A SDN Solution for System-on-Chip World

Soultana Ellinidou*‡, Gaurav Sharma†‡, Jean-Michel Dricot*‡ and Olivier Markowitch†‡
*Wireless Communications Group – OPERA Department
† QualSec Group – Computer Science Department
‡Cyber Security Research Center
Université Libre de Bruxelles, Belgium
{soultana.ellinidou, gsharma, jdricot, olivier.markowitch}@ulb.ac.be

***Abstract*—System on chips (SoCs) are all around us in today's world. Therefore, in this paper we propose a flexible, technology-aware SoC design, named as Cloud-of-Chips (CoC), which is able to change its characteristics, such as routing logic, transmission paths, priorities, IC clustering, etc. We focus particularly on inside communication of CoC architecture. The basic idea of CoC is the creation of an architecture, which will be able to support all the requirements of a vast number of todays applications by adopting and changing its characteristics according to them. The valorization perspectives are of importance since the outcomes of this research will be applicable for embedded systems, real-time systems, communication systems, as well as for mainstream systems such as Internet-of-Things and Internet-of-Everything. As far as the inside communication of our CoC platform is concerned, we leverage on algorithms and strategies developed within the field of Software Defined Networking (SDN), which was introduced to deal with hardware redesign and ultimately provide cloud-like flexibility. At the end, we describe registration and authentication of every entity in our network.**

***Index Terms*—SDN, SoC, P2P communication, ID-based Cryptosystem**

I. Introduction

The rise of new computing paradigms such as Internet-of-Things (IoT) and Internet-of-Everything (IoE) (billions of devices foreseen in 2020) requires extremely versatile IC (Integrated Circuit) solutions. The IoT and the IoE have been proposed as the next revolution of digital systems application [2]. However IoT/IoE paradigms will bring a wide variety of applications, with highly variable volumes, for which traditional System-on-Chips (SoCs) are not capable of supporting them.

A SoC is an integrated circuit that integrates all components of an electronic system [6]. The SoCs provide the designers an opportunity to optimise the system design metrics, such as: performance, cost, size, power, and run-time. On a SoC, it is important to efficiently and optimally address the application requirements with respect to the system components. Furthermore the SoCs provide scalability and higher throughput for heterogeneous multi-core systems.

Network-on-chip (NoC) is a scalable solution to address communication problems for Multi-Processor Systems-on-Chips (MPSoC). NoC was introduced in order to scale down the concepts of computer networks, and be adaptable directly on the design process of a multi-core integrated circuits.

As of today, general designs of SoC platforms for multi-core Network-on-Chip (NoC) are usually not capable of supporting different processing requirements such as performance, power, reliability and response time due to the fact that most of them are designed for specific applications. The need of a versatile architecture, supporting a wide variety of applications (such as Internet of Things), leads to the design of an extremely configurable system both at design time and at run time.

The CoC refers to a large amount of interconnected ICs (Integrated Circuits) and IC cores, which can have different communication speeds and hierarchy levels. One of the key challenges in the CoC paradigm is the communication between the clusters of cores and ICs. For that reason an interesting approach could be to adopt the algorithms and strategies developed within the field of SDN. SDN emerged to support the dynamic nature of future network functions and intelligent applications while lowering operating costs through simplified hardware, software and management [12]. The approach proposed by the SDN paradigm is that the data travels across multiple nodes of the network and efficient and effective data transfer is supported by a centralized controller. The forwarding decisions are done first in the controller and then moved down to the overseen switches which simply execute these decisions. In the SDN paradigm, switches provide vendor-agnostic protocols for remote controllers, one of the most common communication protocol is OpenFlow [7]. This protocol provides to the controllers a way to discover the OpenFlow-compatible switches, defines the matching rules for the switching hardware and collects statistics from switching devices.

As far as the communication over the CoC is concerned, SDN turns the embedded system susceptible to security breaches. Among others, cryptographic protocols and schemes aim at proving confidentiality and integrity of exchanged information. Specifically, as far as the inside CoC communication, the fact that we have Point to Point (P2P) communication lead us to use identity-based cryptography to assist in the security and performance critical assignment of user identities in P2P systems.

