Secure On-Chip Communication Architecture for Reconfigurable Multi-core Systems

Ahmed Saeed[1], Ali Ahmadinia[2]*, Mike Just[3]

[1]School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow, UK

[2]Department of Computer Sciences, California State University San Marcos, USA

[3]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK

Abstract

Security is becoming the primary concern in today's embedded systems. Network-on-Chip (NoC) based communication architectures have emerged as an alternative to shared bus mechanism in multi-core SoC devices and the increasing number and functionality of processing cores has made such systems vulnerable to security attacks. In this paper a secure communication architecture has been presented by designing an identity and address verification (IAV) security module, which is embedded in each router at the communication level. IAV module verifies the identity and address range to be accessed by incoming and outgoing data packets in a NoC-based multi-core shared memory architecture. Our IAV module is implemented on a FPGA device for functional verification and evaluated in terms of its area and power consumption overhead. For FPGA-based systems, the IAV module can be reconfigured at run-time through partial reconfiguration. In addition, a cycle-accurate simulation is carried out to analyse the performance and total network energy consumption overhead for different network configurations. The proposed IAV module has presented reduced area and power consumption overhead when compared with similar existing solutions.

1 Introduction

Security attacks against embedded systems are becoming more serious and sophisticated. New solutions for embedded systems are required since current techniques face several challenges with security requirements and performance constraints of these systems. When the number of processing elements are large and each core has high bandwidth demands, communication between the cores over a single shared bus is impractical, and the use of direct connections (such as grid or mesh communication) becomes necessary [5, 15]. The increase in software content and network connectivity of multi-core System-on-Chip (SoC)

*Corresponding Author: aahmadinia@csusm.edu

makes them vulnerable to fast spreading software-based attacks such as viruses and worms, which were hitherto primarily the concern of personal computers, servers and the internet[24]. The characteristics of FPGA-based reconfigurable devices such as low power consumption and parallel architecture have enabled detection and prevention of security attacks at system level while promising the required energy and computation efficiency without compromising the performance. Furthermore, advantages such as reduced time-to-market as compared to Application Specific Integrated Circuits (ASICs), run-time reconfiguration of the same hardware platform to adapt to different applications, partial reconfiguration support to modify certain part of the hardware, integration with other hardware components such as memory modules and other peripherals have made the use of FPGAs more interesting and favourable for developing new embedded devices.

With the increasing number and functionality of processing cores in multi-core SoC devices, Network-on-Chip (NoC) has emerged as an alternative to shared bus mechanism. Although technology scaling has improved the chip security by increasing the cost and technological requirements of an attacker, however several components like the interconnection, be it a conventional bus-based or a NoC, remain susceptible to attacks[25]. A typical NoC architecture comprises of processing elements (or more commonly known as processing cores), network interfaces, routers and links. A NoC-based system consisting of 16 processing cores and configured in a mesh topology is shown in Fig. 1. Such system

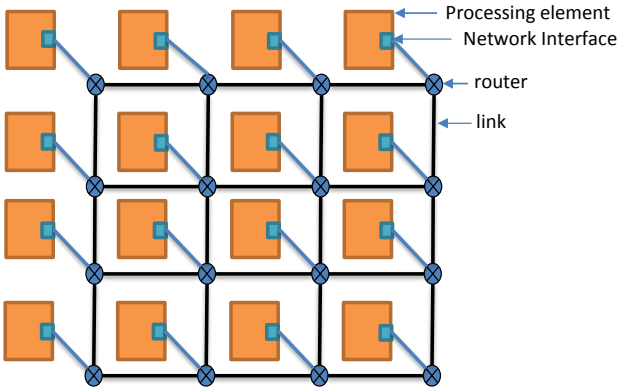


Figure 1: Typical architecture of 4x4 mesh NoC architecture

can be characterised by its topology (the organization of the processing cores and routers) and the approaches used to implement the flow control mechanism, routing algorithms, arbitration and buffering. The flow control deals with data traffic on the channels (physical communication links between routers) and inside the routers. Routing algorithms define the path taken by a message from a source to a destination. The arbitration establishes priority rules when two or more messages request the same communication link. Buffering is used to store messages when a requested output channel is not available. The message-based

(also known as packet-based) communication model is normally used in NoC-based systems. Each processing core is attached to a network interface which is responsible for sending and receiving messages to and from the corresponding router. The packet transmitted by the source processing core is processed by the attached network interface and communication path to the destination is determined based on the routing logic and availability of resources. Depending on the network implementation and flow control mechanism, packet can be split into smaller units named flits (flow control units). The length of a flit is defined at the communication link level and in most cases it corresponds to the minimum number of bits that can be transmitted over one physical channel. Packet-based communication presents a better resource utilization and avoids congestion as packets are short in length and reserve a smaller number of channels during their transfer.

In this paper, a new hardware-assisted security solution has been presented for reconfigurable multi-core SoCs that use NoC as a communication architecture. The proposed solution exploits various NoC features in order to prevent security attacks that target such systems to gain illegal access to trusted areas of the memory. The NoC communication architecture has introduced new weaknesses and made such systems vulnerable to security attacks. NoC architecture has been studied widely from different design perspectives such as network topologies, system performance, routing mechanisms and application-specific implementations. Security aspects related to such systems have emerged recently in the literature. Especially in shared memory architectures for many-core systems the protection of data is a key concern. Once a processing core is compromised, the sensitive application data can be altered by gaining illegitimate access to secured blocks of the memory such as through buffer overflow attacks[6].

A hardware-assisted security solution in the form of Id and Address Verification (IAV) module has been designed and implemented at communication level for FPGA-based multi-core devices. The IAV module targets security attacks by intercepting the transferred packets between processing cores and verifying the identity and memory address before passing them to their destination nodes. In addition to securing against code modification attacks, the designed IAV module has reduced area and power consumption overhead when compared to similar solutions[8, 7] with the same configuration settings. The designed solution is rigorously tested in terms of its functionality (using different configuration settings depending on the router channel width and the number of nodes present in the network) and evaluation (area, power consumption and performance are tested on different network sizes ranging from 16 nodes to 256 nodes).

The paper is organized as follows: Section 2 presents an overview of current work related to the security aspects of shared memory architectures for many-core systems. Section 3 describes the architecture of the proposed security mechanism at the router level. Section 4 defines the hardware implementation used to obtain area and power consumption estimation results followed by a performance evaluation in Section 5. Section 6 concludes this paper.

2 Related Work

Diverse hardware-based security solutions have been proposed in the literature depending on the hardware architecture of embedded devices to handle security attacks. Simple embedded systems are usually comprised of single processor along with other peripherals to execute the program code whereas modern multi-core devices have many processing cores to perform distributed execution of programs.

Hardware security components in embedded systems are typically focused on monitoring processor-executed code by comparing it to a predefined model and the security subsystem operates in parallel with each processor[16]. The hardware-assisted monitors[3, 20, 23] are based on the concept of sensing deviation in program execution at run-time by comparing behaviour against a static model for the purpose of detecting code modification attacks. Arora et al.[3] presented a security mechanism using hash values at basic block level (containing several instructions) along with inter- and intra-procedural control flow graphs. Shufu et al.[20] have also used hashed-based patterns for basic blocks comprising of few instructions. On similar basis, Gebotys et al.[11] have presented security solution for the exchange of cryptographic keys in a NoC-based system. Their solution prevents illegal access to unencrypted keys and provides protection against power and electromagnetic analysis attacks for system containing both un-trusted and trusted processing cores. Only secure IP cores running trusted software are allowed to access the keys. At the network level, symmetric key cryptography is implemented where each trusted core has a dedicated security wrapper where private network key is stored in a non-volatile memory.

