA logic cell. There are tradeoffs between these two alternatives. Coarse-grain approaches are less defect-tolerant - fewer defects can damage the system - but are more power-efficient and faster, since the correctly functioning substitutable unit is implemented in fixed hardware. Fine-grain approaches can tolerate more defects, but realizing a system in an FPGA introduces high performance overhead and high power cost.

One of the primary challenges in the DeSyRe project is to define a more efficient underlying hardware substrate for the DeSyRe SoC. DeSyRe explores a granularity mix of fine- and coarse-grain underlying hardware in order to provide increased defect-tolerance without giving away significant parts of the system performance and power efficiency. Fig. 5 depicts such an example with two DeSyRe RISC components. In this example, each RISC processor is divided in smaller sub-components (implemented in fixed hardware), i.e. pipeline stages, functional units, etc., surrounded by reconfigurable interconnects/wires. In the absence of defects, the sub-components "S" will form the RISC component. However, in case a sub-component is defective, it can be isolated using the reconfigurable interconnects, and subsequently be replaced either by an identical unused neighboring sub-component (S), or by a functionally equivalent instance (S') implemented in fine-grain reconfigurable hardware.

VI. Applications

The DeSyRe framework will be applied to two medical SoCs, namely an artificial cerebellum and an artificial pancreas. Both applications have been provided by Neurasmus BV, one of the DeSyRe-project industrial partners.

Application 1: Artificial Cerebellum

Medical SoC for artificially restoring part of the cerebellum to recover sensorimotor control

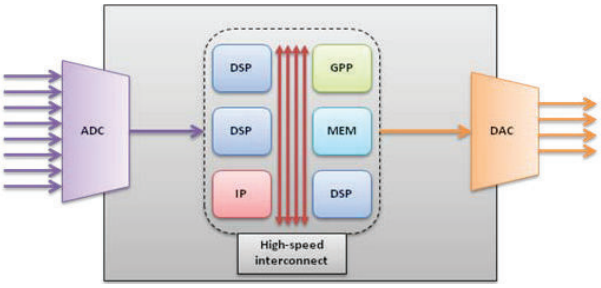

Fig. 6. Portable artificial-cerebellum system for restoration of damaged olivocerebellar modules in the brain.

Neurasmus BV focuses on developing an **artificial, real-time, cerebellar medical MPSoC for rescuing damaged parts of an actual, biological brain** suffering from various brain diseases stemming from loss of sensorimotor control, such as Calcium-Channelopathies, Fragile-X and Autism. Sensorimotor integration in healthy subjects relies on the proper *morphology* and *timing* of neural (spike-coded) signals exchanged between the cerebellar and the cerebral cortices.

An artificial cerebellum effectively is a *multi-level, closed-loop-control* system which will entail extensive and diverse *processing tasks* as well as *real-time interfacing* to multiple recorded (input) and stimulated (output) neural structures. In its first generation, it is expected to be portable and has the following requirements:

- high reliability due to its medical application;
- high throughput (for replacing a large number of biological neurons);
- low latency (for "real-brain-time" processing);
- power efficiency for portability; and
- adaptiveness to different input patterns.

By containing multiple parallel heterogeneous parts, the artificial cerebellum is a heterogeneous MPSoC and a natural fit for the DeSyRe framework which will provide the above features by design.

For the requirements of the DeSyRe project, we focus on the *olivocerebellar modules* of the brain. Neurasmus has already devised and maintains accurate neural models in-house, which can be mapped to an artificial cerebellum. If this is done correctly, the noise levels in the transmitted neural signals are expected to revert to normal levels. However, the current software implementation of the Neurasmus models is not able to achieve real-time execution. Ultimately but beyond the timeframe of this project, a real-time artificial cerebellum SoC, can lead to rescue approaches in various other pathogenic instances such as altered Purkinje-cell calcium influx, inhibited Purkinje-cell feed-forward pathways, etc..