The rest of the paper is organised as follows: In Section II, we refer to the related work about SoC platforms and the main challenges about proposed approach. In Section III, we introduce our three-levels architecture: Printed Circuit Board (PCB) Level, IC Level, Processing Cluster (PC) Level. In Section IV, we analyse the networking aspects of our platform introducing a new communication protocol called SoC-Flow.

978-1-5386-5899-4/18/$31.00 ©2018 IEEE

In Section V we describe the security aspects of our platform and it follows, the conclusion and future work in the last Section.

II. Related Work

A number of architectures and implementations for SoC platforms is presented in literature. Nomadik is a SoC platform aims to provide video-coding algorithms and chip implementation schemes for the mobile industry [1]. Another interested architecture presented on OMAP-2 platform, derives from the TI OMAP architecture, which is designed to address mobile entertainment and communications such as all-in-one smartphones or converged portable multimedia devices [8], [5]. Moreover, the PNX8550 (Viper2) set-top box SoC, based on the Philips Nexperia platform [6], is a highly integrated multimedia SOC targeted at advanced set-top box available on the market. Last but not least, Amazon Elastic Compute Cloud (Amazon EC2) [11] is a web service that provides secure, resizable computing capacity in the cloud. Amazon EC2 is a programmable instance with Field Programmable Gate Arrays (FPGAs) that creates custom hardware accelerations for your applications.

The [1] and [8] focus on the mobile and smart phone industry. However, the design in [6] is pertaining to set-top boxes SoC. The last platform, that is presented, is a new tool available from Amazon able to customize FPGAs for hardware acceleration. They are all heterogeneous multi-processor platforms and they use a central Reduced Instruction Set Computer (RISC) host processor to control the system: a MIPS for Viper-II and ARM processors for the OMAP-2 and Nomadik. As far as the inside communication of platforms, they use different approaches, however there are some notable similarities. All systems depart from the classic shared-bus architecture and use forms of interconnects that can be assimilated to partial cross-bars. The AMBA multi-layer of the Nomadik is one of the most known and used forms of a partial cross-bar. Much attention has been given in the design of the Viper-II interconnect to accommodate the different types of traffic on the SoC and thus address the issue of the latency predictability. The OMAP-2 interconnection takes a different approach to the same issue. Latency predictability is obtained by allocating pre-determined time-slices to the various connections in the system. Finally, all three chips use combinations of caches and of DMAs to hide the latency of the interconnect and of the memory sub-systems.

As far as the current state of the involvement of SDN in SoC is limited to a research state. Specifically The first research paper found in literature is [4], where the authors propose Software Defined Network on Chip (SDNoC) which splits the network into control and data plane and control logic is performed through the chip hardware design. The authors use the SDN approach as a routing method and they made a comparison between SDNoC and static XY routing and dynamic DyAD routing with traditional NoC architecture. Afterwards in [9], the authors applied SDN principles in order to propose a SDNoC architecture. This architecture is focused on abstraction layers and interfaces that permit its deployment in a modular fashion and it has the potential to overcome the NoC management problems in the many-Core era. However the authors propose an architecture without providing enough details about the security aspects or the communication protocols. Another interesting contribution of the same authors is presented on [10], where they evaluate the SDNoC architecture among the Processing Elements(PEs) in a many-Core system with System C simulator, focusing on the configuration time, delay, and throughput of their architecture.

III. Our Architecture

The CoC architecture consists of an embedded SoC connected with soft-multicore processors with the purpose of creating an MPSoC of many-cores imported on PCB. In order to design a heterogeneous system architecture, High Performance (HP) cores and Low Performance (LP) cores will be chosen to present the different perspectives. Specifically the different cores categorized according to their performance into Processing Clusters (PCs). Our CoC platform is comprised of a PCB of ICs where each IC contains scalable PCs of Cores [Fig. 1]. For the communication on a PCB, on-board SDN switches are placed on the boundaries of all ICs, in order to provide quick communication among the ICs and corresponding PCs. In order to provide communication among the PCs, hardware switches with SDN functionalities are adopted. Specifically, a SDN switch consists of Data Plane, which carries packet traffic and Control Plane, which handles the communication between switches and controllers. The network as a whole will follow a mesh topology where the path of each packet is dynamically mapped. As far as the controller we are planing to include one controller ion IC level, running on a dedicated PC and one central controller on PCB level, in order to carry the traffic between ICs.