Diguet et al.[7] proposed a security mechanism for NoC-based reconfigurable systems which is based on a centralized security and configuration manager (SCM) for safe hardware and communication configurations and attack monitoring. They designed a dedicated secured network interface (SNI) for each processing element to handle different attack scenarios by implementing access control rules at the time of configuration. The address extracted from a packet is compared with a valid address bound stored in the SNI. The SNI configurations and monitoring are performed by the SCM through a separate secured virtual channel (VC) so that the configurations cannot be altered by any malicious code. It is configured for each shared memory region resulting in multiple SNIs for a single memory unit and the area overhead is provided at system level for the particular network configuration only. Area overhead of the SNI is presented in terms of the number of slices required when a complete system consisting of 22 nodes is synthesized with SNIs for a FPGA device whereas energy overhead or power consumed by each SNI is not presented.

Fiorin et al.[9] proposed a Data Protection Unit (DPU) for a NoC architecture to handle unauthorized accesses to selected blocks of memory using address and access rules lookup tables. This DPU is placed inside the network interface of a processing core or memory unit and requires a complete packet to be fetched to start processing. This work is extended[8] by presenting the implementation analysis of DPU in terms of area, energy and performance by placing

it either at the network interface (NI) of the processing cores or in the NI of the memory unit for a network of 10 nodes. The area estimation is achieved by comparing the DPU chip area with the IP core chip area when the complete system is synthesized at fabrication level. They have also presented a comparable approach[10] to that of Diguet et al.[7]. Probes are implemented within the network interface of each processing core and there is a centralized network manager to monitor alert signals generated by each core. A dedicated communication medium separate from the normal data traffic is proposed to handle monitoring traffic towards the network manager.

Lukovic et al.[18] presented a solution to handle buffer overflow attacks by adding a hardware module named processor protection system (PPU) within the network interface of each processor. The Data Protection Unit (DPU)[9] is embedded in NI attached to shared memory unit to filter unauthorized memory access rights. Stack Protection Unit (SPU)[17] is used to copy functions return addresses to provide protection against buffer overflow attacks. Dedicated secure NoC is implemented in parallel to regular NoC connecting security elements embedded in NIs with central Network Security Manager (NSM). The area overhead for each PPS represents 77% of the enhanced NI area. Neither the area overhead for the dedicated NoC, nor the application details and power/energy analysis of the proposed system were presented. This work is further extended by presenting a conceptual approach[19] to secure multi-core NoC-based systems at different levels of design. It is based on implementing security agents in a hierarchical manner stopping the transmission of attacks all the way through the NoC. It is a conceptual solution and experimental setup and synthesis results are the same as they have presented in their previous solution[18]. The complete system is not synthesized as cluster level security agent (CLSA) implementation is avoided to simplify the experimental setup.

Porquet et al. [22] presented a hardware and software based security mechanism to isolate multiple applications and process data securely in a shared memory NoC-based systems. They implemented a hardware module inside the network interface of each processing core and divided the shared memory address space so that each software application is isolated and has its own secure memory region. A trusted software model has also been tested to work along with the hardware protection module. They called each application address space a *compartment*, tagged it with a unique identity and associate access rules with each compartment. At the time of execution, isolation is achieved in the NoC by intercepting the transmitted packet and verifying the information against the set of pre-defined access rights for each compartment.

Ancajas et al.[2] presented a three layer security framework for NoC-based systems where a compromised NoC (C-NoC) provided by a third party can enable a range of security attacks with an accomplice software component. They implemented their security mechanism as a hardware firmware that interfaces processing core with the network interface of the NoC. Their solution is based on scrambling of transmitted data through hardware encryption, attaching encrypted tag with each packet before transmitting it in the NoC and obfuscation of processing nodes by migrating the applications at run-time. The area and

power consumption overhead is measured by implementing proposed security features within the network interface.

Wassel et al.[27] proposed a non-interfering and low-latency approach in order to secure NoC- based systems. The aim of the proposed solution is to provide security by maximizing temporal and spatial separation of communication flows in the NoC. The network traffic is statically separated in the network without sacrificing NoC performance. Static isolation of resources are desired to prevent sharing of information through side-channels. The proposed solution is based on implementing network scheduling in a wave manner which allows multiple traffic flows in the network to be processed in a non-interfering nature while also avoiding the overheads of a cycle-by-cycle time multiplexing. This technique does not detect software-based attacks such as code modification and extraction of secret information through applications running on compromised cores.

The above mentioned security solutions are centralized in nature and provide single layer of protection either at initiator or target node. Moreover, they have only been tested for small network sizes. In contrast, we propose a scalable solution based on id and address lookup tables which is distributed in nature and provides two levels of protection by embedding the IAV module in the local channel's input and output port of the router. At the first level, the IAV module performs an identity and address check at the local input port of the router attached to the transmitting node. At the second level, if the packet contents are modified during its flight time in the event of a security attack, the source identity and address is again verified at the local output port of the router attached to the receiving node. The scalability of the proposed solution is evaluated in terms of area, power and performance overhead for various NoC sizes up to 256 nodes. Performance evaluation results are achieved through trace-driven simulations of real world multithreaded applications as well as synthetic traffic patterns using a cycle accurate simulation framework.

3 ID and Address Verification Module

In this section, the architecture of IAV module is presented, along with the general characteristics of the overall system in which the IAV would operate. The Proposed IAV module is designed for multi-core NoC-based systems having a shared memory architecture where a memory unit can be accessed by several different processing cores. Data coming from the processing cores is processed by the network interface into the required packet format which is used within the NoC. The communication mechanism is based on the transactions of packets between different processing cores and memory units, therefore it is based on the assumption that all the processing cores on the NoC are memory mapped. NoC architectures can be manipulated to get illegal access to the storage unit due to the sharing of memory areas among different processing cores and accessing them via NoC. To verify the memory accesses, the IAV module works in similar fashion as a firewall operates in the network security system by monitoring

the incoming and outgoing data traffic. IAV is a hardware module that applies access control rules to the memory accesses by specifying particular processing cores that can initiate a transaction to a predefined memory blocks in a shared memory NoC. The partitioning of the memory unit into blocks allows the separation between trusted and non-trusted data for the different processing cores attached to the NoC communication architecture.

The communication among different processing cores is controlled through routers, therefore to achieve secure communication, IAV module has been implemented in the local channel's input and output port of the router attached to the network interface of the processing core or memory unit. The block diagram of a standard router is shown in Fig. 2 and it consists of five ports or channels namely east, west, north, south and local port and a central crosspoint switch.