The conceptual diagram of the envisioned DeSyRe system is illustrated in Fig. 6. The DSP, GPP and custom-IP ensemble will be used to emulate clusters of hundreds of olivocerebellar modules so as to match the massively parallel nature of the cerebellum. The system will further consist of A/D and D/A converters for interface to the biological tissue. Within DeSyRe, recorded real neural-signal traces will be used as inputs for the evaluation of the proposed system.

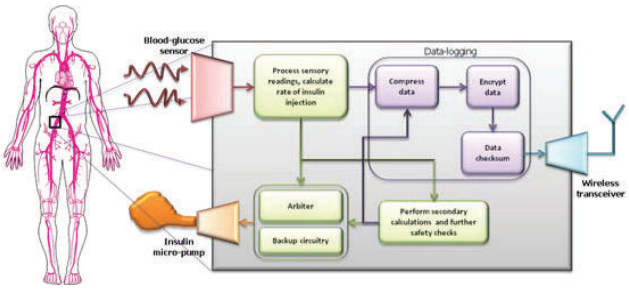

Fig. 7 Implantable artificial-pancreas system for treatment of diabetic patients.

Although significant research has already been done on software and hardware neural modeling [45, 46], no solution exists yet which allows scientists to simulate cerebellar structures of realistic sizes (on the order of magnitude of thousands of neurons) at real brain speeds. With this application we aspire to build not only a realistically sized and timed cerebellar subsystem but also a system that will continue working in the presence of large numbers of defects.

Application 2: Artificial Pancreas

Artificial pancreas for diabetic patients: Closed-loop control for automatic regulation of glucose-concentration in the blood.

In diabetic patients the pancreas cannot produce insulin. The sheer number and increasing rates of such patients worldwide is major incentive for developing a so-called "artificial-pancreas" device. In essence, such a device is a closed-loop-control implant which samples the glucose levels in the blood stream and releases insulin as needed (see Fig. 7). Even though an actual, chronic artificial pancreas has not been developed yet, glucose-sensing implants have constantly increased in numbers and improved over the years [37, 38].

Given the large and constantly increasing number of diabetic patients worldwide, Neurasmus BV envisions implementing the **artificial-pancreas implant as a DeSyRe-based, implantable system for automatically and accurately regulating glucose levels in the blood**. It shall contain a module for processing sensory data (measuring glucose concentration in the blood), a controler for insulin injection, and a third module implementing the calculations for the closed-loop control of the system. A user interface will also be included. In developing this application, we will adopt the latest achievements in artificial-pancreas research [47, 48], while improving the state of the art by providing highly defect-tolerant devices, suitable for chronic implantation.

Although both DeSyRe medical applications exhibit high reliability requirements, the artificial pancreas system will be radically different from the artificial cerebellum, a difference which will help to demonstrate the diversity and flexibility of the DeSyRe framework. The artificial-cerebellum system requires significant processing of a large number of modeled cerebellar nuclei. On the contrary, the artificial pancreas requires – by comparison – few computations for processing and controlling sensors and actuators, and for interfacing (security, communication, compression) to a treating physician or the patient. It requires, however, more rigorous closed-loop control, and should be ultra-low-power and adaptive to new treatment descriptions.

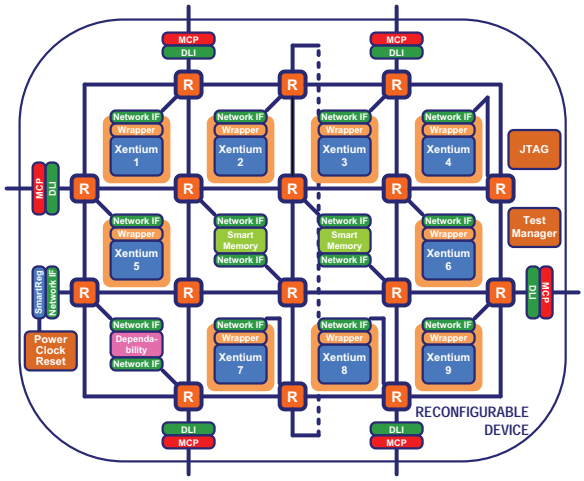

Fig. 8: CRISP Reconfigurable Fabric Device.