Our work is to create an automated framework which provides a wrapping of the existing IPs (cores, processors) and complimentary hardware services, to create efficient hardware templates for CoC, according applications requirements. Using a flexible interconnection for each many-core template, we will achieve better on chip and CoC communication. With the FPGA dynamic partial reconfiguration our system will be able to use multiple hardware templates, at run-time according to the application requirements. From Networking side, we propose a new SDN protocol that supports a secure communication between multiple PCs. Moreover, we consider to use hardware on-chip SDN switches at PC level and SDN switches which communicate with a centralised controller, hence we will be able to apply forwarding rules for packet switching. About the security of our system, we propose to integrate atomic security primitives like encryption schemes, hash functions, random bit generators and implement them in the light of secure universal composability theory. For the sake of clarity, we investigate now each level of our architecture separately.

As far as the PCB Level, we consider to use the Parallella FPGA board, which is a high performance, credit card

978-1-5386-5899-4/18/$31.00 ©2018 IEEE

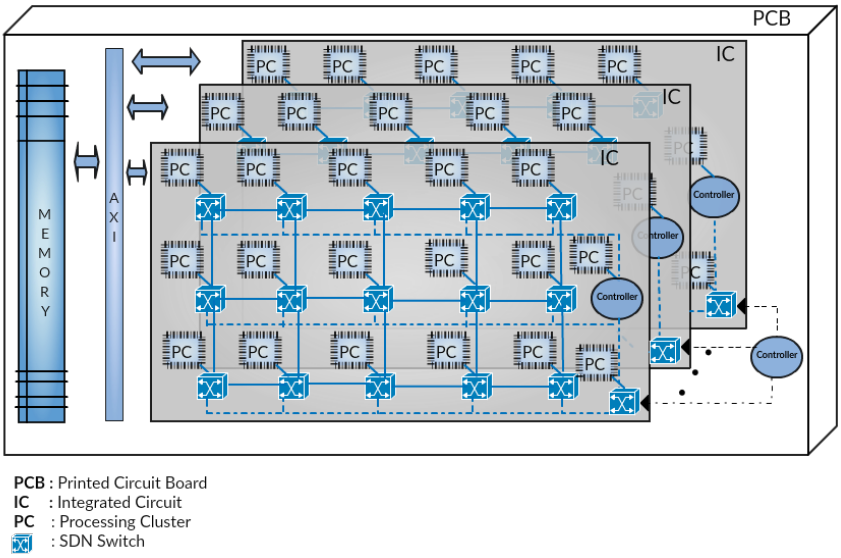

Fig. 1. CoC architecture

sized computer based on the Epiphany multi-core chips from Adapteva. The Parallella can be used as a standalone computer, an embedded device or as a component in a scaled out parallel server cluster. The Parallella includes a low power dual core ARM A9 processor and runs several of the popular Linux distributions, including Ubuntu.

An IC consists of multiple Processing Clusters and hardware switches. That hardware switches provide high speed inter-processor communication among PCs. It is important to mention that there is no local shared memory and the communication between different ICs (or clusters) is achieved through main memory and through a SDN switch, which is placed in the boundary of IC. As far as the hardware switches on PC level, which connect the on-chip processor nodes, they follow a mesh network topology and they efficiently handle traffic patterns in high-throughput real-time applications. The network takes advantage of spatial locality and an abundance of short point-to-point on-chip wires to send complete transactions consisting of source address, destination address, and data in a single clock cycle. Each routing link can transfer up to 8 bytes of data on every clock cycle, allowing 64 bytes of data to flow through every routing node on every clock cycle, supporting an effective bandwidth of 64 GB/sec at a mesh operating frequency of 1GHz.

IV. SDN Approach

The SDN model has three layers: Application, Control, and Networking. The three layers with some appropriate changes are presenting in our approach in order to be adaptable in our CoC platform . The main difference between a simple SDN architecture and our architecture is that we are considering both network entities: switches(Network Layer) and controllers(Control Layer) to have a session and long term key for the registration and authentication process. Furthermore, the fact that the switches are involved in different layers about the hardware lead us to the creation of different controllers to support the traffic in different hardware layers. The layers are the following;

- Control Layer- Responsible for making decisions on how packets should be forwarded by one or more network devices and pushing such decisions down to the network devices for execution. The primary job of the control plane is to fine-tune the forwarding tables that reside in the forwarding plane, based on the network topology or external service requests.
- Network Layer - Responsible for monitoring, configuring, and maintaining network devices, e.g., making decisions regarding the state of a network device.
- Application Layer - The layer where applications and services that define network behavior reside. Applications that directly (or primarily) support the operation of the forwarding are not considered part of the application plane.