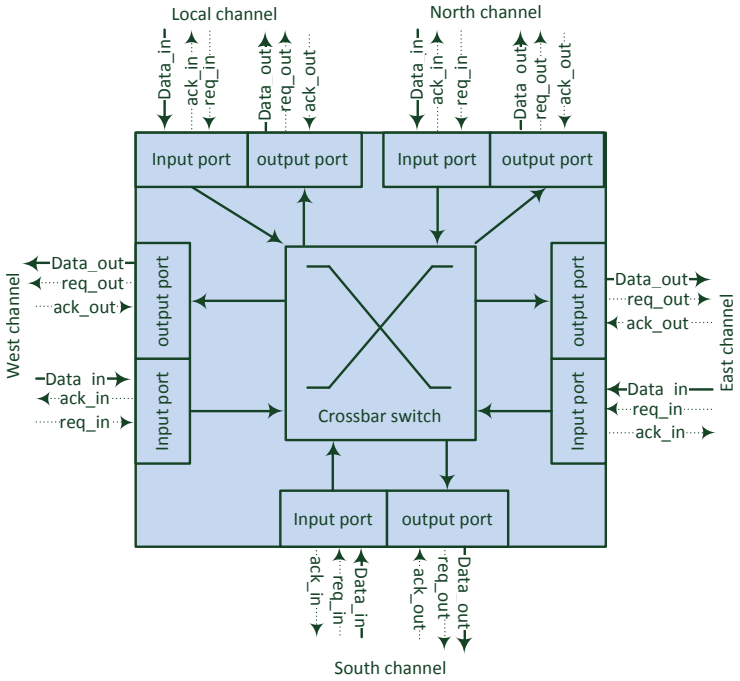


Figure 2: Standard five port router

Each channel has a dedicated input port and output port. Data packet moves in to the input port of one channel of router by which it is forwarded to the output channel of the other port.

Each input and output port has its own decoding logic which increases the performance of the router. Buffers are present at all ports to store the packets. Store and forward mechanism is used as the buffering method. Arbitration decisions are made by the control logic which is implemented inside each input port and output port whereas the crossbar switch forwards the data packet. As per routing algorithm being deployed, the data packet is transmitted from one

of the receiving input ports to the output port.

Based on the destination node, the transmitted packet contains a unique identifier of the destination and address to be accessed in its header flit. Before forwarding the packet to the other routers, the IAV module verifies the identity and memory address by retrieving them from header flit and comparing them with the information stored in its own lookup table. The local port of the router is connected with the network interface while other channels are connected with the neighbouring routers. As the local channel of the router communicates with the processing core, it is the ideal place to embed IAV module in order to filter out unauthorized packets at the time of injection.

The format of the header packet is compatible with OCP-IP interface[21]. For reference, the header packet packet format for 64 node network using 16-nit channel router is shown in Fig. 3. The header flit keeps the identity of the source and destination processing core and the second flit contains the address of the memory block to be accessed at the destination node. The `Id_dest` and `Id_src` fields identify target and initiator of the packet respectively that is assigned directly by network interface at the time of packet generation. The `id_src` field

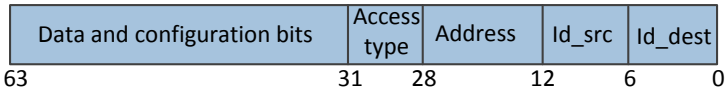


Figure 3: 64-bit header packet format for 64 node network having 16-bit channel routers

can also be configured by OCP-IP interface signals, *MConnID* and *MThreadID*, which provide identifiers associated with processing core and thread respectively. Similarly, *MAddr* signals of the OCP-IP interface can be used by the network interface to generate `Address` field in the header packet.

3.1 IAV Enabled Local Input Port

As mentioned earlier, two level protection is delivered by our proposed solution. At the first level, packets to be transmitted by the processing core are verified by modifying the architecture of the local channel's input port and embedding the IAV module. For reference, the block diagram of the standard input port without IAV module is shown in Fig. 4. A standard input port has three main components i.e., input FIFO, routing logic and control logic. Input FIFO acts as a buffer and stores incoming flits. This buffer has the capacity to store four flits at a time where the size of each flit is either 16 or 32 bits depending on the width of the physical channel. The incoming packet consists of four flits and each input port has a buffer to store packets.

The incoming packet is processed by the input port depending on the FIFO status. The read/write operation to the FIFO is controlled by *control logic*. For instance, if the FIFO is empty, the *control logic* generates *ack_in* signal whenever it receives the *req_in* signal from the output port of its neighbouring router.

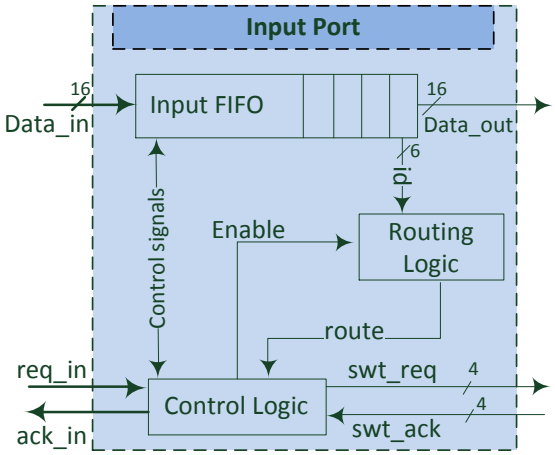


Figure 4: Router standard input port internal architecture without IAV module

When a complete packet is received, the FIFO generates *FIFO_full* signal to the *control logic* which in return disables *ack_in* to stop receiving incoming data packet. The *routing logic* extracts the destination from the first flit of the packet, computes the output port to which the data packet should be transmitted and enables the *route* signal. XY routing algorithm is implemented and a cut-through flow control mechanism is used to handle packet flits. On receiving *route* and *FIFO_full* signals from the *routing logic* and the FIFO respectively, the *control logic* generates *swt_req* signal to the crossbar switch so that the packet can be forwarded to the corresponding output port of the router. On receiving *swt_ack* signal, the *control logic* initiates transfer of packet by sending read signal to the FIFO.

The block diagram of the local input port with an IAV module is shown in Fig. 5. On receiving the first two flits, the FIFO generates *enable* signal and then IAV module reads the identity of the destination node from the first flit and the memory address from the second flit. First 6 bits of header flit represents the identity of the destination node and next 6 bits defines the identity of the source node. The internal architecture of the IAV module for the local channel's input port is shown in Fig. 6. The identities stored in the first column of the lookup table correspond to those destination processing cores that are allowed to communicate whereas the memory bounds in the next two columns correspond to the lower and upper bounds of particular shared memory areas that are accessible by specific processing cores only. Each address entry in the IAV lookup table defines 64 byte memory blocks for 16-bit address. If any processing core tries to access memory region other than the permitted address bounds, it will be considered as a security breach and alert signal will be generated to the *control logic* which stops further processing of the data packet and attack signal can be conveyed to central manager core. In case of successful verification of the id and address, the IAV module generates enable signal for the *routing logic*

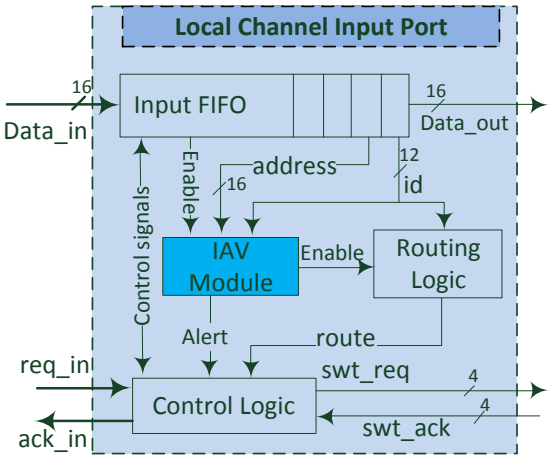


Figure 5: Router local input port internal architecture enabled with IAV module

which computes the output port and generates the *route* signal.

