14th FPGAworld CONFERENCE

SEPTEMBER 19th-21st, 2017
STOCKHOLM + COPENHAGEN

EDITORS

Lennart Lindh, Ketil Røed, Vincent J. Mooney III, Johnny Öberg, Johan Alme, Santiago de Pablo and Mohamed Shalan.

ACADEMIC PROCEEDINGS 2017

The FPGAworld Conference addresses all aspects of digital and hardware/software system engineering on FPGA technology. It is a discussion and network forum for researchers and engineers working on industrial and research projects, state-of-the-art investigations, development and applications. The proceedings contain the academic presentations, a separate report contains also the industrial presentations; for more information see www.fpgaworld.com.

SPONSORS

ISBN 978-1-4503-5154-6

Proceedings of FPGAworld 2017
Index

FPGAworld @ Copenhagen

Academic General Chairman's Message

Welcome to the 14th edition of the FPGAworld conference! With the dramatic shrinkage of chip technologies continuing to advance, capabilities of FPGAs are only increasing their expansion into areas traditionally only covered by ASICs. We cordially welcome you to this premier event mixing industrial and academic perspectives in an open forum of debate and inclusion.

As in previous years, this year the conference is again taking place in two locations, Copenhagen (Denmark) and Stockholm (Sweden). Earlier editions of FPGAworld have been organized in Finland (Tampere), India (Udaipur), Germany (Munich) and Sweden (Lund).

The FPGAworld conference has academic reviewed papers, industrial reviewed papers, keynote addresses and product presentations, resulting in a fertile mix of presentations. FPGAworld also has in both locations exhibitors as well as an occasional tutorials. This aims to cover an increase in demand from the academic and industrial participants for FPGA knowledge.

Please check out the website (http://www.fpgaworld.com) for more information about FPGAworld 2017. In addition, you may contact Mia (mia@fpgaworld.com) for more information about product presentations, web advertisements, sponsoring, tutorials and exhibits. For academic and industrial presentations, please see the FPGAworld website for more information.

FPGAworld already now opens to receive suggestions for next year's conference in September 2018. We are interested in suggested keynote speakers, web advertisements (year around), sponsoring, technical papers, product presentations, student projects, exhibits and tutorials. Submissions are open to students, academics and industrial professionals. Together we can help make the FPGAworld conference exceed our best expectations!

The organizers of FPGAworld would like to thank all contributors; we hope that attendees will truly enjoy and benefit from the FPGAworld conference.

Sincerely,

Ketil Røed, University of Oslo, Norway

2017 Academic Organization

General Academic Chair
Ketil Røed, University of Oslo, Norway

Academic Program Chair
Johan Alme, University of Bergen, Norway

Academic Publication Chair
Santiago de Pablo, University of Valladolid, Spain

Academic Publicity Chair
Mohamed Shalan, American University of Cairo, Egypt

Steering Committee Members
Vincent J. Mooney III, Georgia Institute of Technology, USA
Peeter Ellervee, Tallinn University of Technology, Estonia
Johnny Öberg, KTH Royal Institute of Technology, Sweden
Lennart Lindh, Jönköping University, Sweden

Academic Programme Committee Members
Peeter Ellervee, Tallin University of Technology, Estonia
Reiner Hartenstein, TU Kaiserslautern, Germany
Leandro Soares Indrusiak, University of York, United Kingdom
Pramote Kuacharoen, National Institute of Development Administration, Thailand
Johnny Öberg, KTH Royal Institute of Technology, Sweden
Santiago de Pablo, University of Valladolid, Spain
Adam Postula, University of Queensland, Australia
Ketil Røed, University of Oslo, Norway
Timo D. Hämäläinen, Tampere University of Technology, Finland
Johan Alme, University of Bergen, Norway
Erno Salminen, Tampere University of Technology, Finland
Mohamed Shalan, American University of Cairo, Egypt
Shashi Kumar, Jönköping University, Sweden
Anshul Kumar, Indian Institute of Technology, IIT Delhi, India
Paul Kolin, Indian Institute of Technology, IIT Delhi, India
Vincent J. Mooney III, Georgia Institute of Technology, USA
Attiq Ur Rehman, University of Bergen, Norway

2017 General and Industrial Organization

Industrial Program Chair
Lennart Lindh, FPGAworld, Sweden

Industrial Programme Committee Members
Solfrid Hasund, Bergen University College
Kim Petersén, HDC, Sweden
Mickael Unnebäck, ORSoC, Sweden
Fredrik Lång, EBV, Sweden
Niclas Jansson, BitSim, Sweden
Göran Bilski, Xilinx, Sweden
Per Henricsson, Elektroniktidningen, Sweden
Espen Tallaksen, Bitvis, Norway
Tommy Klevin, ÅF, Sweden
Tryggve Mathiesen, InformASIC, Sweden
Fredrik Kjellberg, Net Insight, Sweden
Daniel Stackenäs, Altera, Sweden
Stefan Sjöholm, Realfast, Sweden
Torbjorn Soderlund, Xilinx, Sweden
Anders Enggaard, Axcon, Denmark
Doug Amos, Synopsys, UK
Guido Schreiner, The Mathworks, Germany
Stig Kalmo, Engineering College of Aarhus, Denmark
Hichem Belhadj, Microsemi, USA
Rolf Sylvester-Hvid, Aktuell Elektronik, Denmark
Tony Eriksson, Future Electronics, Sweden
Ann-Luise Vestrup Kristensen, Silica, Denmark
Mircea Alexandru Dabacan, Digilent Ro, Romania
Andreas Engberg, ConMed, USA
Abbas Bigdeli, Nicta, Australia
Siegfried Weigert, ibw, Germany
Nikolay Rognlien, Arrow Norway AS, Noway
Basavaraj Hooli, FPGAworld, India
Clint Cole, Digilent, USA
Gagan Puri, Coreel, India
Sudarshan Natu, Symphony, India
Udayprakash Raghunath Singh, SPSU, India
Yehoshua Shoshan, Innofour, Sweden
Hai Migdal, Gidel, Israel
Thorsten Trenz, Trenz Electronic GmbH, Germany
Gerd Prillwitz, Ansys, Germany
Juergen Kessler, BlackForest EDA, Germany
Willem Groter, HDL Works, Netherlands
Maurizio Casti, Thales Group, Italy
Andreas Schwarztrauber, MSC, Germany
Ben Liu, Digilent, Taiwan
Mattias Karlsson, Saab, Sweden
Antti Innamaa, Synopsys, Finland
Jacky Cheng, Huafan Tech, China
Mikko Rasa, Arrow, Finland
Thony Johansson, ÅF, Sweden
Henrik Eeckenhaut, Sigasi, Belgium
Mike Dini, Dini Group, USA
Jan Viktorin, RehiveTech, Czech Republic

Svend Modtgard, Wdiag, Germany
Soren Manicus, TekPartner, Denmark
Rune Domsten, IndesmaTech, Denmark

Industrial Program and Publicity Manager
Lennart Lindh, FPGAworld, Sweden

Sales, Registration and Finance Manager
Mia Lindh, FPGAworld, Sweden

Sponsors, Exhibitors and Product Presenters

ÅF, Sweden
DTU, Technical University of Denmark
Aktuel Elektronik, Denmark
Elektroniktidningen, Sweden
Prevas, Sweden
XILINX, USA
Linear Technology, USA
Dini Group, USA
Innofour, Netherlands
Terasic, Taiwan
Avnet Silica, Denmark
Avnet Silica, Sweden
Synective Labs, Sweden
Samtec, USA
Arrow, Europe
Motion Control, Sweden
AGSTU education, Sweden

In cooperation with ACM

FPGA™ world

FPGAworld 2017 @ Stockholm

Frösundaleden 2A
169 70 Solna, Sweden

Conference Programme

08:30 Registration

09:00 Conference opening – Room Renen
Lennart Lindh, FPGAworld.

09:15 Key Note Session
RF Data Converters in an All Programmable MPSoC FPGA
Brendan Farley, XILINX Inc., Ireland

10:00 Coffee Break & Exhibition

10:30 Parallel Sessions

12:00 Lunch Break & Exhibition

13:00 Mike Dini Talk
FPGA events during the year that has gone and gossips

13:30 Break

13:45 Parallel Sessions

14:45 Coffee Break & Exhibition

15:15 Key Note Session
Programmable Technologies: New Challenges and New Opportunities
Hichem Belhadj, Chief Systems Architect, CTO Office, Microsemi Corp. USA

16:00 Go Home Drink in the Exhibition Hall

The exhibition will be open during the day.
Coffee will be served in the exhibition area.

FPGA™ world

Room: Renen
Key Note Session @ Stokholm
Speaker: Brendan Farley
XILINX Inc., Ireland

§ **RF Data Converters in an All Programmable MPSoC FPGA**
Brendan Farley, XILINX Inc.
Room: Renen

Recent state-of-the-art FPGAs have seen the integration of multi-giga-sample RF data converters to address the requirements of next generation wideband digital communications system. The keynote presentation will give an overview of the RFSoC FPGA which integrates such functionality and will discuss some potential applications and future trends.

Brendan Farley is a Senior Director of Engineering at US multinational technology corporation Xilinx Inc. where he is responsible for Analog and Digital-RF Research and Development. Brendan holds a Bachelor Degree in Electronic Engineering from Trinity College Dublin and a Master of Science Degree in Technology Management from NUI Galway.

Room: Renen
Key Note Session @ Stokholm
Speaker: Hichem Belhadj
Microsemi Corp., USA

§ **Programmable Technologies: New Challenges and New Opportunities**
Hichem Belhadj, CTO Office, Microsemi Corp., USA.
Room: Renen

Hichem Belhadj has been with Microsemi for close to 20 years. He is currently the Chief System Architect at the CTO Office. Prior to joining the CTO Office, Hichem held executive management positions in Corporate Sales and Field Systems and Applications at Microsemi, Actel, IST, and INPG. Hichem holds a Master and PhD from the Polytechnic Institute of Grenoble, France.

FPGA™ world

Room: Renen
Sessions A1-A2-A3
Session Chair: Kim Petersén
HDC, Sweden

§ The Impact of Place and Route on FPGA Logic Synthesis

For a quarter century, synthesizing an RTL design into an FPGA circuit has required only a loose understanding of the impact of Place and Route (P&R) software. By estimating route delays during Logic Synthesis based on graph properties and with accurate timing constraints, it was possible to achieve timing closure even for high-frequency clocks. In this presentation, we explain useful techniques to improve system performance and to achieve success in P&R more reliably.

Presenter: Pieter J. Hazewindus, *USA*.

§ Portable Stimulus Specification
The Next Big Wave in Functional Verification

In this paper we will describe the upcoming proposed standard for "Portable Stimulus Specification" (PSS) from Accellera. We will show how a single model of stimulus and scenarios can be re-used across different environments such as High-level C models, UVM simulations or even embedded SW, thus providing the verification engineers with a unified way to model interaction with complex SoC's or FPGA's containing CPU cores and embedded SW.

More information: Accellera has been working on the new proposed PSS standard since 2014. At DAC 2017 the Working group released the first "Early Adopter" version of the standard. This new proposed standard has received tremendous amounts of interest from the industry - at DAC and DVCon the seminars about PSS were completely overbooked and only standing room was available.

For more information please see http://accellera.org/news/press-releases/244-accellera-portablestimulus-early-adopter-specification-now-available-for-public-review

Presenter: Staffan Berg, *Sweden*.

§ Constrained Random and Functional Coverage for VHDL testbenches controlled in a structured manner

OSVVM provides a good library for CR and FC. But how should we apply this in a TB to avoid the normal verification traps?

- Bad overview
- Bad readability
- Bad maintainability & extensibility
- Inefficient reuse

Even most well-structured TBs do not sufficiently avoid these problems.

This presentation will show how easy it is to combine OSVVM and UVVM to get a ‘Unified VHDL Verification Methodology’ that provides advanced CR and FC, - and at the same time promotes overview, readability, maintainability, extensibility and reuse.

Presenter: Espen Tallaksen, *Norway*.

Room: Räven
Sessions B1-B2-A4
Session Chair: Ketil Røed
University of Oslo, Norway

§ Use of Analog Signatures for Hardware Trojan Detection

Malicious Hardware Trojans can corrupt data which if undetected may cause serious harm. We propose a technique where characteristics of the data itself are used to detect Hardware Trojan (HT) attacks. In particular, we use a two-chip approach where we generate a data "signature" in analog and test for the signature in a partially reconfigurable digital microchip where the HT may attack.

This paper presents an overall signature-based HT detection architecture and case study for cardiovascular signals used in medical device technology. Our results show that with minimal performance and area overhead, the proposed architecture is able to detect HT attacks on primary data inputs as well as on multiple modules of the design.

Authors: Taimour Wehbe, Vincent J. Mooney, David Keezer, Omer T. Inan, Abdul Qadir Javaid, and Chinmoy Kulkarni, *Georgia Institute of Technology, USA*

§ Synthesis of VLIW Accelerators from Formal Descriptions in a Real-Time Multi-Core Environment

Designing, programming and design space exploration of predictable Real-Time systems on Heterogeneous Multi-Core platforms is a very complex task. The increasing validation costs and time-to-market pressure creates a desire to build systems that are correct by construction.

Formal description based on Model of Computations (MoCs) is a convenient way to create high-level models of such systems. The MoCs provide abstraction and high level modeling through a clear set of rules based on mathematics, which can be used as input for system synthesis. A formal synthesis flow would then ensure that the resulting real-time system is both predictable and correct by construction, provided that all transformations used in the flow can be verified/trusted.

In this paper we show how a Real-Time computation node in an MPSoC system, described using the Synchronous MoC, can be transformed into a VLIW accelerator. The created accelerator is incorporated as a computation node in a heterogeneous multi-core system implemented on an FPGA.

Author: Johnny Öberg, *Royal Institute of Technology (KTH), Sweden*.