VII. Heterogeneous Baseline SoC

The two applications described in the previous section will be ported to two heterogeneous SoCs. These two SoCs will be designed based on the CRISP multicore SoC template, integrating techniques developed within the CRISP project [43]. The CRISP Reconfigurable Fabric Device is depicted in Fig. 8 showing a multicore SoC template with 9 reconfigurable Xentium DSP cores and 2 embedded memory tiles. The DeSyRe baseline SoCs will consist of Xentium DSP cores [40], on-chip memory blocks, RISC processors, and custom blocks interconnected with a fault-tolerant Network-on-Chip (NoC). Fault-tolerance extensions to existing NoC approaches [41, 42] as implemented in [43] and [44], respectively, will be considered. The fault-free DeSyRe section will be realized on a separate general-purpose processor. The artificial-cerebellum SoC will consist primarily of multiple DSP processors to perform the computationally intensive tasks of modeling brain-cells. The artificial pancreas will be more lightweight; it is expected to consist of a RISC processor to carry the necessary computations, a second one for the user-interface, and custom IP blocks.

VIII. Conclusions

The increasing need for fault tolerance imposed by the currently observed technology scaling introduces significant performance and power overheads. In our attempt to alleviate these overheads, the DeSyRe project will deliver a new generation of – by design – reliable systems, at a reduced power and performance cost. This is achieved through the following main contributions. Rather than aiming at totally fault-free chips, DeSyRe designs fault-tolerant systems built using unreliable components. In addition, DeSyRe systems are on-demand adaptive to various types and densities of faults, as well as to other system constraints and application requirements. A new dynamically reconfigurable substrate is designed and combined with runtime system software support in order to leverage on-demand adaptation, customization, and reliability at reduced cost. The above will result in a well-defined, generic and repeatable design framework for a large

variety of SoCs. The proposed framework is applied to two medical SoCs with high reliability constraints and diverse performance and power requirements.

REFERENCES

[1] S. Borkar “Designing reliable systems from unreliable components: The challenges of transistor variability and degradation”, IEEE Micro, 25(6):10–16, 2005.

[2] P. Hazucha, T. Karnik, J. Maiz, S. Walstra, B. Bloechel, J. Tschanz, G. Dermer, S. Hareland, P. Armstrong, and S. Borkar, “Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-µm to 90-nm generation,” Electron Devices Meeting, 2003. IEEE International Technical Digest (IEDM), Dec. 2003

[3] D.P. Siewiorek, Architecture of Fault-Tolerant Computers: A Historical Perspective. IEEE Proceedings 19:1710-1734, Dec 1991.

[4] F. F. Sellers, M.-Y. Hsiao, and L. W. Bearnson, Error Detecting Logic for Digital Computers, McGraw-Hill Inc., 1968.

[5] M.-W. Bartlett and M.-L. Spainhower, "Commercial Fault Tolerance: A Tale of Two Systems," IEEE Trans. Dependable Secur. Comput., vol. 1, iss. 1, pp. 87-96, 2004.

[6] J. Gaisler. LEON3FT-RTAX Data Sheet and User's Manual. http://www.gaisler.com/doc/leon3ft-rtax-ag.pdf

[7] D. Ernst, et.al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," 36th IEEE/ACM Int’l Symp. on Microarchitecture (MICRO), 2003, p. 7.

[8] D. Blaauw, S. Kalaiselvan, K. Lai, Wei-Hsiang Ma, S. Pant, C. Tokunaga, S. Das, D. Bull. Razor II: Situ Error Detection and Correction for PVT and SER Tolerance, 2008 IEEE Int’l Solid-State Circuits Conf.

[9] C. Strydis,“Universal Processor Architecture for Biomedical Implants – The SiMS Project” PhD Dissertation, TU Delft, March 2011.

[10] X. Zhang, H.G. Kerkhoff, "Design of a Highly Dependable Beamforming Chip", 12th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN (DSD), IEEE Computer Society, August 2009

[11] N. Aggarwal, P. Ranganathan, N.P. Jouppi, and J.E. Smith. Configurable isolation: building high availability systems with commodity multi-core processors. SIGARCH Comput. Archit. News 35, 2, pp. 470-481, 2007.