A. Openflow Protocol

OpenFlow (OF) is one of the first SDN standards. This communication protocol provides the ability to the SDN controller to interact with the forwarding plane of network devices (switches and routers), both physical and virtual, to adapt according to the requirements. The OF architecture consists of three basic concepts: (1) The network is built up by OpenFlow-compliant switches that compose the data plane; (2) the control plane consists of one or more OpenFlow controllers; (3) a secure control channel connects the switches with the control plane.

An OpenFlow-compliant switch is a device which mainly forwards packets according to flow tables. Each flow table includes a set of flow table entries. Each flow entry consists of match fields, counters and instructions. In our SDN network we are going to use Open vSwitches (OvS), which are an

978-1-5386-5899-4/18/$31.00 ©2018 IEEE

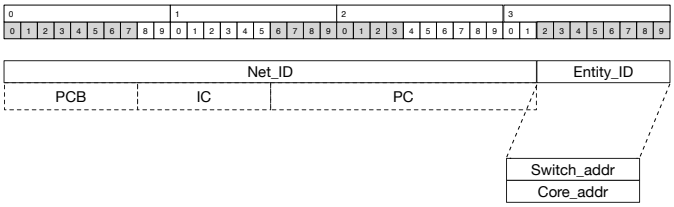

Fig. 2. Address format

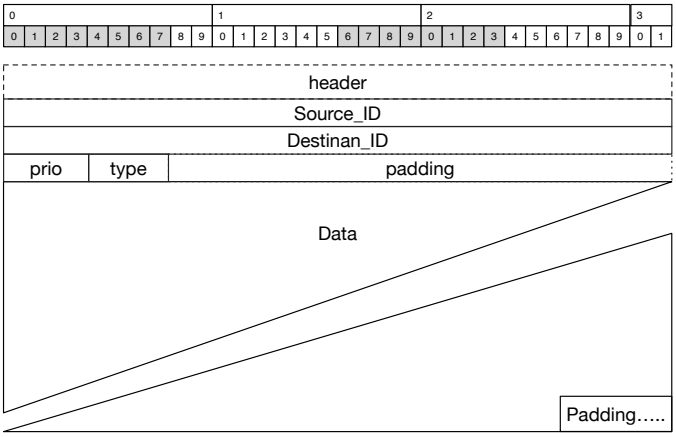

Fig. 3. Message format

open-source implementation of a distributed virtual multilayer switch. The main purpose of OvS is to provide a switching stack for hardware virtualization environments, while supporting multiple protocols and standards used in computer networks. OvS is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols. In our case OvS is suitable for our platform because it is the first entry point for all the Virtual Machines (VMs) sending traffic to the network and is the ingress point into overlay networks running on top of physical networks in the data center. Moreover OvS for virtual networking is considered the core element of many datacenter SDN deployments and the main use case is multi-tenant network virtualization. It is highly important to use network virtualization in our platform because it solves a lot of the networking challenges in todays data centers, helping organizations centrally program and provision the network, on-demand, without having to physically touch the underlying infrastructure.

B. SoC-Flow Protocol

In order to design our new communication protocol, named as SoC-Flow protocol, which is based on OpenFlow protocol, some major changes should be considered. However the OF architecture will remain the same since we are going to follow the three basic concepts that we have already mentioned. Moreover our main idea is to launch a more lightweight version of OpenFlow, capable of carrying the packet traffic in our network.

The SoC-Flow Protocol supports a secure communication between multiple PCs and multiple ICs, which is presented in the following section. Moreover, we consider to use harware SDN switches inside on an IC and SDN switches in the boundaries if and IC, which communicate with a centralised controller, hence we will be able to apply forwarding rules for packet switching. For that reason we took into account a new way to address all network entities. Our addressing format is shown at [Fig. 2]. Specifically it consists of two main fields: the Network ID, which is 32-bits and the Entity ID, which is 8-bits. With our approach, we can support 256 different entities in one network. The Network ID field is comprised of the PCB and IC fields, which are 8-bits each and the PC field which are 16-bits. Consequently our system can support 256 different PCB and IC networks and 65.536 different PC networks.