§ Highly optimized streaming FFTs for FPGAs

In this work we show how streaming FFTs can be highly optimized on FPGAs. Compared to previous state-of-the-art, we increase the throughput per slice by about a factor of five for both Virtex-4 and Virtex-6 FPGAs without increasing the number of DSP blocks nor the amount of memory used. The results are obtained by better utilizing the FPGA resources rather than any novelty in the FFT algorithm nor the FFT architecture. Different optimization levels are presented with the fastest ones operating at the maximum clock frequency of the device.

Authors: Carl Ingemarsson, Petter Källström, and Oscar Gustafsson, *Linköping University, Sweden*.

Use of Analog Signatures for Hardware Trojan Detection

Taimour Wehbe[1], Vincent J. Mooney[1,2], David Keezer[1], Omer T. Inan[1,3] and Abdul Qadir Javaid[1]
[1]School of Electrical and Computer Engineering, [2]School of Computer Science, [3]Department of Biomedical Engineering
Georgia Institute of Technology, Atlanta, Georgia, USA
taimour.wehbe@gatech.edu, {mooney, dkeezer, omer.inan}@ece.gatech.edu, aqjavaid@gatech.edu

ABSTRACT
Malicious Hardware Trojans can corrupt data which if undetected may cause serious harm. We propose a technique where characteristics of the data itself are used to detect Hardware Trojan (HT) attacks. In particular, we use a two-chip approach where we generate a data "signature" in analog and test for the signature in a partially reconfigurable digital microchip where the HT may attack. This paper presents an overall signature-based HT detection architecture and case study for cardiovascular signals used in medical device technology. Our results show that with minimal performance and area overhead, the proposed architecture is able to detect HT attacks on primary data inputs as well as on multiple modules of the design.

CCS Concepts
• Security and privacy~Embedded systems security • Security and privacy~Hardware security implementation • Security and privacy~Malicious design modifications *• Security and privacy~Security in hardware*

Keywords
Hardware Trojans, Analog Signatures, Reconfigurable Logic, Ballistocardiography

1. INTRODUCTION
The chip manufacturing process is becoming more and more dis-aggregated leading to an increase in chip fabrication vulnerabilities to malicious activities and alterations referred to in the literature as Hardware Trojans (HTs). Effects of HT modifications can be disastrous if the attack targets sensitive applications. Therefore, hardware security, and more specifically, HT detection mechanisms are gaining increasing popularity in recent years.

Embedded devices are typically constrained in terms of energy consumption and computing power, making the process of designing methods to catch HTs not a trivial task. Therefore, an HT detection circuitry should try to provide the highest possible security while maintaining low area and power overhead.

In this paper, we present a technique to capture ultra-small HTs which attempt to modify the functionality of digital chips. These types of attacks are not easily detected by other mechanisms due to their extremely small size [5]. Our work is motivated by a health monitoring application which captures heart signals and transmits them for further processing and analysis [8]. The physiological signals have a known relationship which we take advantage of to create signatures that check for the integrity of the captured data. Specifically, during data harvesting, we create analog-based signatures, and then we check for the validity of these signatures on a digital chip using reconfigurable logic to ensure that the chip has no HT attacks and that the data's integrity is maintained before transmission.

The paper is divided as follows. Section 2 presents background and prior work about concepts that are used throughout the paper. Section 3 introduces our threat scenario. Section 4 discusses our HT detection architecture in detail, and Section 5 presents possible HT attacks and explains how our proposed architecture catches them. Section 6 reports our simulation and synthesis results. Finally, a discussion is presented in Section 7 before we draw our conclusions in Section 8.

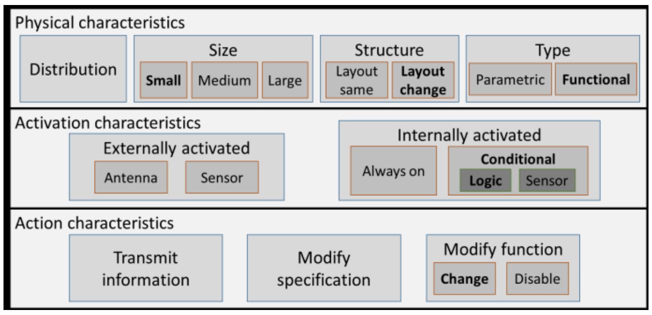


Figure 1. Hardware Trojan taxonomy.

2. BACKGROUND AND PRIOR WORK
2.1 Hardware Trojan Attacks and Detection Methods
Hardware attacks and specifically those related to Hardware Trojans have been receiving increased attention in the past several years. This is driven in a major way by industrial concerns about the integrity of their chips, especially with the recent paradigm of the Internet-of-Things (IoT). HT attacks on highly interconnected embedded devices can cause severe damage.

A formal HT taxonomy has been introduced which divides HT attacks according to three broad characteristics: (i) physical characteristics, (ii) activation characteristics and (iii) action characteristics (Figure 1) [2]. The (i) physical characteristics divide up HT attacks according to their distribution on the chip, their size, their structure and their type (parametric or functional). The (ii) activation characteristics divide them into internally activated HTs and externally activated ones. Externally triggered Hardware Trojans wait for an activation signal coming from outside the chip. Internally triggered ones can be classified into two subtypes: (a) an always on HT and (b) a conditionally triggered HT with the latter having the condition dependent on some logic in the circuit or some sensor attached to the circuit. The (iii) action characteristics divide the HT attacks according to their effect, i.e., whether the HT is going to leak information, corrupt functionality and/or modify the circuit's specification.

Several HT detection methods [1-7] have been proposed to address specific types of the aforementioned attacks. These methods can

]

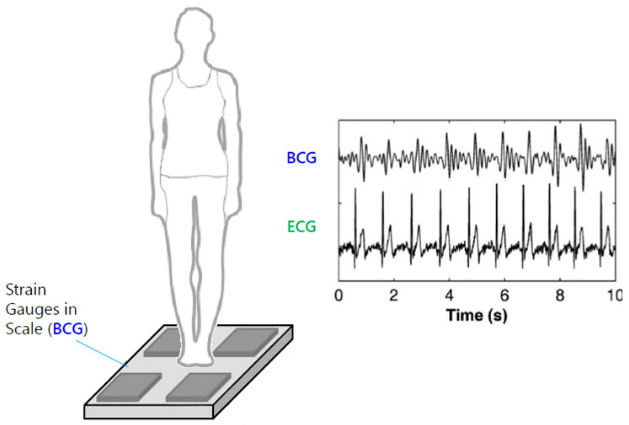


Figure 2. BCG force-plate.

be broadly divided into side-channel analysis techniques, HT triggering techniques, and correct functional verification [2]. For example, the authors of [2,6,7] provide a recent survey of multiple types of HT detection methods including methods that are based on power analysis, timing analysis, HT activation mechanisms and architecture-level detection. In addition, the authors of [1] introduce an HT prevention and detection mechanism for integrated circuits (IC) where they prevent a wide variety of HT attacks during IC testing and during operation in the field.

In our previous work [3-5] we also addressed multiple types of HT attacks by implementing an architectural-level detection mechanism. Specifically, we designed signature-based HT detection architectures to detect attacks that attempt to modify the functionality of a digital chip by modifying the structural logic of internal modules in the design.

2.2 Ballistocardiography

An alternative to the Electrocardiogram (ECG) measure of heart activity is the Ballistocardiogram (BCG) which utilizes Newton's second law ($F = ma$), i.e., the reaction of the body to the pumping action of the heart [8,9]. Specifically, given a properly instrumented scale, three-dimensional forces can be non-invasively captured to represent the cardiogenic vibrations of the body [8,9,10]. The BCG reveals information about heart rate, etc., but, unlike the ECG, the BCG but does not require electrodes or gel to obtain a high fidelity measurement of the body. Figure 2 shows a force plate measuring the BCG.

In this work, we use BCG sensors whose analog output values fall within a range of -0.9999 to 0.9999 and have required accuracy of four significant digits after the decimal. Thus, to cover the range and provide the needed accuracy, we use signed 16-bit fixed-point numbers. In some cases, values will be squared and added, in which case the range of 0 to 1.9999 needs to be supported. Thus, in this paper we use a fixed-point format with the most significant bit as the sign bit, the next bit as a representation of a value of 1 or 0, and the remaining 14 bits representing the fractional part of the number.

3. THREAT SCENARIO

Figure 3 shows the main HT threat model that we consider in this work. The HT is composed of trigger circuitry and a payload. This model is representative of most prior work [1-7].

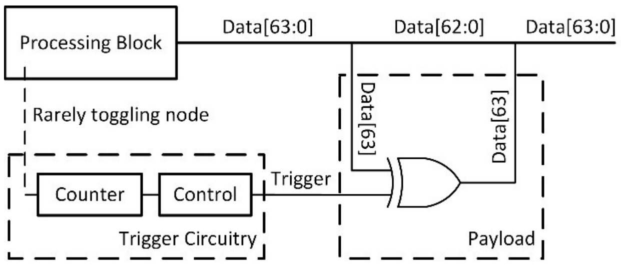


Figure 3. Hardware Trojan threat model.

The trigger circuitry is responsible for waiting for an activation characteristic to trigger the HT. The activation characteristic, as described in Section 2.1 and as shown in Figure 1, can be externally triggered or internally triggered. In our threat scenario, we consider HT trigger circuitry which is based on a conditionally triggered Trojan. As Figure 3 shows, a trigger circuitry could be composed of a counter and minimal control logic. The counter is attached to a rarely toggling node in the design such as a one coming from a processing block. The counter is incremented every time the node toggles. The control logic monitors the output of the counter and asserts the trigger once the counter reaches a specific predefined value. In addition, we assume in our threat model that once the HT is triggered, it remains on indefinitely. That is because if an attacker wants to intelligently turn on the HT for finite periods to bypass specific detection techniques, the trigger circuitry (shown in Figure 3) will have to be more complex resulting in an HT with a larger size. Such types of larger sized HTs are beyond the scope of our work and, as indicated in Section 2.1, can be caught by other HT detection techniques [2,6,7].

The payload in our threat model is composed of an exclusive-or gate that is connected to one of the data's bits in a way such that when the HT is triggered, the data bit is complemented resulting in an attack on the data stream. For example, Figure 3 shows an HT altering the most significant bit of Data[63:0].

In conclusion, we consider, as shown in bold face at each level in Figure 1, small size HT logic which is conditionally triggered on certain internal logic conditions or states able to be externally activated by an attacker where the HT attack aims to alter the functionality of the microchip [2]. The specific scenario involves analog measurements of health sensors where the data have known physiological relationships among the signals.

4. HARDWARE TROJAN DETECTION ARCHITECTURE

To increase protection against HT attacks which aim to disrupt chip functionality, we approach the overall embedded system design by separating the analog measurements from the digital processing by using a minimum of two chips as shown in Figure 4 [4]. The first chip shown on the left of Figure 4 – "Chip 1: Analog chip" – contains all of the analog components required for data acquisition. For example, this might include most of the sensor hardware, some filters, amplifiers, and analog-to-digital converters. The second chip shown on the right-hand side of Figure 4 – "Chip 2: Digital chip with embedded reconfigurable logic" – contains most of the processing modules in digital logic. For example, the digital chip can perform encryption/decryption on the data before it is sent to the cloud. In this work, we take advantage of the *security by design* approach where security is treated as a first class citizen in the design process [11]. Thus, we incorporate critical security features as early as we can in the design process of the data path. Specifically, our architecture enhances security by the generation

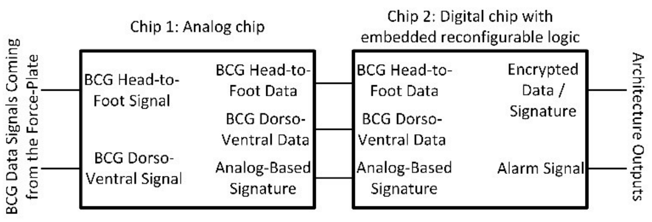


Figure 4. Two-chip architecture overview.

of data signatures in analog. This not only enables checking the integrity of the data as early as possible but also further complicates the job of an attacker. An attacker trying to simultaneously modify the data and its associated signature will now face much more difficulty as the attacking team should now incorporate expertise in both analog and digital designs.

4.1 Chip 1: Analog Chip

4.1.1 Strategy

A separate microchip focused on analog sensor components enables the generation of an analog signature. This approach provides a separate chip from the main digital computational circuitry, thus complicating the efforts of an attacker.

4.1.2 Example

Figure 5 shows an example of an architecture of the analog chip of Figure 4. Ballistocardiography (BCG) data harvested from sensors are amplified and fed to bandpass filters for initial processing and to improve the signal-to-noise ratio of the captured data. Normally, data would be directly fed to analog-to-digital converters and passed to digital logic for further processing on the same microchip. In our approach, as the data is being harvested, we simultaneously create an analog signature using the vector sum (i.e., square root of the sum of the squares of the two BCG measurements) of the captured data as shown in the bottom right-hand side of Figure 5. One of the major reasons behind choosing the vector sum as an analog signature is due to its importance and need in analyzing BCG data in later stages [12,13]. In addition, creating such a type of signature in analog is relatively simple due to its use of common analog components (adders, squarers and square root modules) [14]. The created analog signature is also fed to an analog-to-digital converter, where the analog signature is sampled, quantized, and fed to the digital logic of the second chip in Figure 4.

4.2 Chip 2: Digital Chip with Embedded Reconfigurable Logic

4.2.1 Strategy

In our architecture, the goals of the digital chip are (i) encryption/ decryption, (ii) signature comparison testing and (iii) application-specific computation. In order to test for the integrity of the data before transmission, a digital version of the analog-based signature is compared to what should be the same signature value recalculated from the raw data. Additional confidence in the encryption logic can be gained by decrypting the data (we assume that in typical operation, the embedded system is either sending or receiving data; thus, in the case of encrypting in order to send data, the decryption unit is not needed for sending and so is available to decrypt the encrypted data for test purposes). We perform the signature testing in embedded reconfigurable logic making it harder for an HT to attack our testing mechanism as the attacker would not know the mechanism's exact location prior to deployment. In addition, reconfigurability allows our architecture to be more flexible and scalable where the same digital chip can work with multiple versions of the analog chip performing different types of signature generation and testing. In summary, our testing mechanism compares a digital version of the analog-based signature arriving in a separate input from the analog chip to a regenerated version of the signature value using raw data after encryption and decryption. If the signatures do not match (within a tolerance level due to possible low order differences in least significant bits due to small variations which may occur in analog-to-digital conversion), we activate an alarm signal.