[12] J.C. Smolens, B.T. Gold, B. Falsafi, and J.C. Hoe. Reunion: Complexity-Effective Multicore Redundancy, Int’l Symp. on Microarchitecture, pp. 223 - 234, 2006.

[13] B.T. Gold, B. Falsafi, and J.C. Hoe. Chip-Level Redundancy in Distributed Shared-Memory Multiprocessors, Proceedings of the 15th IEEE Pacific Rim Int’l Symp. on Dependable Comp., pp. 195-201, 2009

[14] D. Sylvester, D. Blaauw, and E. Karl. ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon. IEEE Design and Test, 23, pp. 484-490, 2006.

[15] S. Gupta, S. Feng, A. Ansari, J. Blome, and S. Mahlke, "StageNetSlice: a reconfigurable microarchitecture building block for resilient CMP systems," Int’l Conf. on Compilers, architectures and synthesis for embedded systems (CASES), New York, NY, USA, 2008, pp. 1-10.

[16] T. Austin, V. Bertacco, S. Mahlke, and Y. Cao, "Reliable Systems on Unreliable Fabrics," IEEE Des. Test, vol. 25, iss. 4, pp. 322-332, 2008.

[17] NASA Tech briefs, FPGAs with Reconfigurable Fault-Tolerant Redundancy. www.nasa.gov

[18] J. Ramos, et. Al.. High-Performance, Dependable Multiprocessor, IEEE Aerospace Conference, 2006.

[19] S. Tzilis, I. Sourdis, G.N. Gaydadjiev. "Fine-grain Fault Diagnosis for FPGA Logic Blocks", Int. Conf. on Field-Programmable Technology (FPT 2010), Beijing, China, December 2010.

[20] J. M. Emmert, C. E. Stroud, and M. Abramovici, "Online fault tolerance for FPGA logic blocks," IEEE Trans. Very Large Scale Integr. Syst., vol. 15, iss. 2, pp. 216-226, 2007.

[21] T. M. Austin, "DIVA: a reliable substrate for deep submicron microarchitecture design," in Proc. MICRO 32: Proceedings of the 32nd annual ACM/IEEE int’l Symp. on Microarchitecture, 1999, pp. 196-207.

[22] K. Seongwoo, A.K. Somani. On-Line Integrity Monitoring of Microprocessor Control Logic, Int’l Conf on Computer Design: VLSI in Computers & Processors (ICCD), 2001, pp. 314-319.

[23] A. Meixner and D. J. Sorin, "Error Detection Using Dynamic Dataflow Verification," PACT '07: 16th Int’l Conf. on Parallel Architecture and Compilation Techniques, Washington, DC, USA, 2007, pp. 104-118.

[24] A. Meixner, M. E. Bauer, and D. Sorin, "Argus: Low-Cost, Comprehensive Error Detection in Simple Cores," in Proc. MICRO '07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, Washington, DC, USA, 2007, pp. 210-222.

[25] N. Nakka, Z. Kalbarczyk, R. K. Iyer, and J. Xu, "An architectural framework for providing reliability and security support," Dependable Systems and Networks, International Conference on, pp. 585-594, 2004.

[26] G. Chen, M. Kandemir, and F. Li, "Energy-aware computation duplication for improving reliability in embedded chip multiprocessors," in Proc. ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation, Piscataway, USA, 2006, pp. 134-139.

[27] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra. ERSA: error resilient system architecture for probabilistic applications. Conference on Design, Automation and Test in Europe (DATE). 1560-1565. 2010

[28] Wayne Luk, “SELF-OPTIMIZING AND SELF-VERIFYING DESIGN: A VISION”, Book chapter in: The Future of Computing, essay in memory of Stamatis Vassiliadis, pp. 69-79, September 2007.