The message layer is the core of the protocol stack. It defines valid structure and semantics for all messages. Every message starts with the same header structure, which is 32-bits long [Fig. 3]. The header message format consists of three main fields. Firstly, the version field indicates the version of communication protocol which this message belongs. Secondly, the length field indicates where this message will end in the byte stream starting from the first byte of the header. Thirdly, the xid, or transaction identifier, is a unique value used to match requests to responses. The type field which indicates what type of message is present and how to interpret the payload, is version dependent and we can see the messages that it is including above. Furthermore every message that travels across our network is consisted of the same header of 32-bits. However the payload size depends on the length field that is provided through header message and it can vary according to the type of the message. Afterwords it includes the source and destination ID following the addressing format that we previously presented. Another field is the type of the packet, for example it could be opcodes for a given processor, following by some padding and the Data.

As far as the messaging layer, we are considering to use less messages than OpenFlow Protocol, in order to have a simple and lightweight protocol but also a simple network that would be able to carry the traffic of our platform. The messages that will be used in our network are shown in the pseudocode above, our protocol contains 23 messages in contrast to the last version of OpenFlow, which contains 35 messages. Furthermore we did not follow the last version of OpenFlow, since it was necessary to include some messages from previous version of OpenFlow according to our platform needs. Consequently, there are three classes of communication in the our protocol: controller-to-switch, asynchronous and symmetric communication. The controller-to-switch communication is responsible for feature detection, configuration, programming the switch and information retrieval. Asynchronous communication is initiated by the switch and it used to inform the controller about packet arrivals, state changes at the switch and errors. Finally, symmetric messages are sent without solicitation from

978-1-5386-5899-4/18/$31.00 ©2018 IEEE

either side, i.e., the switch or the controller are free to initiate the communication without solicitation from the other side.

```
enum coc_type {

    /* Immutable messages. */
    HELLO                   = 0, /* Symmetric msg */
    ERROR                   = 1, /* Symmetric msg */
    OFPT_ECHO_REQUEST       = 2, /* Symmetric msg */
    ECHO_REPLY              = 3, /* Symmetric msg */
    EXPERIMENTER            = 4, /* Symmetric msg */

    /* Switch configuration messages. */
    FEATURES_REQUEST    = 5, /* Controller/switch msg */
    FEATURES_REPLY      = 6, /* Controller/switch msg */
    GET_CONFIG_REQUEST = 7, /* Controller/switch msg */
    GET_CONFIG_REPLY    = 8, /* Controller/switch msg */
    SET_CONFIG          = 9, /* Controller/switch msg */

    /* Asynchronous messages. */
    PACKET_IN           = 10, /* Async msg */
    FLOW_REMOVED        = 11, /* Async msg */
    PORT_STATUS         = 12, /* Async msg */

    /* Controller command messages. */
    PACKET_OUT          = 13, /* Controller/switch msg */
    FLOW_MOD            = 14, /* Controller/switch msg */
    GROUP_MOD           = 15, /* Controller/switch msg */
    PORT_MOD            = 16, /* Controller/switch msg */
    TABLE_MOD           = 17, /* Controller/switch msg */

    /* Queue Configuration messages. */
    QUEUE_GET_CONFIG_REQUEST= 18, /* Controller/switch msg */
    QUEUE_GET_CONFIG_REPLY  = 19, /* Controller/switch msg */

    /* Asynchronous message configuration. */
    GET_ASYNC_REQUEST  = 20, /* Controller/switch msg */
    GET_ASYNC_REPLY    = 21, /* Controller/switch msg */
    SET_ASYNC          = 22, /* Controller/switch msg */

};
```

The first set is symmetric basic messages that every SDN Protocol contains, it follows the switch configuration messages, which are messages between switch and controllers every time that a new switch join the network or when a switch is configured. Afterwards, there are asynchronous messages about the packets, flows and ingress ports. Additionally, the controller command messages contain main commands in order to achieve a successful routing of the packets. Thereafter we have the queue configuration messages, controller role change request messages and asynchronous configuration messages. The queue configuration messages are not included on the last version of OpenFlow but we consider to use them in our communication protocol. Also, the controller role change request messages are introduced by the last versions of OpenFlow and they are a set of messages used by the controller to modify its role among multiple controllers on a switch.