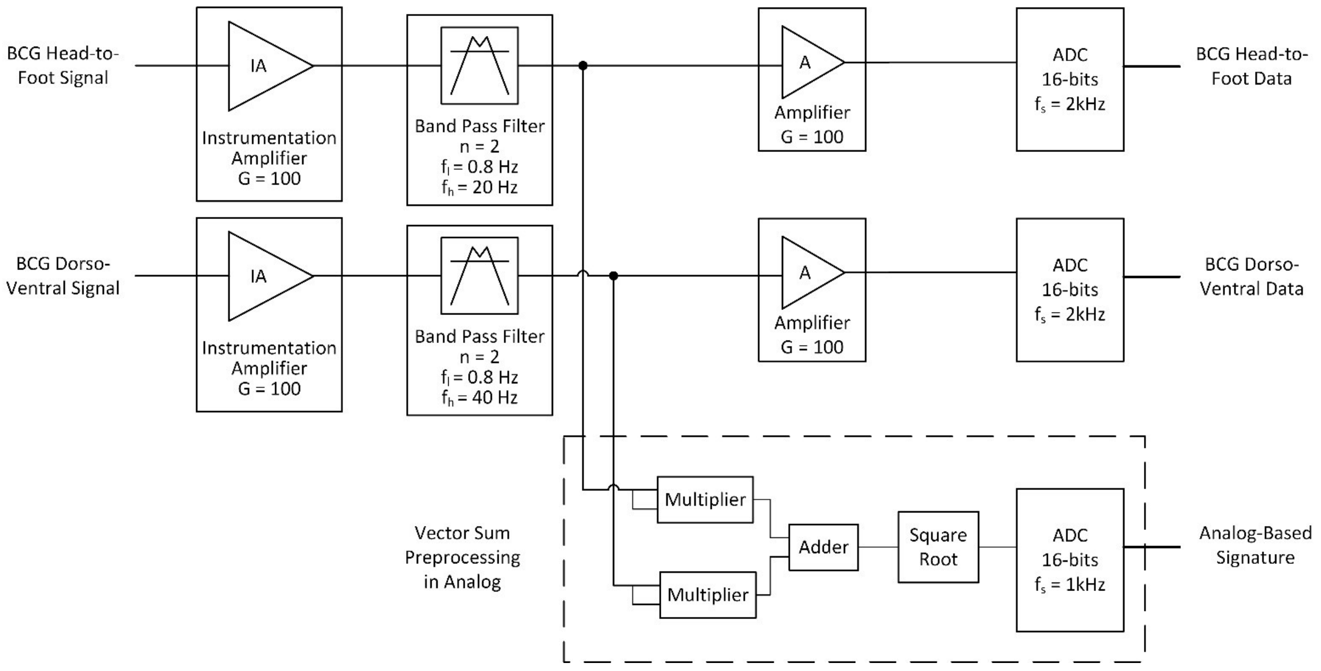


Figure 5. Analog chip sampling data and generating vector sum as a signature.

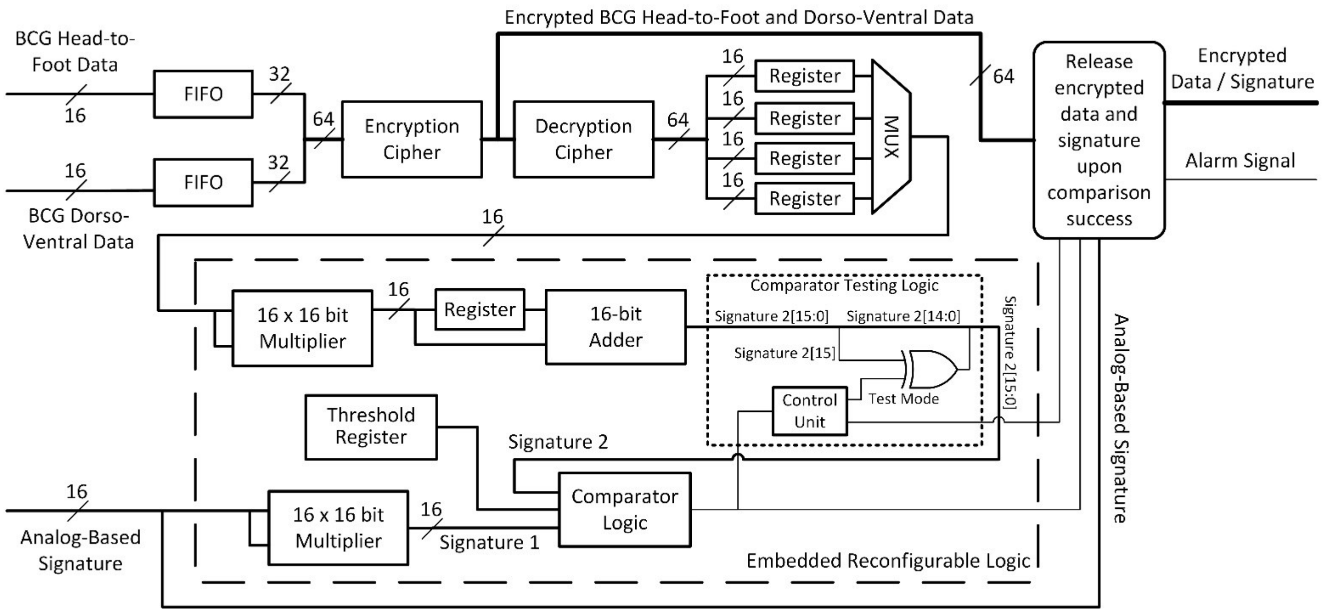


Figure 6. Digital chip validating the integrity of the data by checking for signature correctness in reconfigurable logic.

4.2.2 Example

Figure 6 displays an example of an architecture for the digital chip in our design approach. As mentioned in Section 4, this chip contains most of the processing modules and encryption blocks. As shown in Figure 6, the digital chip receives three data inputs from the analog chip. The first two inputs represent BCG data components, namely, (i) the BCG Head-to-Foot Data and (ii) the BCG Dorso-Ventral Data [12,13]. The third input to the chip is the analog-based signature which in our scenario is the vector sum (i.e., square root of the sum of the squares) of each sample of the BCG data (i) and (ii).

Figure 6 compares the digital version of the analog-based signature with the BCG raw data using reconfigurable logic as follows. First, the 16-bit vector sum (i.e., the digital value of the analog-based signature) is input to a 16 x 16 bit multiplier, operating as a squarer, thus generating a 32-bit output. This value represents the sum of squares of the two BCG data inputs. It is to be noted that since the analog chip is sampling the BCG data using 16-bit analog-to-digital converters and since our data is represented using fixed point number values between -1 and +1, the 32-bit result of the multiplier is truncated and the most significant 16-bits of the result are used as inputs to the next stage with the fixed-point format described in Section 2.2. The created 16-bit version of the sum of squares is referred to as Signature 1 in Figure 6.

Simultaneously, the input set of BCG data is buffered and concatenated to form blocks of 64-bits for transmission. Each 64-bit block of data is then passed through an encryption cipher such as PRESENT [15]. The encrypted data (ciphertext) is then passed through a decryption cipher to regenerate the plaintext. The reason for decrypting the ciphertext is to help detect HT attacks on the encryption and decryption modules [5]. The regenerated plaintext is partitioned and each data set is passed through a series of multiplications and additions to recalculate the sum of squares of the input data. This generated sum of squares of each data set is referred to as Signature 2 in the block diagram. Signature 1 and Signature 2 are then fed through a comparator logic. The comparator logic, shown in Figure 6, performs the following:

- *if (|Signature 1 – Signature 2| ≤ threshold)*
  declare a match
- *if (|Signature 1 – Signature 2| > threshold)*
  declare a mismatch

where the *threshold* is an input set by the user due to the analog nature of the application and the signature. The result of the comparator is then passed to a release logic block (top right-hand side of Figure 6). The release logic is responsible for saving the values of the encrypted data and signature of each BCG data set until the comparator decision is made. By monitoring the comparator's output, the release logic takes the decision as to whether the encrypted BCG data set and the corresponding signature is to be released or not. Ideally, if the architecture is not attacked, the two internally generated signatures (Signature 1 and Signature 2) should match. However, if an HT attack attempts to corrupt any of the significant bits of the data or signature values, the comparison logic will declare a mismatch and the data will not be released.

To verify the correct operation of the comparator logic, a "comparator testing logic" block, shown in the small dotted box in Figure 6, is inserted into our architecture. Specifically, the control unit of the comparator testing logic periodically initiates a test of an altered signature to verify the correct operation of the comparator block in case of a failed comparison. To do so, the comparator testing logic triggers a "test mode" signal to intentionally modify Signature 2, as shown in Figure 6. It then monitors the comparator's result. If the comparator declares a match, the comparator testing logic catches the discrepancy and sends an alarm to the release logic block to stop the transmission of data. A more detailed description of the functionality of the comparator testing logic is presented later in Section 5.2.1.

It is important to note that in this work our architecture focuses only on creating means of HT detection by asserting an alarm signal when an HT is believed to be present. The decisions and countermeasures to such types of attacks are kept to be processed and analyzed by higher level policies and protocols.

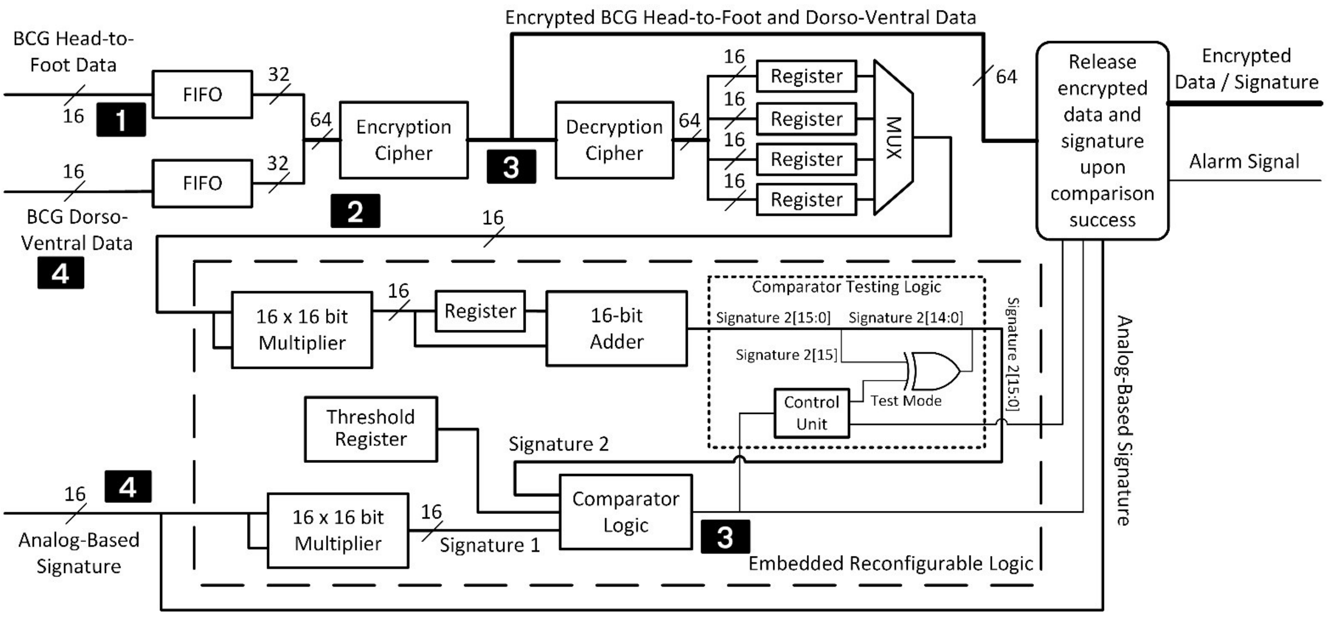


Figure 7. The digital chip architecture with the different types of HT attacks discussed in Section 5.

5. HARDWARE TROJAN ATTACK AND DETECTION

Figure 7 shows the digital chip architecture with the specific HT attacks that we discuss and simulate in this work. The attacks we aim to address are numbered according to their type from 1 to 4. Attack types 1 and 2 are single attacks, i.e., targetting one point in the architecture, and attack types 3 and 4 are coordinated attacks, i.e., targetting multiple points in the architecture.

5.1 Single Attacks

Attack types 1 and 2 target a single point in the architecture as shown in Figure 7. The goal behind these two types of attack is to modify the data either as it arrives at the input of the chip or at the output of any of the internal modules along the data path. Both types of attack attempt to modify the data in the same way as presented in our threat scenario described in Section 3 and shown in Figure 3. Namely, HT trigger circuitry is connected to an exclusive-or gate such that when the Trojan is triggered, one bit of the targeted data is complemented resulting in data modification.

5.1.1 Attack Type 1

Attack type number 1 (Figure 7) is an attack on the input data immediately after it reaches the chip and before any processing or signature generation has happened [4]. This type of attack is important as its detection depends on the analog signature generation. Other types of signature-based HT detection techniques do not cover this type of attack as their Signature 1 generation relies on the input data [1,5]. In our detection approach however, only Signature 2 in Figure 7 will be affected. The comparison then with Signature 1 will result in a mismatch and the release logic will prevent the data and signature transmission out of the chip.

5.1.2 Attack Type 2

Attack type number 2 targets the intermediate data as it passes through the different modules in our architecture. Figure 7 shows an example of this type of attack where the HT tries to modify the output of the multiplexer, right before the data is fed to the squarer module. This results in the generation of an altered Signature 2, which when compared to Signature 1 results in a comparison mismatch and alarm trigger. Contrary to HT attacks of type 1, HT attacks of type 2, i.e., inserted to affect the output of the different modules in an architecture, have been studied earlier in literature. Different detection techniques including signature based ones were proven to be effective [1,3-5]. Our architecture not only detects these types of attacks but also detects attacks that target the input data as shown in attack type 1.