3.2 IAV Enabled Local Output Port

At the second level of our twofold protection, the packets to be received by the processing core are verified by modifying the architecture of the local channel's output port and embedding the IAV module. For a network of 64 nodes and physical channel width of 16 bits, the block diagram of the standard output port without the IAV module is presented in Fig. 7. As shown in this figure, the standard output port consists of output FIFO, arbiter and the control logic. The output FIFO and *control logic* works similarly as described for the input port in the previous section. For output port *routing logic* is replaced with the *arbiter* which is responsible for handling multiple requests coming from the input ports of the other channels of the router. When the output port receives request signals from more than one input ports, the arbiter selects the request based on the rotating priority to each of the input ports. The priority of each port is lowered after it has been served. The *arbiter* receives the *swt_req* signals from the input ports and after making the decision sends the *swt_ack* to the selected input port. The *request* signal is also generated for the *control logic* which asserts the write enable signal for the FIFO.

The block diagram of the local output port with an IAV module is shown in Fig. 8. On receiving the first two flits, the FIFO generates *enable* signal and then IAV module reads the identity of the source and destination nodes from the first flit and the memory address from the second flit. First 6 bits of header flit represents the identity of the destination node and next 6 bits defines the identity of the source node. The internal layout of the IAV module for the local output port is shown in Fig. 9. For example, the ids stored in the first column of the lookup table correspond to those processing cores that

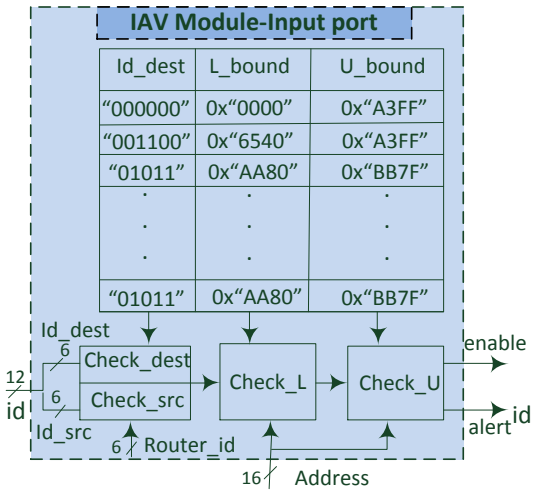


Figure 6: IAV module internal architecture for the local input port

are allowed to communicate whereas the memory bounds correspond to those particular shared memory areas that are accessible by specific processing cores only. If any processing core tries to access memory other than the permitted address bounds, it is considered as a security attack and alert signal is generated for central manager core.

Each address entry in IAV module defines 64 byte memory blocks for 16-bit address and a 4KB memory block for an address size of 32 bits. On receiving the *enable* signal from FIFO, the IAV module extracts the source and destination identities in the form of 12-bit coordinates from the header flit. The 6-bit destination id is compared with the router id and the 6-bit source id is matched with each of the ids stored in IAV lookup table. After verification of the identities, address bits are compared with the corresponding lower and upper bounds stored in the second and third column of the lookup table. If the address is found within the bounds, an enable signal is generated by the IAV module, otherwise an alert signal will be transmitted to the manager core for further action.

3.3 Effectiveness Evaluation

The effectiveness of the IAV module is evaluated based on the fact that in case of software attack (such as thorough buffer overflow), the compromised core can transmit the copy of the packet to an unintended processing core by executing the malicious application. This can only be achieved by altering the header flit and replacing the destination node identity and the IAV module can successfully detect such modifications.

The proposed IAV module has generic architecture and can be embedded inside any multi-core system that uses NoC as a communication architecture.

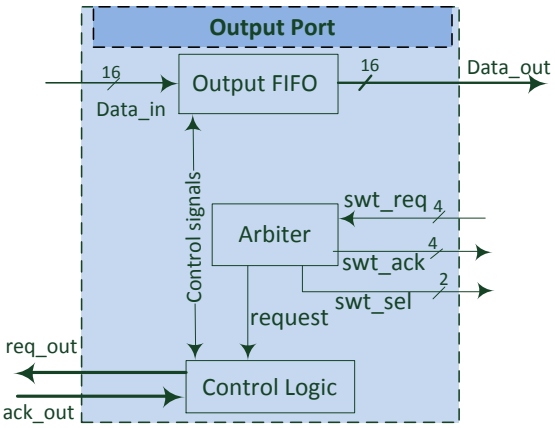


Figure 7: Router output port internal architecture without IAV module

The IAV module will be embedded inside the local channel of each router and it will not require any extra modifications to be made at system level. Here FPGA has been used only as a testing platform. When implemented for FPGA based systems, the IAV module will be configured at design time. It is assumed that the designer will have memory layout of all the applications to be loaded from which the IAV entries can be configured. For FPGA based systems, the IAV module can be reconfigured at run-time through partial reconfiguration by uploading updated bitstream containing new entries for each IAV module. The code modification attacks i.e., buffer overflows will be detected by this module whereas denial of service and information extraction attacks cannot be handled through this proposed technique. The security enhancement has been verified by implementing IAV enabled router for a FPGA device and running behavioural simulations by configuring IAV with various test cases. For invalid test cases, the IAV module has generated alert signal successfully. Depending on the security policy, the manager core on receiving alert signal may disable the compromised core under attack or reboot the system with default configurations. Our solution could also be easily augmented with more granular access control rules (e.g., read only) to provide an extra security layer.

4 Hardware Implementation

In this section, the hardware cost analysis of the IAV module has been presented by evaluating the area and power consumption overhead for different channel widths and number of nodes in the network. The proposed module must have minimum impact on the system area and power consumption in order to have a scalable security solution. The IAV module has been designed to be embedded inside the router without requiring any modifications at the network level. For this purpose as a baseline, the standard five port router architecture, as shown

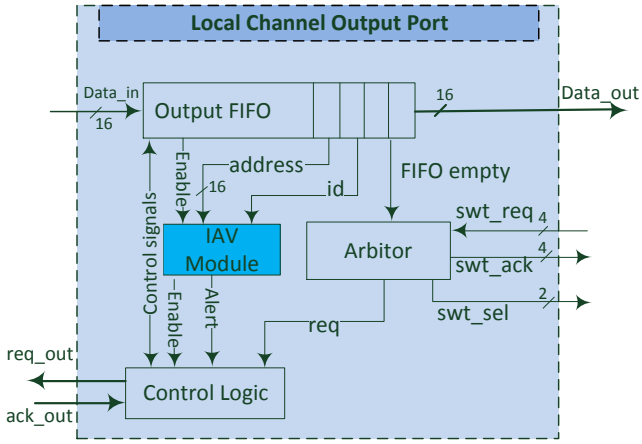


Figure 8: Router local output port internal architecture enabled with IAV module

in Fig. 2) is designed and implemented for a Virtex-7 FPGA device using the register transfer logic (RTL) where target technology is 28 nm.