The main routing actions that are available through our network are: add, forward, drop, modify. By applying these actions in our network, we will be able to guide every packet through our SDN network and at the same time we will be able to transfer the packets through hardware entities in an easy and safe way by using our SDN network.

V. Security aspects

As far as the security is concerned, this paper explicitly mentions the registration and authentication of involved entities. Other security issues such as boot security (verification and validation of hardware and software binaries), sharing and protection of memory and, further any side channel attacks are outside the scope of this discussion. Moreover, during the execution of a secure application, the inside communication among PCs needs to be encrypted and is a separate issue to address.

Following the secure booting process, the first step is registration and the authentication of every entity in our network. Moreover, we are planning to have the same approach for every entity that wants to join the existing runtime cluster of processors. In this paper, we have adopted the identity based framework in the structured peer to peer system. The assignment of node IDs is therefore critically important to the efficiency and security of the P2P system. Malicious entities that can control ID assignment can probabilistically or deterministically assert themselves as the source of selected content or routing messages, and can therefore subvert the routing protocols, pollute the delivered content, or prevent placement of legitimate content. Our idea comes from [3], where the authors presented a P2P protocol with identity based setting.

The communication setting in CoC framework is P2P authentication whenever there is a need to expand the runtime cluster size. The following table includes some notations used in registration and authentication process:

TABLE I
NOTATIONS USED FOR THE SECURITY

RE:	registering entity requesting for node ID and private key
NE:	new entity which wants to join the cluster
EE:	existed entities already part of the cluster
$Address_A$:	address of an entity
ID_A:	ID of an entity
$K_{\overline{A}}$:	private key of an entity
$Sign_k(m)$:	signature of the messages m with the key k
$E_k(m)$	encryption of message m with the key k

A. Scenario: Set-up and Registration

In the setup phase, prior to operation, the trusted authority (PKG-Private Key Generator, in identity based settings) takes some input parameter (to decide the security level) and publishes the corresponding system parameters. These system parameters allow the entities to compute the public key from the identifying string (e.g., IP address) of other entities.

The registration of all the entities in ID based cryptosystem, is itself a challenging task. The PKG generates the private keys for all the entities involved (all the processing units, switch, controller) and deliver them securely. In the presented CoC model, the PKG will be a tamper resistant secure core equipped with True Random Number Generator (TRNG), crypto accelerator and secure memory. For the sake of simpler design, we can consider PKG inside the switch.

Recall that, as per the SDN consideration, all the PCs are directly connected to the switch. At the time of bootstrapping, either we have to assume there is no malicious entity

978-1-5386-5899-4/18/$31.00 ©2018 IEEE

installed on the system or a common session key has been already exchanged between the requesting entity and PKG. The common session key can be exchanged using some authenticated Diffie-Hellman key exchange protocol. The PKG generates a master secret key for himself and this master secret key is used to derive the private key for all the requesting entities. All the PCs will send their addresses with some node information, to the PKG and PKG provides them the node identity and corresponding private key. Additionally, the PKG will send a token with a timestamp. This token can be a signature to ensure the authenticity of the PKG. The registration process will be as follows:

1) $RE \rightarrow PKG : Address_{RE}$
2) $PKG \rightarrow RE : ID_{RE}, TS_1, E(K_{\overline{RE}}, K_{PKG.RE}),$ $Sign((ID_{RE} \parallel TS_1), K_{\overline{PKG}})$

Note that the session key $K_{PKG.RE}$ is used to encrypt the communication between PKG and Registering Entity (RE).
Thereafter, to run an application, the operating system will allocate some PCs to run the tasks in parallel. All the PCs will verify the tokens received from PKG as all of them have the public key of PKG. PCs will only be allowed to participate in the cluster if the above token verification is successful.