5.2 Coordinated Attacks

Hardware Trojan attacks of types 3 and 4 attempt to initiate a coordinated attack targeting simultaneously two points of the architecture as shown in Figure 7.

5.2.1 Attack Type 3

A detailed view of an HT of attack type 3 is presented in Figure 8. The HT trigger circuitry is connected to two payloads. The first payload (Payload 1 in Figure 8) affects the architecture in the same way as discussed earlier in attack types 1 and 2 as well as in Figure 3. The second payload (Payload 2 in Figure 8), which is triggered at the same time, attempts to set the output of the comparator logic to a fixed value indicating a matched comparison.

In this example of attack type 3 as shown in Figures 7 and 8, once the trigger is set, this attack simultaneously targets (i) the output of the encryption cipher by modifying it and (ii) the output of the comparator logic by forcing the result of the comparison to show a match even if the signatures at the input of the comparator logic do not match. Since this tiny HT remains always on once triggered, the comparator logic will always show a match regardless of input. The effect of both payloads result in the modification of the data and its passage undetected. To prevent such a type of attack, the comparator testing logic block is inserted to frequently test for this specific case. Periodically, the test mode signal shown in Figure 7 is asserted so that Signature 2 coming out of the 16-bit adder is intentionally modified. The result of the comparison is then read by the comparator testing logic. If the result of the comparator shows a match, the comparator testing logic will detect the attack and inform the release logic. When the test mode is off, the

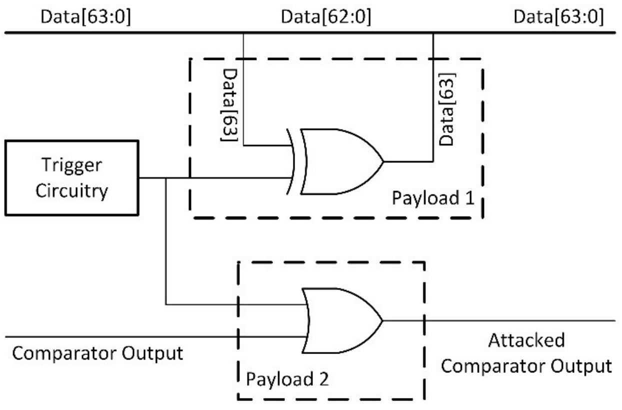


Figure 8. Hardware Trojan comparator attack.

signature is passed as is (i.e., unaltered) so that the comparator testing logic does not affect the functionality of the circuit.

5.2.2 Attack Type 4

Figure 9 shows a detailed view of an HT of attack type 4. The HT trigger circuitry in this case is also connected to two payloads. The first payload (Payload 1 in Figure 9) attacks the BCG input data, and the second payload (Payload 2 in Figure 9) attacks the analog-based signature (see Figure 7). A coordinated attack on the least significant bits of the BCG input data and the analog-based signature (as shown in Figure 9) will result in the modification of the data and, if the modified values result in $|Signature\ 1 - Signature\ 2| \leq threshold$, may result in HT operation going undetected. It is unclear what ability the attacker would gain by changing the low order bits as these slight variations in the values of the inputs and the signature may also occur due to the lack of precision of the analog sensor.

Another variant of attack type 4 is when the HT attempts to modify one of the high order bits of both the BCG input data and the analog-based signature. In this case, the attacker would have to exploit the vector sum relation between the BCG inputs to successfully modify both the BCG data and the signature in a way such that the modifications pass undetected. However, this type of HT would require additional more complex circuitry (such as multipliers and adders) and would therefore fall beyond our threat model of a small sized HT of only a few logic gates as discussed in Section 3. Such types of attacks – including associated techniques for detecting HTs with large footprints (e.g., via power-based techniques in addition to others) – have been well studied in literature [2]. It is important to note that these prior HT detection techniques are complementary to our work and can be incorporated alongside our approach.

6. EXPERIMENTAL RESULTS

The following section reports and discusses the functional simulations and the synthesis results of our analog-based signature HT detection architecture. The digital portions of our architecture were implemented in VHDL code, and our simulations were done using Mentor Graphics ModelSim SE version 10.2b revision 2013.05 for Linux.

In the following simulations, we assume that the digital signatures of our design (Signature 1 and Signature 2 in Figure 6) require an accuracy of at least 3 significant digits after the decimal. Therefore, we set the comparator *threshold* 16-bit register to a hexadecimal value of "0008" which in binary is "0000000000001000" and equals 2^{-11} in our fixed point representation (see Section 2.2).

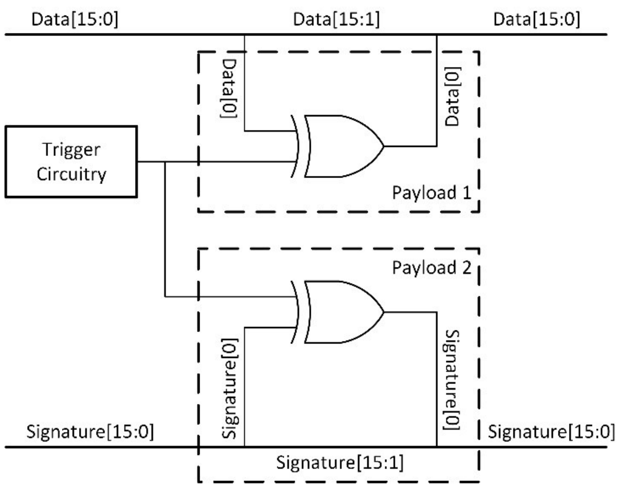


Figure 9. Hardware Trojan attack on data and signature.

6.1 Simulation Results and Functional Verification

Each of the attack types shown in Figure 7 and described in Section 5 were simulated to verify the functional correctness of our design and to demonstrate when the architecture is able to catch the HT attacks.

6.1.1 Simulation of Attack Type 1

To simulate attack type 1, which is at the input of the digital chip, we inserted an HT logic similar to the one shown in Figure 3 targeting the input data as it arrives on chip. The input BCG Head-to-Foot data in Figure 7 was modified such that one of the most significant bits in the 16-bit input was complemented. As mentioned in Section 3, the HT threat scenario that we consider in this work is triggered by some internal conditions or states. For our simulation purposes, the HT trigger waits on an attacker-defined number of occurrences of a specific input data. When the required condition is met, the trigger is set and the payload modifies the input data resulting in the modification of the functional behavior of the chip.

After the altered input data is passed through the encryption and decryption ciphers, the resulting output is then fed through a series of multiplications and additions and finally is compared with the squared value of the analog-based signature. Since the input data was altered by the HT, the values of signatures at the input of the comparator differ by an amount greater than the threshold (2^{-11}) and so the comparator declared a mismatch at its output. The release logic, monitoring the comparator's output, stopped the transmission of the altered encrypted data and triggered an alarm signal indicating the presence of the HT. All of this was verified in VHDL simulation using ModelSim.

6.1.2 Simulation of Attack Type 2

The simulation of attack type 2 was implemented in a similar fashion as attack type 1. The major difference between this type of attack and attack type 1 is the place where the HT attacks. In attack type 2, the HT, once triggered, modifies the value at an output of a hardware block in the design. In our simulations, we performed multiple separate tests by inserting HT logic at the output of the different modules of the architecture.

For example, in one of our simulations, we inserted HT logic at the output of the multiplexer in the design as shown by the black box containing “2” (for attack type “2”) in Figure 7. This resulted in modifying a reasonably significant bit of the BCG data right before signature regeneration, leading eventually to an incorrect Signature 2. Once the regenerated signature (Signature 2) was compared to the analog-based signature (Signature 1), the comparator found that the signature difference exceeded the threshold and thus declared a mismatch so that the release logic prevented the transmission of the data and asserted the alarm signal.

Also, another simulation of the same attack type, this time at the output of the encryption cipher, confirmed the need to decrypt the data and recreate the signature from the regenerated plaintext rather than directly from the input.

6.1.3 Simulation of Attack Type 3

To simulate attack type 3, the HT logic had to wait for the same trigger as in the previous attacks. However, the HT now has two payloads (Figure 8) affecting two different points in the architecture. Figure 7 shows the points at which we set the HT to attack. We inserted the first payload (Payload 1 in Figure 8) at the output of the encryption cipher eventually leading to a modification in the regenerated signature. Simultaneously, the inserted HT attacks the comparator output using a second payload (Payload 2 in Figure 8). This payload forces the output of the comparator to show a match even when the compared signatures did not match.

It is important to note here that the comparator testing logic is periodically checking for this specific case. In our simulations, the periodicity was set to 16 iterations, i.e., the Test Mode in Figures 6 and 7 is set to ‘1’ after 16 sets of data have been processed through the architecture. The Test Mode is asserted for only one clock cycle where the system is stalled and the comparator output is checked for legitimate operation.

Thus, the release logic might transmit altered encrypted data depending on when the HT is triggered. However, performing the testing periodically can solve the problem if the sets of data between two consecutive tests (in our case, 16 sets) can be declared invalid if attack type 3 is detected (a multi-bit Alarm Signal can encode different types of alarm conditions, e.g., a specific bit encoding of the alarm could be used to indicate failure of the comparator testing logic).

In our simulation, we triggered the HT after the processing of six iterations of data. After an additional ten iterations and as soon as the Test Mode was asserted, the comparator testing logic read the result of the comparison and alerted the release logic to halt the transmission of the data while signaling the alarm.

6.1.4 Simulation of Attack Type 4

To simulate attack type 4, and as in previous types of attacks, the HT had to wait for a specific trigger condition. Once the trigger was asserted, the HT attacked the low order bits of both the BCG input data and the analog-based signature as shown in Figure 9. This resulted in a modification of the values of Signature 1 and Signature 2. However, this time the modifications were minimal (below the comparator’s *threshold*). Thus, as expected, the comparator logic declared a match between the signals and the release logic released the encrypted bitstream of the modified data and signature.

6.2 Synthesis Results

Our synthesis results were performed using Synopsys Design Compiler version J-2014.09 for Linux and were mapped to the NCSU 45nm Base Kit Library [16].

Table 1 shows the area results of the main modules of our design post synthesis. It is obvious that a significant amount of area of the architecture is dedicated to the encryption/decryption modules. The security modules that are inserted to regenerate and test for the integrity of the data consume, as expected, significantly lower area.

Table 1. Synthesis Results.

Module	Area (square microns)
Encryption Cipher (PRESENT)	5517
Decryption Cipher (PRESENT)	5431
16-bit Multiplier	1293
16-bit Adder	141
Comparator Logic	297
Comparator Testing Logic	108
Release Logic	2414

To better show the area overhead imposed by introducing our HT detection technique, we present in Table 2 the overall area usage of the digital chip containing only the processing hardware and encryption/decryption units and compare it to the overall area of our modified architecture which includes the HT detection circuitry. An overhead of about 13% is introduced.

Table 2. Overall Area Consumption and Overhead.

Design	Area (square microns)	Overhead (%)
Regular Architecture	13,850	---
HT Detection Architecture	18,763	13%
HT Detection Architecture (not considering digital signature generation part of the signal processing)	18,763	35%

However, it is to be noted that in our experiments, the digital chip contained only encryption and decryption blocks. In more realistic scenarios, such a chip would contain other processing modules which require larger area. For instance, we assume the digital signature generation (i.e. calculation of the vector sum) is considered to be part of the BCG processing hardware. If that is not the case, the overhead of the detection architecture will be 35% as shown in Table 2. Our overall conclusion is that the percentage overheads reported in Table 2 can be considered pessimistic as increasing the overall chip area would cause a significant decrease in the overhead of our HT detection approach.

Our current design achieves a maximum clock frequency of $300\ MHz$. An analysis of the timing results shows that the multiplier that is used in the generation of Signature 2 in Figure 6 falls along the critical path of our architecture. We currently implement the squaring operations in our design using Synopsys DesignWare’s combinational carry save array multiplier. As reported by Synopsys [17], this type of implementation has a delay of $3.25\ ns$. If the application requires a higher clock speed a designer can choose to map the multiplier’s logic to other implementations. For example, DesignWare has a Booth-recoded Wallace-tree multiplier which has a delay of $1.6\ ns$ (for a 16-bit multiplier). In addition, DesignWare provides other options of pipelined and sequential multipliers. Choosing between these types of implementations allow the designer to make various area versus delay trade-offs.

7. DISCUSSION

In this paper, we do not discuss ways to foil attacks on the analog chip. However, the fabrication of the analog chip could be performed in a "trusted fab" where the chip can then be expected to be HT free. This option can be considered feasible as the analog chip can be fabricated using older, less expensive silicon fabrication technology.

In addition, attacks on the output of the chip are not considered. Our reasoning is that attacks on the way out of the chip can be caught by the next stage as the signature will not match the data.

One final comment can be made about the transmission of the signatures. Figures 6 and 7 show our architecture without explicitly showing encryption of the signatures. Clearly, sending the signature unencrypted might open an avenue of attack in later stages since an attacker may be able to exploit the unencrypted signature to reveal information about the encrypted data. To prevent these types of threats, encrypting the signature can be done prior to transmission. Specifically, in a more complete view of a System-on-Chip (SoC) including logic for transmission packet formation, Figures 6 and 7 can be modified as shown in Figure 10 in an SoC implementation to ensure only a properly encrypted bitstream is transmitted. Figure 10 shows multiple 16-bit analog-based signatures input to a FIFO buffer to form a block of 64-bit data that then can be fed to an encryption cipher, such as PRESENT [15], to form an encrypted bitstream that is ready for transmission.

8. CONCLUSION

In this work, we present a way of using analog signatures for HT detection on digital chips. We specifically take advantage of known relationships between health sensor data to create an analog signature and then check for its validity in reconfigurable digital logic. Our architecture targets small-sized HTs which, when triggered, attempt to modify the functionality of the design.