In order to measure the impact of an IAV module, the router architecture is modified as per requirement and the area and power consumption overhead is evaluated in two phases. In the first phase, a standard router is synthesized without an IAV module and it is configured for cut-through packet switching with XY routing logic to compute the next hop for the packet. The FIFO length is also adjusted to accommodate a four flit packet. After successful implementation and verification of a standard router, in the second phase the IAV module is embedded in the local ports of the router. The router with the IAV module is synthesized for 16 and 32 bit channel routers, with networks consisting of 16, 64 and 256 nodes. Here 16 bit channel router refers to a router where its each input port and output port can receive and transmit a flit of 16 bits at a time respectively.

Synthesis results are generated using Xilinx Synthesis Tool (XST). To obtain area and power consumption overhead results, the DSP and BRAM utilization is disabled and XST is set to use lookup tables and flip flops only during synthesis phase.

4.1 Area Overhead

The IAV is embedded inside the local channel's transmitting and receiving ports of the router providing two level protection but it also increases the area utilization. Area overhead is defined as the increase in the slice utilization when the router is synthesized with the IAV module, as compared to the number of slices utilized by the standard router. The FPGA slice utilization for 16-bit and 32-bit channel router, when configured for different network sizes, is given in

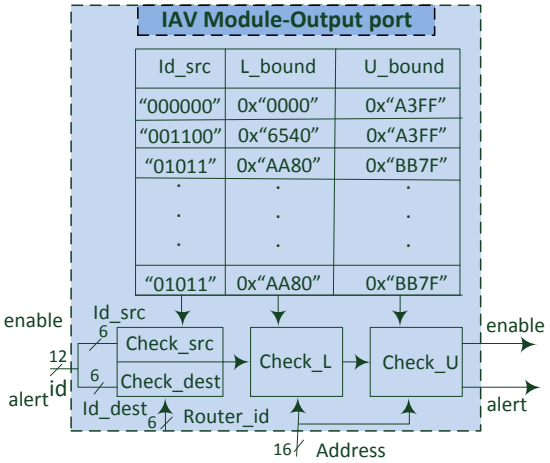


Figure 9: IAV module internal architecture for the local output port

Fig. 10. In our case, the IAV module has been configured with upto 8 entries for 16 nodes network and with maximum of 32 entries for 64 nodes network.

The IAV has been tested with maximum 128 entries for 256 node network and presented area overhead of 83.4% for a 32-bit channel router as shown in Fig. 10. Depending on the security level required and resource constraints, the IAV module can be configured either in input port or output port to reduce the area overhead.

From the results, it is clear that an increase in router channel width has more impact on the area overhead for all the network sizes. This is evident as increasing the channel width results in extra hardware resources for routing and verification of Id and address offset in the IAV module. For example, in our case a 32-bit channel router verifies 20-bit address offset whereas in 16-bit channel router, the address offset of 12 bits is compared respectively which requires less logic to implement. Furthermore, the area increases with the number of entries in the IAV table. As an example, the area overhead for a 16 entries IAV module is 17% as compared to area overhead of 83.4% for a 128 entries IAV module configured in a 256 node network (see Fig. 10).

Table 1 compares the area overhead of our proposed IAV module, when configured inside both local ports of the router, with the existing similar solutions in terms of percentage increase in area utilization for different number of entries in the lookup table. For instance, the SNI based secure system [7] has a 45% area increase when the system is synthesized as a 2-D mesh network of 22 nodes. Each SNI has 2% area overhead per node with a single entry for address verification, whereas the proposed IAV module has resulted in area increase of 1.5% per router when configured in a 64 node network using single entry table for address verification. DPU[8] with 8 entries has area utilization of $0.034mm^2$ while 128 entries DPU occupies $0.581mm^2$ of area. This is a very significant increase in area, whereas the proposed IAV module having 128 entries when configured

with same address lines and channel width as DPU has presented increase of 83.4% as compared to 8 entries IAV which has area overhead of 10%. Moreover, the experimental results presented by Fiorin et al.[8] shows that in a network of 10 nodes, eight DPUs each with 8 entries occupies 9% of the total area (1.125% per DPU) whereas ten routers utilizes 35% of the area (3.5% per router). This means that there is a 24% increase in area for each DPU per router. In our case, a 8 entries IAV module has approximately a 10% area increase compared to a standard router.

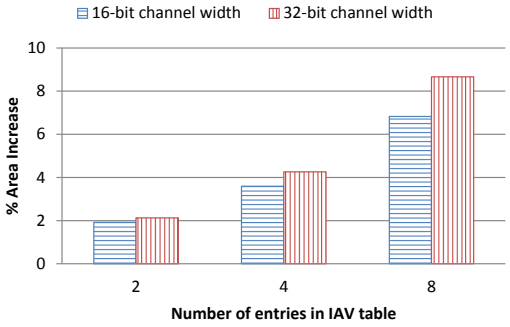
4.2 Power Consumption

Achieving lower power consumption has become one of the main targets in today's FPGA designs as devices have considerably increased in capacity, consuming more power. In our case, the primary interest is in the run-time power consumed by a router when the IAV module is in operation. Here the power consumption overhead is defined as the percentage increase in the dynamic power consumed by an IAV enabled router as compared to a standard router without an IAV module. It is measured using the XPE (Xilinx Power Estimator) tool[28].

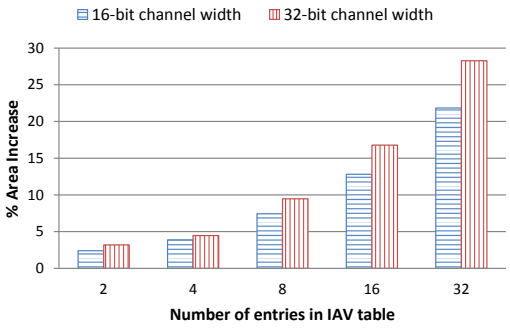
Fig. 11 shows the power consumption overhead results when IAV module is configured inside local input and output ports of the router considering different network sizes and number of entries in the lookup table. The results indicate that the IAV module has less effect on the power consumed as compared to area overhead with similar configurations. To illustrate this, consider a 32-bit channel width router with 16 and 128 entries IAV for 256 node network respectively. The power consumption overhead for the first router is 12.88%, whereas for the second router it is 59.6%. For 16-bit channel router, the 128 entries IAV module has presented 52% increase in power consumption which is lower than 32-bit channel router. This overhead can be reduced further by configuring IAV module inside Local input port of the router only for such systems where low power consumption is one of the main design requirements.