[29] Lars Bauer, Muhammad Shafique, Dirk Teufel, Jörg Henkel: A Self-Adaptive Extensible Embedded Processor. SASO 2007: 344-350

[30] I. Sander, A Jantsch. Modelling Adaptive Systems in ForSyDe, DASMOD Workshop on Verification of Adaptive Systems Kaiserslautern, Germany, September 14th, 2007

[31] Barry Rountree, David K. Lowenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, Tyler K. Bletsch: Adagio: making DVS practical for complex HPC applications. ICS 2009: 460-469

[32] Dong Li, Bronis R. de Supinski, Martin Schulz, Kirk Cameron and Dimitrios S. Nikolopoulos. Hybrid MPI/OpenMP Power-Aware Computing. IPDPS 2010.

[33] S. Feng, S. Gupta, and S. Mahlke, Olay: Combat the Signs of Aging with Introspective Reliability Management. Workshop on Quality-Aware Design (W-QUAD) Jun. 2008.

[34] S. Feng, S. Gupta, A. Ansari and S. Mahlke, Maestro: Orchestrating Lifetime Reliability in Chip Multiprocessors, Proc. Intl. Conf. on High-Perf. Embedded Arch. and Compilers (HiPEAC) 2010, pp 186-200.

[35] Qiang Liu, George A. Constantinides, Konstantinos Masselos, Peter Y. K. Cheung: Combining Data Reuse With Data-Level Parallelization for FPGA-Targeted Hardware Compilation: A Geometric Programming Framework. IEEE Trans. on CAD of Integrated Circuits and Systems 28(3): 305-315 (2009)

[36] R. Mariani, M. Baumeister and P. Fuhrmann, “A single channel, fail-safe microcontroller to simplify SIL3 safety architectures in automotive applications”, Electronic Systems for Vehicles VDI Conference, Baden-Baden, Germany, October 2007

[37] M.C. Shults, R.K. Rhodes, S.J. Updike, B.J. Gilligan and W.N. Reining, A telemetry-instrumentation system for monitoring multiple subcutaneously implanted glucose sensors, IEEE Transactions on Biomedical Engineering, 1994, pp. 937-942.

[38] P. Atanasov et al., Implantation of a refillable glucose monitoring-telemetry device, Biosensors & Bioelectronics 12(7), pp. 669~680, 1997

[39] International Technology Roadmap for Semiconductors: http://www.itrs.net/

[40] Xentium® DSP Core http://www.recoresystems.com/technology/

[41] P. T. Wolkotte, Exploration within the network-on-chip paradigm, PhD thesis, University of Twente, Enschede, The Netherlands, January 2009.

[42] E. Bolotin et al., QNoC: QoS architecture and design process for network on chip, Journal of Systems Architecture, vol. 50, 2004.

[43] T. Ahonen et al., CRISP: Cutting Edge Reconfigurable ICs for Stream Processing, in Reconfigurable Computing - From FPGAs to Hardware/Software Codesign. Springer, 2011.

[44] K.H.G. Walters,. et al., Multicore SoC for on-board payload signal processing, NASA/ESA Conf. on Adaptive Hardware and Systems (AHS), 2011, pp. 17-21.

[45] R.K. Weinstein et al., Methodology and Design Flow for Assisted Neural-Model Implementations in FPGAs, IEEE Trans. On Neural Systems And Rehabilitation Engineering, Vol. 15, No. 1, March 2007.

[46] E. L. Graas et al., An FPGA-based Approach to High-Speed Simulation of Conductance-Based Neuron Models, Neuroinformatics, Vol. 2(4), p. 417-36, 2004.

[47] M.E. Wilinska et al., Simulation Environment to Evaluate Closed-Loop Insulin Delivery Systems in Type 1 Diabetes, Journal of Diabetes Science and Technology, Volume 4, Issue 1, January 2010.

[48] C.W. Chia et al., Glucose sensors: toward closed loop insulin delivery, Endocrinol Metab Clin N America, Vol. 33 (2004), pp 175–195

[49] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki: Toward Dark Silicon in Servers. IEEE Micro 31(4): 6-15, 2011.