Our functional simulation results verified the effectiveness of our architecture in capturing different types of HT attacks including ones that target the architecture at a single point and others that try to foil the detection mechanism by attacking in a coordinated fashion on multiple points in the design.

Our synthesis results show that it is feasible to implement our HT detection architecture with minimal area overhead.

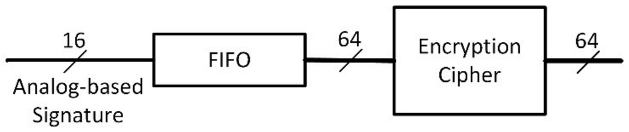


Figure 10. Encrypting the analog-based signature prior to transmission.

9. REFERENCES

[1] T. Wu, K. Ganesan, A. Hu, H. Wong, S. Wong, S. Mitra, "TPAD: Hardware Trojan Prevention and Detection for Trusted Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.PP, no.99, pp.1-17, August 2015.

[2] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10-25, Jan.-Feb. 2010.

[3] T. Wehbe, V. J. Mooney, A. Q. Javaid and O. T. Inan, "A Novel Physiological Features-Assisted Architecture for Rapidly Distinguishing Health Problems from Hardware Trojan Attacks and Errors in Medical Devices," in *IEEE Int'l Symp. on Hardware Oriented Security and Trust (HOST)*, Mclean, VA, USA, 2017, pp. 106-109.

[4] T. Wehbe, V. J. Mooney, D. C. Keezer and N. Parham, "A Novel Approach to Detect Hardware Trojan Attacks on Primary Data Inputs," in *Proc. of the Workshop on Embedded Systems Security (WESS)*, October 2015.

[5] A. Gbade-Alabi, D. Keezer, V. Mooney, A. Poshmann, M. Stöttinger and K. Divekar, "A Signature Based Architecture for Trojan Detection," in *Proc. of the Workshop on Embedded Systems Security (WESS)*, October 2014.

[6] J. Francq and F. Frick, "Introduction to hardware trojan detection methods," *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 770-775, 2015.

[7] S. Moein, J. Subramnian, T. A. Gulliver, F. Gebali and M. W. El-Kharashi, "Classification of hardware trojan detection techniques," *10th International Conference on Computer Engineering & Systems (ICCES)*, pp. 357-362, 2015.

[8] O. T. Inan, P.-F. Migeotte, K.-S. Park, M. Etemadi, K. Tavakolian, R. Casanella*, et al.*, "Ballistocardiography and Seismocardiography: A Review of Recent Advances," *IEEE Journal of Biomedical and Health Informatics,* 2014.

[9] G. K. Prisk, S. Verhaeghe, D. Padeken, H. Hamacher, and M. Paiva, "Three-Dimensional Ballistocardiography and Respiratory Motion in Sustained Microgravity," *Aviat Space Environ Med,* vol. 72, pp. 1067-1074, 2001.

[10] I. Starr, A. J. Rawson, H. A. Schroeder, and N. R. Joseph, "Studies on the estimation of cardiac output in man, and of abnormalities in cardiac function, from the heart's recoil and the blood's impacts; the ballistocardiogram," *American Journal of Physiology,* vol. 127, pp. 1-28, 1939.

[11] B. Curtis, "Delivering security by design in the Internet of Things," *IEEE Int'l Test Conference (ITC)*, pp. 1-1, 2014.

[12] A. Q. Javaid, N. F. Fesmire, M. A. Weitnauer, and O. T. Inan, "Towards robust estimation of systolic time intervals using head-to-foot and dorso-ventral components of sternal acceleration signals," *IEEE 12th Int'l Conf on Wearable and Implantable Body Sensor Networks (BSN)*, pp. 1-5, 2015.

[13] H. Ashouri and O. T. Inan, "Improving the Accuracy of Proximal Timing Detection from Ballistocardiogram Signals using a High Bandwidth Force Plate," *IEEE Biomedical and Health Informatics Conference*, 2016.

[14] R. F. Coughlin and R. S. Villanucci, *Introductory Operational Amplifiers and Linear ICs: Theory and Experimentation*. Harlow, United Kingdom: Pearson Education Limited, 1990.

[15] A. Boganov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems, LNCS, Springer-Verlag*, September 2007.

[16] NCSU 45nm FreePDK^{TM} Process Design Kit. Electronic Design Automation, North Carolina State University. Available at: http://www.eda.ncsu.edu/wiki/FreePDK

[17] A. H Syed, "Performance of Different Multipliers in the DesignWare Building Block IP," *DesignWare Technical Bulletin Article*, Synopsys Inc. Available at: https://www.synopsys.com/dw/dwtb.php?a=multiplier_bldg_block [Accessed on: April 8, 2016.

Synthesis of VLIW Accelerators from Formal Descriptions in a Real-Time Multi-Core Environment

Johnny Öberg
Dept. of Electronics, KTH Royal Institute of Technology
Sweden
johnnyob@kth.se

ABSTRACT

Designing, programming and design space exploration of predictable Real-Time systems on Heterogeneous Multi-Core platforms is a very complex task. The increasing validation costs and time-to-market pressure creates a desire to build systems that are correct by construction.

Formal description based on Model of Computations (MoCs) is a convenient way to create high-level models of such systems. The MoCs provide abstraction and high level modeling through a clear set of rules based on mathematics, which can be used as input for system synthesis. A formal synthesis flow would then ensure that the resulting real-time system is both predictable and correct by construction, provided that all transformations used in the flow can be verified/trusted.

In this paper we show how a Real-Time computation node in an MPSoC system, described using the Synchronous MoC, can be transformed into a VLIW accelerator. The created accelerator is incorporated as a computation node in a heterogeneous multi-core system implemented on an FPGA.

1. INTRODUCTION

As we approach the Sea-of-Cores/Processors era [1], System-level design (SLD) is considered the next frontier in electronic design automation (EDA) [2]. To successfully navigate this sea, new design methods for instantiating, configuring, programming, and validating these systems, together with automatic methods for exploring the design space, is needed. In SLD, resources are defined in terms of abstract functions (system behavior) and blocks (system architecture). Design targets include both software (SW) and hardware (HW), which in many cases is generated automatically to guarantee correct functionality, with design properties close to optimal performance and resource utilization.

Using a formal description based on Model of Computations (MoCs) is one way to achieve correctness by design. A MoC provide abstraction and a clear definition of the systems behavior, in the sense that the MoC describe the semantics of computation and concurrency of the processes in the system, i.e., how the computations communicate, e.g. they are used to model the abstraction of time explicitly [3]. The synchronous MoC is of particular interest, since it allows for description of systems reacting periodically within strict time bounds, which can be used to describe real-time oriented applications.

ForSyDe (short for Formal System Design) is a language defined for system modelling [4][5][6]. It is based on the functional programming paradigm, and allows the designer to model a system as a set of communicating concurrent processes. It is a Multi-core ignorant ESL based on SystemC. With each function, a process constructor is associated, that determines its Model of Computation. When the design is ready, one of several backends/platforms is selected for implementation. Currently, they support GPGPUs, the CompSoC platform from TU Eindhoven, the NoC-System Generator from KTH, and direct synthesis to VHDL.

In this work, we have selected the NoC-System Generator (NSG) tool from KTH for generating the target platform. The NSG tool is a fast-prototyping tool that allows a designer to quickly explore the design space and get a working MPSoC implementation running on soft-cores on an FPGA in very short time. It has a GUI that allows a designer to directly specify the functionality in C, either by dropping C-code in the right section of a process constructor, or by importing C-code generated by third-party tools like Simulink [7][8][9][10].

However, when implementing Real-time systems on FPGAs, we often run in to the situation that the combined Worst Case Execution Time (WCET) of the tasks/processes running on the soft CPUs are too slow compared to the real-time constraints. If no more soft processors can be added to the system due to size/power constraints, it is necessary to move the computations to an accelerator. In the ForSyDe methodology/philosophy, this should be implemented as a correct-by-construction transformation, which provides seamless integration of the produced accelerator into the existing system.

In this paper, we show how a computation node in an MPSoC can be transformed into a Very Large Instruction Word (VLIW) accelerator using a variant of standard High-Level-Synthesis (HLS). The starting point is a Synchronous model of an industrial relevant example, modelled using ForSyDe design principles, and then implemented as a transformation compatible with the ForSyDe/NoC-System Generator backend tool. After transformation, the VLIW accelerator is stored as an IP Block, and put back into the NSG design flow, where it replaces the original node. We also show that the resulting system fulfills the real-time constraints.

2. HLS vs VLIW generation

Most methods for designing accelerators start at the processor level. They typically analyze the given C-code and try to identify proper functions that are suited for acceleration. The identified functions are then synthesized and the control of them is stored back into the assembly code using unused codes in the instruction set or using predefined custom instructions. An example of the former is the Molen co-processor [11], targeted for Xilinx, while the latter is used in the Altera NIOS processor [12]. The main advantage of these approaches is the tight integration of the accelerated function with the processor. The major problem with them is that transferring parameters and results to and from the functions take considerable processing time. Also, interrupts on the main processor requires special handling while an accelerated function is executed.

Other approaches, like [13] start with an implementation of a VLIW, analyze the C-code, and then try to optimize the implementation of it by removing unused wires and registers. The advantage with this approach is that it becomes more similar to HLS, while the disadvantage is that they still require more bits to store the entire instruction space of the VLIW. The No Instruction Set Computer (NISC) processor approach [14][15][16], takes this method one step further in the direction of HLS by deriving the microcode directly, by using the control bits of any given data path as the instruction words, and then compiles the C-code for the intended data path.

Our approach is similar to the NISC-approach in the sense that we also let the operators residing in the data path decide the instruction set, with the difference that we have chosen to encode the control bits for each operator into an instruction to get a unified instruction width, which simplifies VHDL code generation.

Another difference with our approach is the view of the Register File (RF). In the NISC-approach, the number of registers in the RF is restricted. We do not have this restriction. Rather, the size of the register file is adjusted to the number of required registers to implement the functionality. This is not optimal from an area point of view, but it lets us maximize the throughput of the accelerated algorithms, since temporary variable storage in an external memory is no longer needed.

Removing the RF restriction also removes the final difference between NISC-architectures and High-Level Synthesis. In our opinion, there is no fundamental difference between compiling a C-program towards a VLIW NISC with unrestricted number of registers and implementing it using HLS.

Consider the data paths shown in Figure 1 below. In the HLS case, each operation in the data path is supplied with two operands, either from the registers in the design, from an input port or it is set to some constant. The result is stored back to a register or forwarded to an output port. Thus, the set of all registers can then be viewed as a (multi-ported) register file, where the enable bits of the registers correspond to the addressing bits of the registers in a register file used in a VLIW. In the same way, we can view the operations in the data path as the operations in the execution pipelines of the VLIW. An operation not used in a control stage of the HLS is equivalent to a NOP-instruction in the VLIW case. As the number of implemented instructions grows, and more and more registers are used in the RF, the HLS data path becomes more and more similar to a VLIW data path. Thus

$$\lim_{\#instructions \to \infty} HLS = VLIW_{NISC}$$

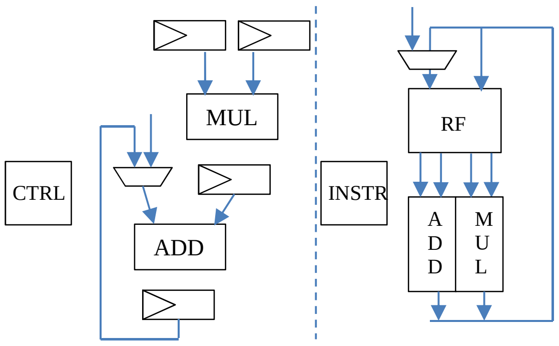


Figure 1. Typical HLS Datapath vs a typical VLIW Datapath

3. ForSyDe/NSG Synchronous MoC Semantics

In the ForSyDe Methodology, a system is described as a set of concurrent communicating processes. Each process has a *process constructor* associated with it, where the type of constructor decides its Model of Computation, i.e., the execution semantics of the computational part of the process and how it communicates data over a communication channel. Each process constructor has an *Init* function and a *Main* function. The *Init* function specifies what should happen when booting the system, while the *Main* function specifies what should happen during normal operation. The functions are not allowed to have side effects, i.e., no global variables are allowed.

Simulink [17] is a system-level language favored by industry to describe real-time control systems. Caspi et al [18] identified the periodic execution semantic of a subset of Simulink blocks, by transforming the Simulink model into an intermediate layer, described in the language Lustre [19], which executes functions periodically triggered by a synchronous signal.

Although automated mapping and refining of system level models onto MPSoCs has been shown to be very difficult to achieve[20], Robino et al showed in [10] how Simulink system models can be imported to the ForSyDe/NSG-tool, and then manually mapped to a MPSoC implementation. They extracted the C-code produced by the Embedded Encoder, and dropped it into a Synchronous MoC (SMoC) process constructor. The implementation of the SMoC itself was done using their Heartbeat methodology [9], which suggests using a globally distributed signal/clock together with a bare-metal SW layer to provide the synchronization to processes run on the MPSoC's CPUs.

Their method is similar to the Time Triggered Architecture (TTA) model [21], which enables events to happen periodically during specific time slots. However, the TTA model requires an OS to support the synchronization between Processing Elements (PEs) of the target MPSoC, thus requiring large memories, in contrast to the low memory overhead imposed by the bare-metal OS of the NSG-tool.

The semantics of the Synchronous MoC implementation in the Heartbeat Methodology is similar to how Synchronous HW is functioning. The Worst Case Execution Time (WCET) of the processes correspond to the Critical Path in the combinational part of a circuit, while the communication channels correspond to registers placed between the combinational parts. The Worst Case Communication Time (WCCT) on the communication channels corresponds to the setup time of the registers plus the propagation delay on the wires. The reset signal to the registers corresponds to an initial value function that the processes execute while the processor is booting, see Figure 2, below.