Table 1: Comparison of area and power consumption overhead of the IAV module with existing solutions

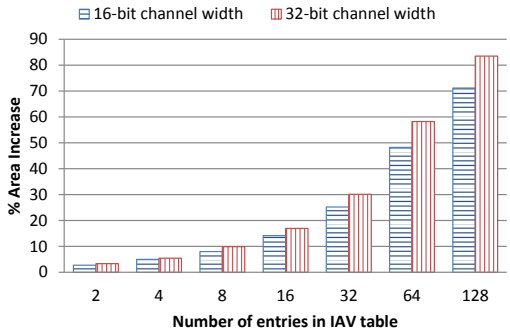
SecuritySolution	Table entries	Area Overhead (%)	Power Consumption (mW)
SNI[7]	1	2	Not provided
Proposed IAV	1	1.5	10
DPU[8]	8	24	18
Proposed IAV	8	10	14
DPU[8]	16	36	30
Proposed IAV	16	17	16



(a) 16 node network

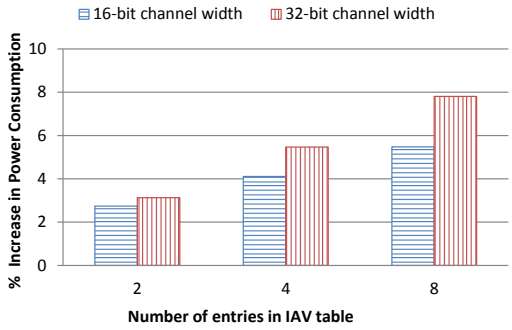


(b) 64 node network

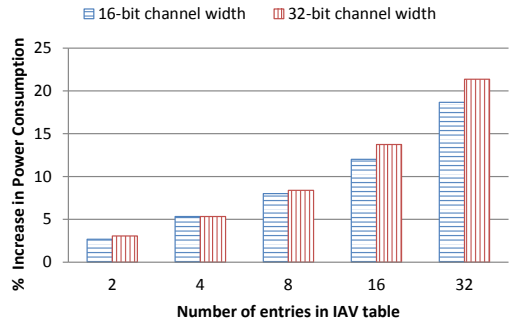


(c) 265 node network

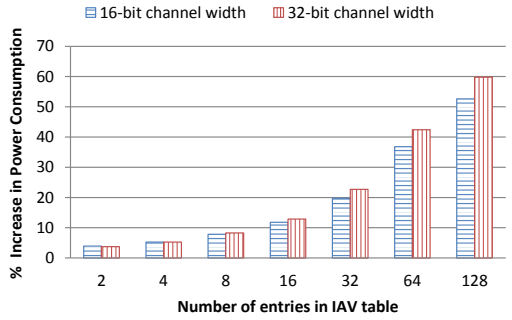
Figure 10: Area overhead as percentage increase for security enhanced router (IAV embedded inside both local ports) as compared to a standard router.



(a) 16 node network



(b) 64 node network



(c) 256 node network

Figure 11: Power consumption overhead as percentage increase for security enhanced router (IAV embedded inside both local ports) as compared to a standard router

In addition, these results are generated by considering the IAV module to be dynamically active after receiving first flit of the packet. In normal mode, the IAV module will be active only when the header of each packet is received and for the rest of the packet flits it will be in idle mode. The proposed IAV module consumes less power as compared to existing solutions. Table 1 summarizes the comparison of power consumption and area overhead of the proposed IAV module with the similar solutions[8, 7]. For example, a 16 entry DPU[8] dissipates $59.9pj$ of energy when synthesized with 500 MHz clock frequency. This is equivalent to $30mW$ of power consumption, whereas a 16 entries IAV module consumes only $16mW$ of power. Similarly, 128 entries DPU dissipates $508pj$ of energy, which is equivalent to $254mW$, while a 128 entries IAV module consumes only $104mW$ in similar configuration. The power consumption of the SNI is not measured in by Diguet et al. [7].

5 Performance and Energy Evaluation

In addition to the area and power consumption overhead, evaluation of the performance and energy consumption overhead is also very important as any modifications in the communication sub-system must have minimum impact on the overall system performance. To measure the performance impact on each router the delay introduced by the IAV module is measured. The total delay is calculated by using Equation 1.

$$Delay(Cycles) = \frac{t_2 - t_1}{t_1} \times total\_clock\_cycles \quad (1)$$

where t_1 and t_2 are the respective times taken by a router without and with an IAV module, and the $total\_clock\_cycles$ is the number of clock cycles taken by the standard router, to complete transmission of four flit packet from one input port to the output port. The synthesis results show that t_2 and t_1 remain the same for an IAV module with up to 8 entries and hence do not introduce any delay in the router. It is also observed that the delay caused by the IAV module is directly proportional to the number of entries in the IAV module lookup table. For instance, the standard router has an operating frequency of 248 MHz when implemented and synthesized without IAV module, while considering the worst case scenario, an IAV enabled router having 128 entries operates at frequency of 156 MHz.

To determine the performance and energy consumption overhead of the proposed IAV module for different network sizes, a cycle accurate interconnection network simulator[14] is used. The synthesized router is configured in this simulator for a mesh network of 64 and 256 nodes with virtual cut-through packet switching and XY routing. The packet size is fixed to four flits. Each simulation is carried out for 200,000 cycles and first 2000 cycles are used as a warm-up period during which the performance is not measured. The results are collected up to the network saturation point where it is no longer possible to inject new messages in the system unless previous messages are cleared out.

Performance overhead is measured considering total packet latency which is defined as the total number of cycles taken by all the injected packets to reach their destination nodes. At first, the packet latency is measured by using the standard router architecture. In the second phase, the router configurations are modified to adjust the delay introduced by IAV module and its impact on the message latency is measured for different number of lookup table entries and number of nodes in the network.

Regarding energy consumption evaluation, the Orion[26] power model is used within the simulation framework. Xilinx Power Estimator tool is used to measure the extra energy being required by the IAV module within the router. To get the total network energy consumption overhead, the simulator framework is extended to incorporate IAV module energy overhead. The network energy is measured by first simulating the network with the standard router and then with the modified IAV enabled router, using different configuration settings. In our case, the packet latency and total network energy consumption are measured for two different types of traffic patterns: synthetic and realistic.

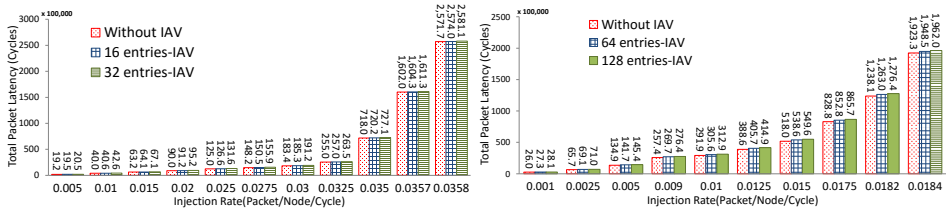
5.1 Synthetic Traffic Patterns

At first, simulations are carried out with three different synthetic traffic patterns: uniform, transpose and hotspot. With a uniform traffic pattern, the source and destination is selected randomly for each generated packet with the equal probability of communication among different nodes. For non-uniform traffic patterns such as transpose, a fixed source-destination pair is generated for each packet (a node with binary value $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $a_{n/2-1}, ..., a_0, a_{n-1}, ..., a_n/2$). For realistic applications, some nodes might receive more packets as compared to other nodes resulting in hotpot nodes in the network. Therefore, the hotspot traffic patterns are generated considering different number of hotspot nodes with higher packet injection rate.