B. Scenario: When a new PC wants to join the cluster.

In the initial bootstrapping phase, a PC transmits its IP address to a trusted entity PKG and receives the node ID, private key and a token as proof of authentication. For the cluster formation, all the required PCs will verify the tokens of one another. Further, during the application running, some more processors may be required to already running cluster. The addition of new processors is only possible after the token verification which is originally provided by the PKG. More formally, we can say:
Suppose a New Entity (NE) attempts to join the cluster, any of the neighboring Existing Entity (EE) unit, which is already a part of the cluster, can verify the authenticity of NE. The first two steps of the presented protocol are necessary only when private key renewal needs to be performed. The message exchange is as follows:

1) $NE \rightarrow PKG : Address_{NE}$
2) $PKG \rightarrow NE : ID_{NE}, TS_1, E(K_{\overline{NE}}, K_{PKG.NE}),$ $Sign((ID_{NE} \parallel TS_1), K_{\overline{PKG}})$
3) $NE \rightarrow EE : ID_{NE}, TS_1, Sign((ID_{NE} \parallel TS_1), K_{\overline{PKG}})$

VI. Conclusion and Future Work

This paper introduced a new CoC platform, which consists of interconnected ICs and IC cores. The ICs can have different communication speeds and hierarchy levels. Furthermore, we focused on the networking aspects of our platform by presenting our approach using SDN in order to provide cloud-like flexibility but also to have a secure communication inside the platform. At the end, we proposed a security protocol which will be applicable for the registration and authentication of every entity in our network.

From the networking perspective, we plan to focus on the addressing format of every entity in order to have an efficient and lightweight communication between the entities of our network. Furthermore it is important to focus on the design process of our platform before we move to the implementation part. As an initial step we are planing to simulate an NoC-based MPSoC platform with the help of SDN approach. Our main idea is not only to adopt the SDN approach but the creation of a full SDN communication protocol that will be adaptable for the communication inside a NoC but also for the communication of the proposed CoC platform.

Acknowledgment

The research has done in the context of Self-Organising circuits For Interconnected, Secure and Template computing (SOFIST) project, which is supported by Project ARC (Concerted Research Action) of Federation Wallonie-Bruxelles.

References

[1] A Artieri, V Dalto, R Chesson, M Hopkins, and C Marco Rossi. Nomadiktm open multimedia platform for next generation mobile devices, 2003.
[2] Robert E Balfour. Building the internet of everything (ioe) for first responders. In *Systems, Applications and Technology Conference (LISAT), 2015 IEEE Long Island*, pages 1–6. IEEE, 2015.
[3] Kevin RB Butler, Sunam Ryu, Patrick Traynor, and Patrick D McDaniel. Leveraging identity-based cryptography for node id assignment in structured p2p systems. *IEEE Transactions on Parallel and Distributed Systems*, 20(12):1803–1815, 2009.
[4] Liu Cong, Wang Wen, and Wang Zhiying. A configurable, programmable and software-defined network on chip. In *Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on*, pages 813–816. IEEE, 2014.
[5] Peter Cumming. The ti omap? platform approach to soc. In *Winning the SOC Revolution*, pages 97–118. Springer, 2003.
[6] Santanu Dutta, Rune Jensen, and Alf Rieckmann. Viper: A multiprocessor soc for advanced set-top box and digital tv systems. *IEEE Design Test of Computers*, 18(5):21–31, 2001.
[7] Open Networking Foundation. Openflow switch specification version 1.5.0 (protocol version 0x06), 2014.
[8] Justin Helmig. Developing core software technologies for ti?s omap platform. *Texas Instruments*, 2002.
[9] R Sandoval-Arechiga, JL Vazquez-Avila, R Parra-Michel, J Flores-Troncoso, and S Ibarra-Delgado. Shifting the network-on-chip paradigm towards a software defined network architecture. In *Computational Science and Computational Intelligence (CSCI), 2015 International Conference on*, pages 869–870. IEEE, 2015.
[10] Remberto Sandoval-Arechiga, Ramón Parra-Michel, JL Vazquez-Avila, Jorge Flores-Troncoso, and Salvador Ibarra-Delgado. Software defined networks-on-chip for multi/many-core systems: A performance evaluation. In *Architectures for Networking and Communications Systems (ANCS), 2016 ACM/IEEE Symposium on*, pages 129–130. IEEE, 2016.
[11] Amazon Web Services. Amazon ec2 f1 instances, 2017.
[12] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

978-1-5386-5899-4/18/$31.00 ©2018 IEEE