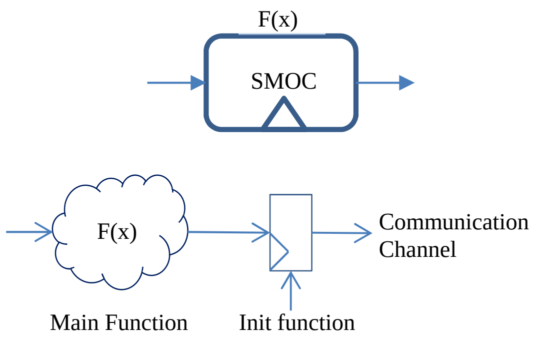


Figure 2. SW Process execution semantics for a Synchronous Model of Computation (SMOC) in the Heartbeat Methodology, modelled as a HW circuit equivalent.

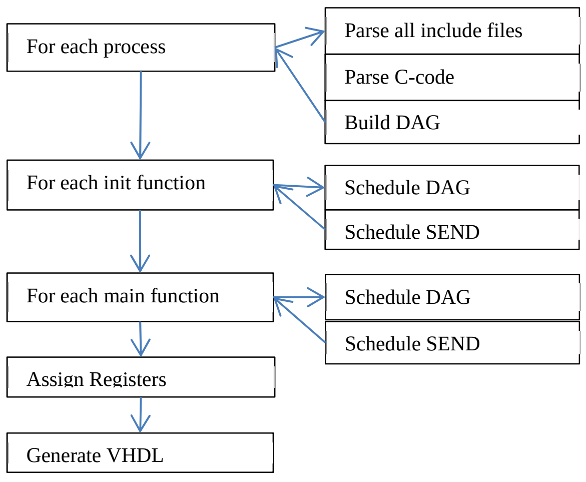
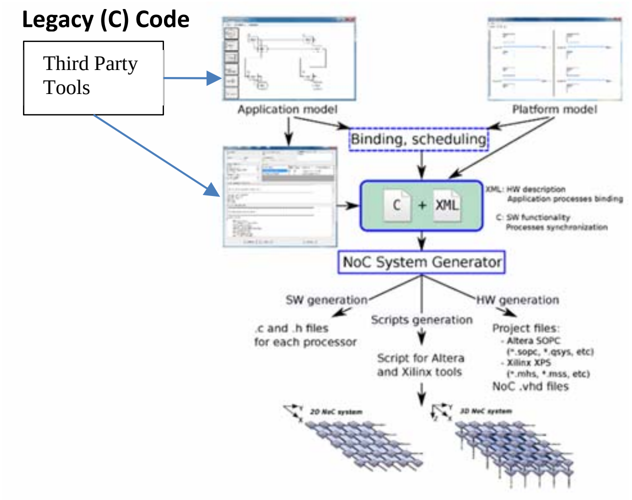


Figure 4. VLIW Synthesis process

4. VLIW accelerator generation

4.1 The Calc2HW transform

When a designer uses the NSG-tool to generate an MPSoC system for an FPGA platform, the tool generates the corresponding image of the system in a form understandable by the target FPGA technology (mhs-files for Xilinx, sopc/qsys-files for Altera), sets up the appropriate SW-structure so that it can be compiled, and instantiates any IP-blocks used in the system, including the NoC structure implementing the communication system. After logic synthesis, the FPGA can be configured with the system. The resulting system is easy to debug due to the well-defined semantics of the process constructors, letting the designer focus on how to get the functionality right. The ForSyDe/NSG design flow for creating the system is shown in Figure 3.



Create HW System (GUI/edit XML)
Place & Edit/Import SW processes onto HW nodes
Generate FPGA Image
Debug
Measure WCET on target platform
Improve design until design constraints are met.

a) Redistribute SW processes to other nodes to even out total WCET on each processor.
b) Create VLIW accelerator from SW processes at a node and replace node contents with it.
c) Regenerate FPGA Image

Figure 3. ForSyDe/NSG Design Flow

In the case that the Real-Time constraints of executing the processes on the CPUs on the platform cannot be met, an accelerator of the culprit process(es) needs to be created. This can either be done using 1) third-party tools, 2) existing HLS tools available in the target FPGA technology, or 3) inserted by hand.

We have defined a transformation that allows to transform synchronous ForSyDe/NSG C-processes, that contain pure calculations, into a NISC-style VLIW accelerator using a variant of standard High-Level Synthesis. To allow seamless integration with the rest of the system, we use the streaming Direct Access Port (DAP) provided by the NoC's Resource Network Interface (RNI), and replace/accelerate everything residing in the target node. The generated VLIW is stored as an IP Block, and then put back into the NoC, replacing the original node, before performing logic synthesis. The synthesis process is outlined in Figure 4.

4.2 VLIW Synthesis Process

After running the ForSyDe/NSG-tool, all SW process files end up in a sub-directory of the target directory. Besides the C-code for the processes, it also contains a software_configuration.h file that is generated by the NSG-tool. It contains information about how the communication channels are mapped on the RNI of the NoC. This information is parsed together with the rest of the C-code, and a Directed-Acyclic-Graph (DAG) is built of each *Init* and *Main* function in the process file. All assignments must be pure expressions, where the result is stored in a double precision floating point variable. Hierarchical function calls, for-loops and integer arithmetic are not currently supported in the current implementation. Port accesses are translated into STORE-instructions that directly access the RNI memory through the DAP and communication primitives are translated into SEND-instructions. If-statements are translated to CMP-instructions that set the Sign and/or Zero flag. All instructions have a conditional field that tells whether or not it should be executed conditionally based on these flags.

After the DAGs of the functions have been built, the *Init* functions are scheduled first, and then the SEND-instructions are scheduled to ensure that they end up last. The last used memory position used for the *Init* functions is stored to remember the start

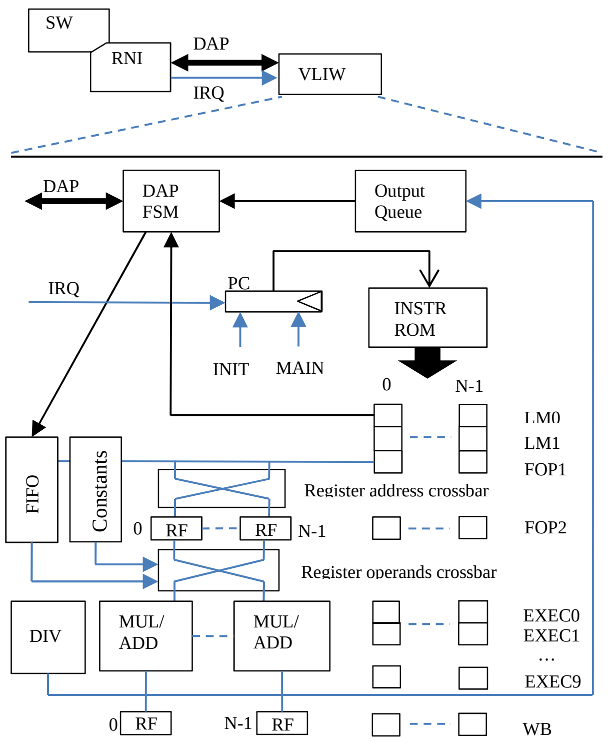


Figure 5. Target VLIW architecture.

address of the *Main* code. The procedure is then repeated for all Main functions.

Instructions are scheduled using an ASAP-methodology, described later in section 4.4, with a preference to lower execution pipe numbers. An instruction can never be scheduled earlier than the time slot where it is written to a register, nor before the start of the function type it belongs to (*Init/Main*). In case of scheduling conflicts (*result is not available yet, conditional boundary, etc*), NOPs are inserted, and the instruction scheduled at a later time step. A new instruction can replace an earlier inserted NOP. Due to its large size, only a single division unit is allowed. ADD/MULs are relatively small, so each Execution pipeline is allowed to have one each.

After all functions have been scheduled, registers are assigned. A new register is allocated whenever needed, and de-allocated when it is not used anymore, making sure that the number of used registers never exceeds the number of maximum alive variables. Global variables that should remember their value from each execution round are kept alive across the iterations. The register allocation algorithm is described in detail in section 4.5.

The resulting schedule is used to configure and generate the VLIW. The instruction schedule is checked to determine the number of unique instructions. For each used instruction type, an instruction code is assigned. Thus, the VLIW will never use more instruction codes than necessary, making it a NISC style VLIW.

4.3 Target VLIW Architecture

The target VLIW architecture is shown in Figure 5 above. It contains as many execution pipelines as needed, either because of timing constraints or because of user constraints. The generated accelerator contains an FSM that handles the communication with the RNI of the NoC. The RNI has a Direct Access Port (DAP) that allows streaming applications to read and write data quickly. Only one memory access can be done per VLIW word.

Whenever a Heartbeat of the system happens, the RNI sends an interrupt signal to the VLIW. The first time this happens after system reset, the *Init* function is executed. For all IRQs after this, the *Main* function is executed.

The VLIW execution pipelines consist of 15 execution stages, of which 10 are execution stages used by the floating point operators. The LM0 initiates a Port read if required. The LM1 stores the result of the read in a FIFO. The FOP1 stage forwards the addresses to the RF files and the constant ROMs. Constants are always positive when stored and accessible from all pipeline stages simultaneously; i.e., they are synthesized to one logic circuit per execution pipeline.

In the FOP2 stage, the accessed values are routed to their respective destination. The functions are executed in the EXECn stages. The number of stages is determined by the minimum number of pipeline stages that is required to run the ADD, MUL and DIV floating point units at the available system clock. At present, the Calc2HW transformation only functions for Altera platforms. The minimum number of execution stages their floating point IPs has in common is ten.

During WB, the results are stored back, either to a port (memory access) or to a register. Two reads and one write is allowed to any RF at any one time (multiple accesses to the same register are allowed). Each execution pipeline has one register file (RF) associated to it. To save one crossbar, the execution pipelines writes only to its own RF.

Results written to output ports are stored in an output queue. The FSM handling the DAP-port prioritizes read accesses since the execution pipeline cannot be stalled; values to output ports are written whenever an empty slot is available.

4.4 Scheduling Algorithm

The ScheduleDAG algorithm, shown in Figure 6, takes the instructions in the order they were compiled and assigns them to an execution pipeline. The insertion place is the earliest place that it can be scheduled. The pipe_nr is determined by the

```
ScheduleDAG()::
  for(i=0;i<nr_of_ instructions_in_dag;i++) {
    if (Instr(i)!=SEND) {
      earliest_condition_start=base_earliest_start;
       if (Instr(i)->ConditionalInstruction()) {
          earliest_condition_start=FindConditionInDAG(Instr(i));
      };
      earliest_start_op1=VariableLastWrite(Instr(i)->Op1());
      if (earliest_start_op1==-1)          {// No dependencies
         earliest_start_op1=base_earliest_start;
      } else {
         earliest_start_op1+=WriteBackLatency;
      }
      … // repeat for op2
      earliest_start=min(earliest_start_op1,earliest_start_op2);
      earliest_start=max(earliest_start,earliest_condition_start+1);
      earliest_start=max(earliest_start,process_earliest_start);
      new_earliest_start=CheckConflicts(Instr(i),
                                              earliest_start,nr_of_pipes);
      if ((pipe_nr<0) && (earliest_start==new_earliest_start))
          new_earliest_start++; // This line is full
      while(earliest_start!=new_earliest_start) {
          earliest_start=new_earliest_start;
          new_earliest_start=CheckConflicts(Instr(i),
                                              earliest_start,nr_of_pipes);
          if ((pipe_nr<0) && (earliest_start==new_earliest_start))
             new_earliest_start++; // This line is full
      };
      PadPipeWithNopsUntil(pipe_nr,earliest_start);
      AddInstructionToPipe(Instr(i),pipe_nr,earliest_start);
    }
  }
```

Figure 6. Pseudo code of the ScheduleDAG Algorithm

CheckConflicts() algorithm, which checks the current VLIW instruction line to see if it is allowed to schedule the instruction to a pipe on that line. If there is a conflict, the algorithm returns the next potential line (i.e., earliest_start+1) that the instruction can be assigned to. The assignment rules that are checked against are:

1. Max two different reads from the same RegFile.
2. Max one Port read per VLIW line, unless they refer to the same port address.
3. Max one Port write per VLIW line.
4. If target variable value is “global”, i.e., its value should be kept across iterations, the instruction must be scheduled in the same pipe that already holds the variable.
5. The target pipe must contain a NOP instruction, i.e., no instruction should have been scheduled to this pipe_nr.
6. In case there are several choices, assign it to the place which has the lowest pipe_nr.

4.5 The AssignRegisters()Algorithm

The AssignRegisters() algorithm, shown in Figure 7, goes through all instructions in the schedule. If the instruction is reading a variable for the last time, the register associated with that variable is deallocated. In case a variable is used for the first time, a new register is allocated for it. Variables that should keep their values across iterations do not have a stop cycle. Registers are allocated the first time they are written to, and are then never deallocated.

```
AssignRegisters()::{
   for(i=0;i<nr_of_pipes;i++)  {
        nr_of_registers[i]=0;
        used_nr_of_registers[i]=0;
   }
   for(i=0;i<nr_of_instructions;i++) {
       for(j=0;j<nr_of_pipes;j++) {
           if (Instr(i,j)!=NOP) {
                 nr=VariableNr(Instr(i,j)->Op1());¨
                 if (Variable(Nr)->StopCycle()==i) {
                     (reg_nr,pipe_nr)=ObtainVariableAssignment(nr);
                     if (!(reg_nr,pipe_nr) already released on this line)) {
                          // Release register
                          reg_hash_table[pipe_nr][reg_nr]=-1;
                          nr_of_registers[pipe_nr]--;
                     }
                 }
                 …// redo above for Op2 before checking target
                 nr=VariableNr(Instr(i,j)->Target());
                 if (Variable(Nr)->StartCycle()==i) {
                    // Allocate new register
                     nr_of_registers[j]++;
                     used_nr_of_registers[j]=max(nr_of_registers[j],
                                                  used_nr_of_registers[j]) ;
                     for(k=0;k<nr_of_registers[j];k++) {
                          if (register_hash_table[j][k]==-1) {
                              register_hash_table[j][k]=nr;
                             Variable(nr)->RegNr(k); reg_nr=k;
                             Variable(nr)->PipeNr(j); pipe_nr=j;
                              k=nr_of_registers[j];
                         }; // if
                     }; // for
                     Instr(i,j)->RegNr(reg_nr);
                     Instr(i,j)->PipeNr(pipe_nr);
                 }; // if
           }; // if
      }; // for
   }; // for
   for(i=0;i<nr_of_pipes;i++) {
        total_nr_of_registers+=used_nr_of_registers[i];
   };
};
```

Figure 7. Pseudo code of the AssignRegisters() algorithm

5. EXPERIMENTS

The starting point of the experiments is a Synchronous model of an industrial relevant example, a Motor Controller (MC), modelled in the ForSyDe/NSG using ForSyDe design principles, and then implemented as an 2x2 NoC-based MPSoC on an Altera FPGA DE2-115 board. The equations for the MC are derived from the descriptions of the MC algorithm found in [22]. The block diagram of it is shown in Figure 8 a) below. The algorithm contains one access to a sin(x) and one cos(x)-function from the math.h library. These two functions were trapped in the C-parser and implemented as a MacLaurin-expansion using a series of ADD/MUL-operations for simplicity. The argument x was truncated to be within $\pm 2\pi$ during calculation. The example is compiled into 185 instructions, with little parallelism inside. Thus, it is expected that any schedule of it will contain a lot of NOPs.