5.1.1 Uniform traffic Pattern

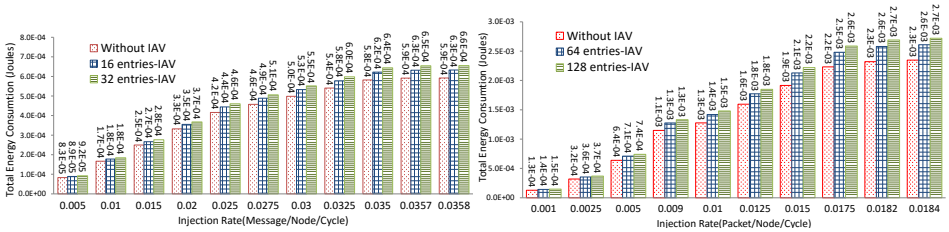
Under uniform traffic pattern the network has presented high inter-node communication density as the packets are generated with equal distribution by all the nodes. The simulation results under uniform traffic pattern are presented in Figs. 12 and 13.

Network performance is more affected when 64 or 128 entries are used in the IAV module. For instance, in a 256 node network, the delay in packet latency in the worst case scenario is increased by 8.12% for a 128 entries IAV module whereas a 64 entries IAV module resulted in only 5.28% increase as shown in Figs. 12(b) and 18(a). This impact can be reduced by placing more frequently communicating nodes in a neighbouring area. To further bring down message latency for a large number of nodes such as in 256 node mesh network, the IAV module can be embedded in selective routers, such as those requiring a higher level of security. It can be also observed from the energy consumption results, as shown in Fig. 13, that the proposed IAV module has a negligible overhead in



(a) 64 node network (b) 256 node network

Figure 12: Packet latency under uniform traffic pattern



(a) 64 node network (b) 256 node network

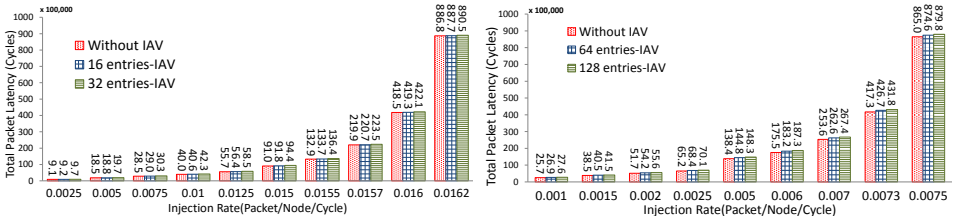
Figure 13: Total network energy consumption under uniform traffic pattern

terms of network energy consumption. For instance, the IAV module having 64 entries has increased network energy consumption by 11.13% in the worst case scenario whereas the 128 entries IAV module has incurred 15.9% increase when simulated for 256 node network. Similarly for 64 node network, the 16 entries IAV module resulted in 6.85% increase in energy consumption as compared to 10.65% for 32 entries IAV module as shown in Figs. 13(a) and 18(b).

5.1.2 Transpose Traffic Pattern

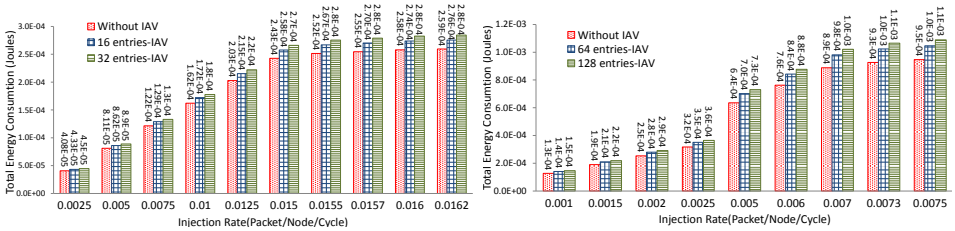
In non-uniform traffic patterns such as transpose, more packets are being generated for communication between remote nodes which result in network saturation with lower packet injection rate as compared to uniform traffic pattern. Furthermore, as the source to destination hop-count varies in non-uniform manner which results in reduced performance overhead as compared to uniform traffic pattern. As it can be seen in Fig. 14(b), the packet latency for 64 and 128 entries IAV module has increased up to 5.02% and 7.71% respectively in the worst case scenario.

The network energy consumption overhead results are presented in Fig. 15. For 64 node network, the energy consumption overhead is increased up to 6.19% when 16 entries IAV module is embedded inside each router whereas 32 entries IAV module has resulted in 9.63% increase in total network energy consumption. Furthermore, as presented in Fig. 15(b), routers enabled with 64 and 128 entries IAV module has incurred 10.54% and 15.06% increase in energy consumption respectively. These results clearly show that the proposed security mechanism is a scalable solution for larger number of nodes in the network.



(a) 64 node network (b) 256 node network

Figure 14: Packet latency under transpose traffic pattern



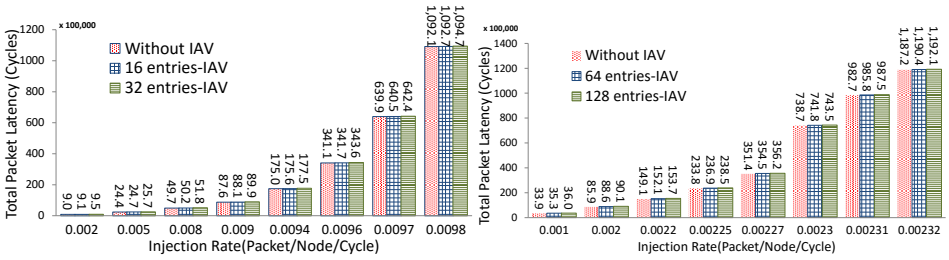
(a) 64 node network (b) 256 node network

Figure 15: Total network energy consumption under transpose traffic pattern

5.1.3 Hotspot Traffic Pattern

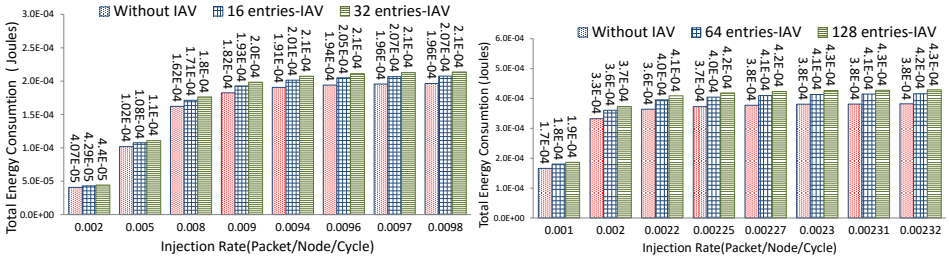
Different number of nodes are selected as hotspot nodes to receive two times more packets as compared to other nodes in the network. Simulations are carried out for three different scenarios by selecting centre node in the middle row, three closely placed nodes in the middle rows and four distributed nodes as hotspots respectively. For example, Figs. 16 and 17 illustrate detailed simulation results when four nodes are selected as hotspot nodes that are separated from each other with minimum hop-count of six and fourteen nodes for the network size of 64 and 256 nodes respectively.