The SW view and the HW views of the base setup for the experiments in the next section is shown in Figure 8 b-d). The target node that will be replaced with an accelerator is the one in the upper right corner. As a reference to trace down bugs, the MC was cloned and run as pure SW in the lower left node. Both processes send the result to the lower right node, where the results were compared. For the design space explorations below, the MC process was cloned N additional times, and the clones put in the upper right corner of the system before the Calc2HW transformation was applied.

5.1 Design Space Exploration

In the first experiment we do a small design space exploration to figure out many MC algorithms that can be simultaneously implemented on one accelerator node without exceeding the real-time constraint of the algorithm, i.e., 3125 clock cycles (ccs). The Figure 9 below shows how the WCET (in ccs) varies with the number of MCs and the number of execution pipelines. We can see that there is no problem at all to meet the WCET deadline on the VLIW, even if we let the VLIW run 20 MC algorithms simultaneously, by adding a second execution pipeline.

In the next experiment, we try to figure out how the size of the system’s various parts will vary with the number of MCs, providing that we still only use a single execution pipeline. The results are shown in Figure 10. The total system size grows linearly with the number of MCs, as expected. The units are in kLUTs, kDFFs, and kMemoryBits for the MC’s values and kLUTs for the Total System Area. As we can see, the number of memory bits used in the MC is a step function, as expected. As soon as the number of registers passes a power of two, another bit is added to the Register File Memory address, which makes the number of memory bits used jump up one notch.

In the final experiment, we do a design space to see how the area of the VLIW varies with the number of implemented MCs and the number of chosen execution pipelines in the VLIW. The smallest one is of course the single MC, with a single execution pipeline (x1). It is ~14.4 kLUTs. A double precision ADDSUB unit is ~1990 LUTs, a double precision MUL unit is ~1160 LUTs, while a DIV unit is ~7500 LUTs. The rest of the area is attributed to the Instruction ROM (synthesized into logic), and RF control. The largest one that we synthesized that still meet the timing constraint of 3125 ccs, is the MC20x2 with ~45 kLUTs.

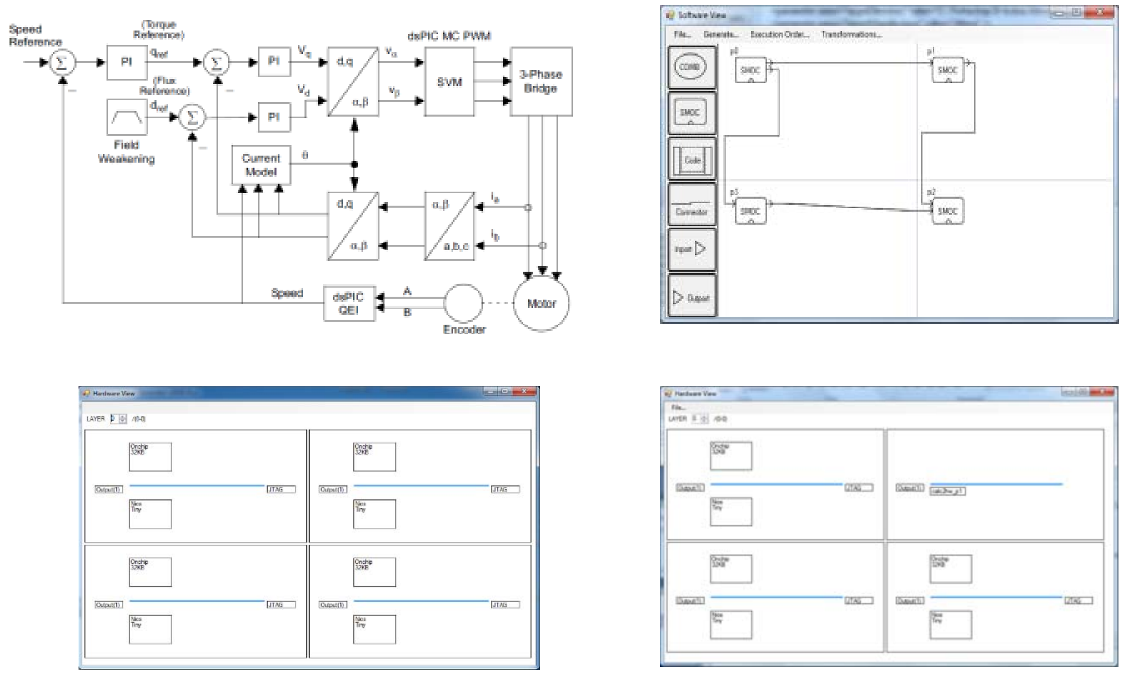


Figure 8. a) Block diagram of the MC algorithm [22] b) SW view of experimental setup. c) HW view before Calc2HW transformation. d) HW view after Calc2HW transformation.

5.2 Discussion

The MC algorithm has a hard real-time requirement, it has to be run at 16 kHz, which means it has to compute its value within 62.5 us (=3125 ccs). To achieve a predictable system, the flit insertion rate of the RNI was reduced to ensure that all flits will have time to reach its destination without conflicts. This resulted in a NoC bandwidth (BW) of 100 Mbit/s, the same as an external Ethernet connection. The MC sends four double precision values plus two 32-bit setup words, i.e., 320 bits per message. This means that the Best Case Communication Time (BCCT) is 3.2 us. In theory, this means that we can implement 19 MCs before exceeding the NoC BW. However, we also have to take into account when the first SEND command will be issued to the pipeline. This effectively restricts the implementation to 15 MCs.

However, it should be noted here that a 2x2 NoC, with the SW placed as shown in Figure 8 b), will never experience any collisions. Thus, it will always be predictable, which means that we could without harm double the injection rate to the NoC. The NoC BW would then increase to 200 Mbit/s, and 30 MCs can be implemented per node. Now, we can also insert another accelerator in another node if there is enough space on the FPGA, then making it possible to run 60 MCs on the FPGA. However, since the board we have only have one Ethernet connection with 100 Mbit/s in and out of the FPGA, we cannot feed the VLIW with enough data to compute so many MCs, Thus, the main bottleneck of the implementation is how to get data in and out of the FPGA.

Also, it should be noted here that the WCET of the SW implementation of the MC running on the comparison node is ~500 us, ie., it is too slow to run at the required speed at a NIOS II/e. Comparisons with the SW node was done at 1 Hz, so that the alt_printf statements to the terminal in the receiving end would have sufficient time to complete its execution.

6. CONCLUSION & FUTURE WORK

In this paper, we have presented how processes implemented using the synchronous Model of Computation, running on a computation node on a NoC-based MPSoC System on an FPGA, generated by the ForSyDe/NSG-tool, can be transformed into a NISC-style VLIW accelerator. The transformation applied is a variant of standard High-Level-Synthesis (HLS), and the generated VLIW IP is used to replace the original Nios II/e computation node in the design. Since it is a done as a refinement transformation, the implemented accelerator will be correct by construction, once the transformation has been formally verified and properly integrated into the NSG-tool's design flow.

We have applied the transformation to an industrial relevant example, and performed a design exploration of the properties of the design if implemented on an Altera DE2-115 FPGA board. The resulting real-time system is guaranteed to be predictable since the NoC can be set up in a way that the traffic patterns on the NoC never collide, and there is only local SW running on the CPUs in the other nodes.

6.1 Future Work

In the current version of our transformation, only double-precision floating point computations are considered. In the future, we will adapt our C-parser and VLIW generator to be able to handle a more elaborate set of types, like pixel structs, integer and fixed point data types. Further, we will also investigate what happens if we apply the transformation to smaller examples with large memory requirements, such as FIR filters and FFTs, so that we may compare the results with what other tools can produce.

What most current HLS approaches have in common is that they all target integer/fixed-point implementations of DSP algorithms, with the primary objective to save area and get it to run a bit faster. However, from an industrial point of view, this approach is slightly problematic. The whole idea with HLS from the beginning was to allow SW engineers to design HW accelerators. Converting double-precision Simulink-functions and algorithms written in Matlab to their fixed-point dittos is not a simple task. On the contrary, it requires quite some skills to get a numerical stable solution. In addition, recent studies have shown that for some applications, the fixed-point solution actually requires more energy for computing the same function than the same floating point version [23]. Thus, the study should also take design time and power consumption into account, to see which style of data types is more effective.

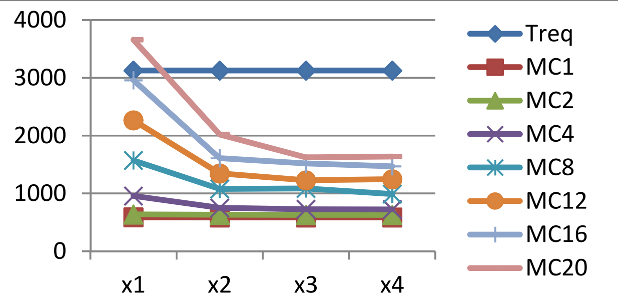


Figure 9. WCET (ccs) vs nr of execution pipelines in the VLIW

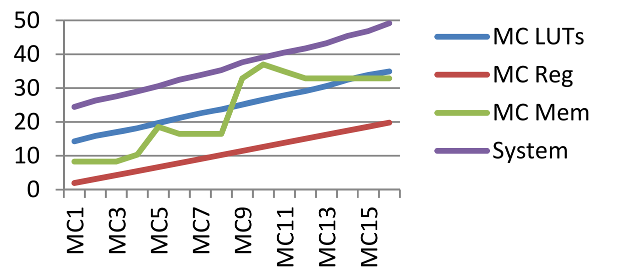


Figure 10. Total System Size vs # of MC calculations

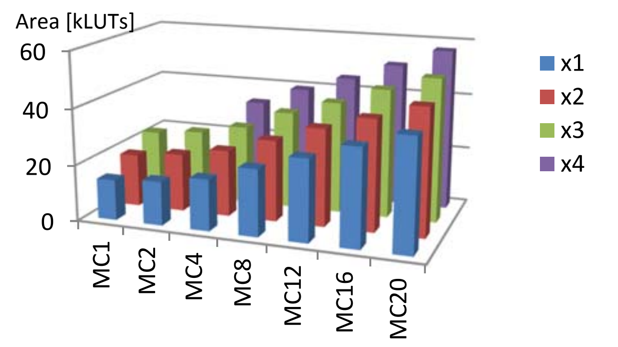


Figure 11. Total Area vs # Execution Pipelines in VLIW

7. REFERENCES

[1] Davidson, S.; "Sailing on a sea of processors," *Design & Test of Computers, IEEE* , vol.16, pp.112, Oct-Dec 1999.

[2] The International Technology Roadmap for Semiconductors (ITRS), System Drivers, 2011, http://www.itrs.net.

[3] Edward A. Lee and Alberto Sangiovanni-Vincentelli, "Comparing models of computation". In Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design, ICCAD '96, pages 234–241, Washington, DC, USA, 1996. IEEE Computer Society.

[4] S. H. Attarzadeh Niaki, M. Jakobsen, T. Sulonen, and I. Sander. Formal heterogeneous system modeling with SystemC. In Forum on Specification and Design Languages (FDL 2012), pages 160-167, Vienna, Austria, 2012

[5] S. H. Attarzadeh Niaki and I. Sander. An automated parallel simulation flow for heterogeneous embedded systems. In Proceedings of Design Automation and Test in Europe (DATE '13), pages 27-30, Grenoble, France, March 2013.

[6] https://forsyde.ict.kth.se/

[7] J. Öberg, F. Robino, "A NoC System Generator for the Sea-of-Cores Era". In proc. of FPGA World 2011. ACM Digital Libraries.

[8] https://forsyde.ict.kth.se/noc_generator

[9] F. Robino and J. Öberg. "The HeartBeat model: a platform abstraction enabling fast prototyping of real-time applications on NoC-based MPSoC on FPGA". In Proc. of ReCoSoC, 2013.

[10] F. Robino, J. Öberg, "From Simulink to NoC-based MPSoC on FPGA", In Proc. of DATE 2014.

[11] Vassiliadis, S.; Wong, S.; Gaydadjiev, G.; Bertels, K.; Kuzmanov, G.; Panainte, E.M.; "The MOLEN polymorphic processor", In Transactions on Computers, vol 53; issue 11, 2004.

[12] Nios II Custom Instruction User Guide, http://www.altera.com

[13] Matai, J. ; Oberg, J. ; Irturk, A. ; Taemin Kim ; Kastner, R.; " Trimmed VLIW: Moving application specific processors towards high level synthesis", In Proc. of Electronic System Level Synthesis Conference (ESLsyn), 2012, Page(s): 11 – 16.

[14] Bita Gorjiara, Daniel Gajski, "Custom Processor Design Using NISC: A Case-Study on DCT algorithm", In Proc. of the 3rd Workshop on Embedded Systems for Real-Time Multimedia, 2005, Sept. 2005, pp.55-60.

[15] Jelena Trajkovic, Samar Abdi, Gabriela Nicolescu, and Daniel D. Gajski, "Automated Generation of Custom Processor Core from C Code", In Journal of Electrical and Computer Engineering, Volume 2012 (2012), Article ID 862469, 26 pages, Hindawi Publishing Corporation.

[16] NISC Technology & Toolset, http://www.ics.uci.edu/~nisc/

[17] Mathworks. Simulink documentation center. Website. http://www.mathworks.se/help/simulink/.

[18] P. Caspi et al. "From Simulink to Scade/Lustre to TTA: a layered approach for distributed embedded applications". In Proc. of conf. on Language, compiler, and tool for embedded systems, LCTES, 2003.

[19] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language Lustre. Proc. of the IEEE, 1991.

[20] A. Sangiovanni-Vincentelli. "Quo vadis SLD: Reasoning about trends and challenges of system-level design". IEEE, 95(3):467–506, 2007.

[21] H. Kopetz and G. Bauer. "The time-triggered architecture". Proceedings of the IEEE, 91(1):112 – 126, Jan 2003.

[22] AN908 - Using the dsPIC30F for Vector Control of an ACIM, Microchip.

[23] D. Guenther, A. Bytyn, R. Leupers, G. Ascheid, "Energy-efficieny of floating-point and fixed-point SIMD cores for MIMO processing systems", In Proc. of SoC Symposium, 2014.

Room: Gasellen
Sessions C1-C2-C3
Session Chair: N/A

§ Mastering Clock Domain Crossing challenges in FPGA Design

Metastabilty from the intermixing of multiple clock signals is not modeled by simulation. Unless you leverage exhaustive, automated Clock Domain Crossing (CDC) analyses to identify and correct problem areas, you will inevitably suffer unpredictable behavior when you go to the lab or when the FPGA is used in the field. Automated CDC verification solutions are mandatory for multiclock designs. Questa CDC Solutions identify errors that have to do with clock domain crossings signals that are generated in one clock domain and consumed in another.

Presenter: Stefan Bauer, *InnoFour, Netherlands*.

§ Input power related challenges

In modern embedded systems, reliability and uptime is also related to what happens on the incoming power wires. There may be power interruption or surges, bringing down the system and possibly also causing permanent damage. The seminar covers what may happen to your system and how to protect it and maintain safe operation.

Presenter: Thomas Ginell, *Linear Technology,* now part of *Analog Devices*.

§ Xilinx Cost-Optimized Portfolio Spartan-7, Zynq Z-7000S

Presenter: Per Boström, *Avnet Silica*.

Room: Renen
Sessions A5-A6-C4
Session Chair: Lennart Lindh
FPGAworld

§ **Interface and visualization for accelerometer in VHDL** [15 min]

During a course in the vocational education TEIS (applied electronics in embedded systems), I designed a system on an FPGA in VHDL on the platform BeMicro MAX 10. The system logs an accelerometer's measured values and communicates via SPI. This logging is displayed on a VGA screen. The construction contains a small custom made processor that runs the entire system and manages the SPI bus, VGA drawing, and the peripheral devices.

Presenter: Anders Axelsson, *Sweden*.

§ **Study on usage of free IP for serial communication in the context of Cyclone-V SoC HPS/FPGA and Linux** [15 min]

The study is about how to use free IP for serial communication like SPI, I2C and UART implemented in the FPGA part of a CycloneV SoC together with software running under a Linux yocto system in the HPS.

To perform this study two different CycloneV SoC Developments boards were connected using three serial communication interfaces, SPI, I2C and UART. The test application transfers data in real time autonomously between the two boards while sampling input data to be transmitted from slide switches and presenting received data onto LEDs.

Presenter: Magnus Karlsson, *Innowicom System Solutions AB / AGSTU, Sweden.*

§ **Intel High Level Synthesis**

Increasing abstraction level when designing with FPGA may give significant productivity gains in both design and verification phase of a project. Join this session to learn more about Intel's HLS compiler due to release with Quartus 17.1 later this year. We will explain the fundamentals and give a little demo.

Presenter: Nikolay Rognlien, *Arrow*.

Room: Räven
Sessions C5-C6
Session Chair: Hichem Belhadj
Microsemi Corp., USA

§ Context-Aware Logic Replication for Higher Speed and Lower Power

Several designs with timing hurdles include large number high fanout nets that cause high delay penalties and routing congestion. This paper proposes a novel approach to tackle the logic replication and brings relief to the timing not only to the high fanout net, but also considers the logic context to ensure optimal logic replication.

Presenter: *Microsemi Corp., USA*.

§ Security-Conscious FPGA Design: The Rule and Not the Exception Anymore!

This paper intends to bring awareness of the MUST nature of considering security as the primary FPGA design goal by architects, design leads, engineers, and procurement/supply management teams. More importantly, Corporate Executives that care about their business and its reputation can no longer blame security breaches on their design engineers, or their IT team, it is their responsibility to mandate "all-out security schemes" to be implemented, to hire security experts, and to plan education programs for their organization . The bulk of the paper provides solutions and recommendations to help implement "comprehensive security-aware"-design.

Presenter: *Microsemi Corp., USA*.

§ Common RTOS-related bugs - How avoid and detect

This presentation will feature a commercial product that we sell (Tracealyzer) but this it is not a pure product presentation. Instead, we use screenshots from Tracealyzer to explain relevant RTOS concepts.

A longer presentation can be seen at
https://percepio.com/2016/12/14/common-rtos-related-bugs-how-avoid-and-detect/

Presenter: Johan Kraft, *Percepio AB, Sweden*.

§ FPGA in Neuroscience

In neuroimaging, the computational demand on the image processing pipelines is increasing as new methodological methods are improved. One computationally demanding method is to look at global brain connectivity in fMRI (functional MRI), where the brain activity of each voxel in the 3D brain volume is correlated over time with every other voxel to obtain a global measure of connectivity. Here, we look at how OpenCL on Intel Arria 10 FPGA can be used for parallel processing of fMRI data.

More information: One commonly used tool for global brain connectivity is 3dTcorrMap that comes with the software package AFNI. This tool is available in an OpenMP version to utilise multi-core processors. We have written our own custom made tool for global brain connectivity written in C and OpenCL for the Arria 10 FPGA and then compared the performance with the OpenMP version of 3dTcorrMap running across 4 CPU cores.

Presenter: Lars Forsberg, *Synective AB, Sweden*.

FPGAworld 2017 @ Copenhagen

Technical University of Denmark
SCION, Building 372, Diplomvej Lyngby
Denmark

Conference Programme

08:30 Registration

09:00 Conference opening
Lars Dittmann, Technical University of Denmark
and *Lennart Lindh, FPGAworld*

09:15 Key Note Session
Acceleration of Convolutional Neural Networks in FPGAs
Hans Holten-Lund, Prevas AB, Denmark

10:00 Coffee Break & Exhibition

10:30 Parallel Sessions

12:00 Lunch Break & Exhibition

13:00 Mike Dini Talk
FPGA events during the year that has gone and gossips

13:30 Parallel Sessions

14:30 Coffee Break & Exhibition

15:00 Parallel Sessions

16:00 Panel Discussion
What for skills & knowledge do a FPGA designers need today?
Moderator: Rolf Sylvester-Hvid, *Aktuel Elektronik (Danish Magazine)*

The exhibition will be open during the day.
Coffee will be served in the exhibition area.

§ Acceleration of Convolutional Neural Networks in FPGAs

Hans Holten-Lund, Prevas AB, Denmark

The keynote presentation will discuss some of the issues we face as FPGA designers when tasked with the computational loads involved in signal processing. New tools are appearing, aiming at making it easier to design signal processing blocks. Convolutional Neural Networks share many techniques with more traditional signal processing. Explore tradeoffs, design-time vs performance. Floating-point vs fixed-point math. GPUs vs FPGAs.

Hans Holten-Lund, is a Senior FPGA Designer at Prevas, and has a Ph.D. and M.Sc. EE from IMM, Technical University of Denmark. He has worked mainly on FPGA design for phased array ultrasound scanners, and and other embedded FPGA based systems, including computer vision. Also has industry experience with multi-gigabit networks and 3D computer graphics.

A longer CV is available here: https://www.linkedin.com/in/hans-holten-lund-a3a53114/

Room: N/A
Sessions C1-C2-C3
Session Chair: Lennart Lindh
FPGAworld

§ Mastering Clock Domain Crossing challenges in FPGA Design

Metastabilty from the intermixing of multiple clock signals is not modeled by simulation. Unless you leverage exhaustive, automated Clock Domain Crossing (CDC) analyses to identify and correct problem areas, you will inevitably suffer unpredictable behavior when you go to the lab or when the FPGA is used in the field. Automated CDC verification solutions are mandatory for multiclock designs. Questa CDC Solutions identify errors that have to do with clock domain crossings signals that are generated in one clock domain and consumed in another.

Presenter: Stefan Bauer, *InnoFour, Netherlands*.

§ The FPGA security challenge: high assurance on low cost devices

Traditionally, in secure chip enrollment a unique private key is generated and burned into one-time programmable memory and so relies on expensive continuous protection of an entity's private key. This presentation is challenging traditional cryptography by introducing a truly private keyless technology – a solution to the ubiquitous problem of managing and protecting private keys.

Presenter: Thomas Ginell, *Linear Technology,* now part of *Analog Devices*.

§ Intel High Level Synthesis

Increasing abstraction level when designing with FPGA may give significant productivity gains in both design and verification phase of a project. Join this session to learn more about Intel's HLS compiler due to release with Quartus 17.1 later this year. We will explain the fundamentals and give a little demo.

Presenter: Nikolay Rognlien, *Arrow*.

Room: N/A
Sessions A1-A2
Session Chair: Lennart Lindh
FPGAworld

§ **Constrained Random and Functional Coverage for VHDL testbenches controlled in a structured manner**

OSVVM provides a good library for CR and FC. But how should we apply this in a TB to avoid the normal verification traps?

- Bad overview
- Bad readability
- Bad maintainability & extensibility
- Inefficient reuse

Even most well-structured TBs do not sufficiently avoid these problems.

This presentation will show how easy it is to combine OSVVM and UVVM to get a ‘Unified VHDL Verification Methodology’ that provides advanced CR and FC, - and at the same time promotes overview, readability, maintainability, extensibility and reuse.

Presenter: Espen Tallaksen, *Norway*.

§ **Portable Stimulus Specification The Next Big Wave in Functional Verification**

In this paper we will describe the upcoming proposed standard for "Portable Stimulus Specification" (PSS) from Accellera. We will show how a single model of stimulus and scenarios can be re-used across different environments such as High-level C models, UVM simulations or even embedded SW, thus providing the verification engineers with a unified way to model interaction with complex SoC's or FPGA's containing CPU cores and embedded SW.

More information: Accellera has been working on the new proposed PSS standard since 2014. At DAC 2017 the Working group released the first "Early Adopter" version of the standard. This new proposed standard has received tremendous amounts of interest from the industry - at DAC and DVCon the seminars about PSS were completely overbooked and only standing room was available.

For more information please see http://accellera.org/news/press-releases/244-accellera-portablestimulus-early-adopter-specification-now-available-for-public-review

Presenter: Staffan Berg, *Sweden*.

Room: N/A
Sessions C4-A3
Session Chairs: Lennart Lindh
FPGAworld

§ Use of FPGA in high speed networking

Silicom Ltd. is an industry leading provider of high performance networking and data infrastructure solutions. At Silicom we are determined to help our customers boost their performance using the latest FPGA technology from both Xilinx and Altera.

Our products offerings include Cyber Security, Network Monitoring and Analytics, Traffic Management, Application Delivery, WAN Optimization, High Frequency Trading, virtualization, cloud computing and big data markets.

Presenter: Michael da Costa Carneiro, *Silicom Ltd., Denmark*.

§ The Impact of Place and Route on FPGA Logic Synthesis

For a quarter century, synthesizing an RTL design into an FPGA circuit has required only a loose understanding of the impact of Place and Route (P&R) software. By estimating route delays during Logic Synthesis based on graph properties and with accurate timing constraints, it was possible to achieve timing closure even for high-frequency clocks. In this presentation, we explain useful techniques to improve system performance and to achieve success in P&R more reliably.

Presenter: Pieter J. Hazewindus, *USA*.

Call for FPGAworld Conference 2018

Academic/Industrial Papers, Product Presentations, Exhibits and Tutorials
September 18th, 2018, **Stockholm**, Sweden, Academic & Industrial programs
September 20th, 2018, **Copenhagen**, Denmark, Industrial program only

Submissions should be at least in one of these areas

- DESIGN METHODS - MODELS AND PRACTICES
  - Project methodology
  - Design methods as Hardware/software co-design
  - Modeling of different abstraction
  - IP component designs
  - Interface design: supporting modularity
  - Integration - models and practices
  - Verification and validation
  - Board layout and verification
  - Etc.

- TOOLS
  - News
  - Design, modeling, implementation, verification and validation
  - Instrumentation, monitoring, testing, debugging, etc.
  - Synthesis, compilers and languages
  - Etc.

- HW/SW IP COMPONENTS
  - New IP components for platforms and applications
  - Real-time operating systems, file systems, internet communications
  - Etc.

- PLATFORM ARCHITECTURES
  - Single/multiprocessor architecture
  - Memory architectures.
  - Reconfigurable Architectures
  - HW/SW architecture
  - Low power architectures
  - Etc.

- APPLICATIONS
  - Case studies from users in industry, academic and students
  - HW/SW component presentation
  - Prototyping
  - Etc.

- SURVEYS, TRENDS AND EDUCATION
  - History and surveys of reconfigurable logic
  - Tutorials
  - Student work and projects
  - Etc.

www.fpgaworld.com