The IAV module has presented almost similar overhead in terms of packet latency as compared to other synthetic traffic patterns. For instance, under single hotspot node, 16 and 32 entries IAV modules configured in 64 node network, have presented packet latency overhead of 1.73% and 7.14% respectively as illustrated in Fig. 18(a). Moreover, 64 entries IAV module for 256 node network has resulted in 4.2% increase in packet latency whereas 128 entries IAV module has presented an increase of 8.42%. As shown in Fig. 18(b), the network energy consumption is least affected when hotspot nodes are distributed with more hop-count as compared to closely placed hotspot nodes. For instance, in case of distributed four hotspots, the 128 entries IAV module has incurred 12.27% increase in energy consumption as compared to 18.31% increase for closely placed three hotspots. Similarly, for packet latency overhead, as illustrated in Fig. 18(a), where distributed hotspots presented 6.22% increase as compared to 8.76% when 128 entries IAV module is used for three closely



(a) 64 node network (b) 256 node network

Figure 16: Packet latency under hotspot traffic pattern with four distributed hotspots.



(a) 64 node network (b) 256 node network

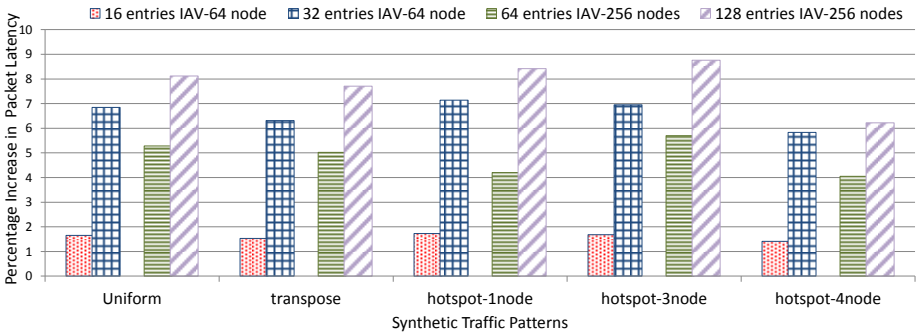
Figure 17: Total network energy consumption under hotspot traffic pattern with four distributed hotspots.

placed hotspots with hop-count of two.

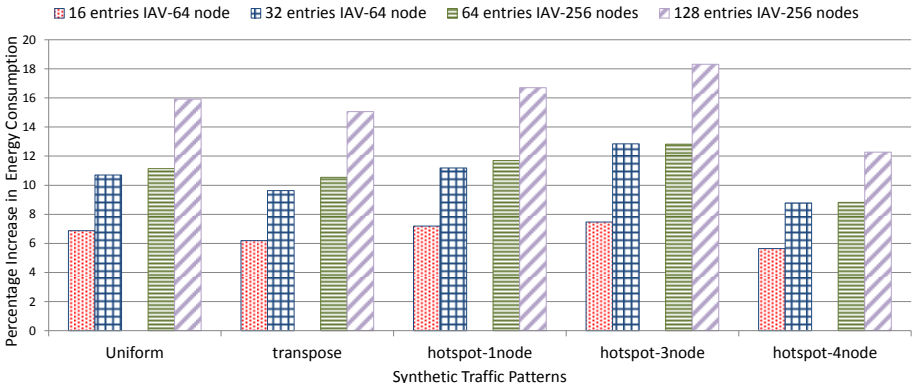
5.2 Real World Applications Traffic

To further check the scalability of proposed solution, the impact on performance is also evaluated through trace-driven simulations of multithreaded real world applications. For this purpose, applications from the PARSEC benchmark suite[4] have been used. The traffic patterns for these benchmark applications are generated using the Netrace[12, 13] tool which has been designed specifically to produce traffic patterns by modelling the multithreaded application communication traffic traces and incorporating the dependences between network packets for a 64-core system.

The traffic patterns for different benchmark applications are obtained in a specific format required for our simulator[1], and simulations are carried out with the IAV having distinct number of entries in a 64 node network environment. The impact on average packet latency is shown in Fig. 19. For real world benchmark applications the network performance is not much affected even when a 128 entries IAV is placed inside each router in a 64 node network, whereas the existing solutions have been tested with a reduced number of nodes



(a) Percentage increase in packet latency.



(b) Percentage increase in total newtork energy consumption

Figure 18: Maximum percentage increase in packet latency and total network energy consumption under different synthetic traffic patterns.

only. For example, the DPU[9] has only been tested for a network of ten nodes claiming no effect on the network performance while SNI[7] is tested for a 4×3 mesh network only with zero effect on the network performance. The proposed IAV module also has zero latency when configured with 16 entries for a network sixteen nodes.

6 Conclusion

In this paper, a scalable and secure on-chip communication mechanism has been presented for shared memory systems using NoC as a communication architecture. The proposed IAV module retrieves and verifies the identity and address of each packet being generated by the processing cores. Security is a prime concern in a shared memory systems and the proposed IAV module provides security layer through identity and address verification for each memory access by embedding itself inside the local channel of each router. The experimental

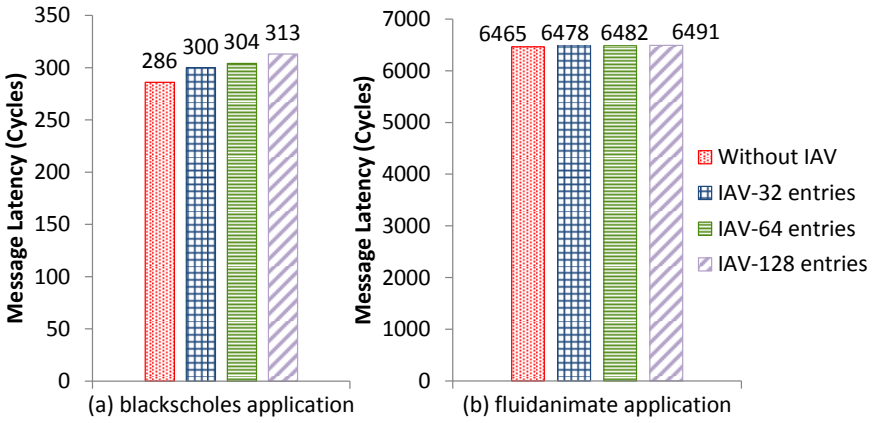


Figure 19: Average packet latency for 64 node network using multithreaded applications from PARSEC benchmark suite

results show the area and power consumption overhead for FPGA-based systems considering different network sizes ranging from 16 to 256 nodes, which are improved compared to the existing solutions. Furthermore, the comprehensive simulation results have been obtained to show the impact of this approach on the network performance and energy consumption. In simulations of real world applications, the proposed architecture has presented lower impact on the network performance.

References

[1] P. Abad, P. Prieto, L.G. Menezo, A. Colaso, V. Puente, and J.-A. Gregorio. Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 99–106, 2012.

[2] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. Fortnocs: Mitigating the threat of a compromised noc. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 158:1–158:6, New York, NY, USA, 2014. ACM.

[3] D. Arora, S. Ravi, A. Raghunathan, and N.K. Jha. Secure embedded processing through hardware-assisted run-time monitoring. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 178–183 Vol. 1, 2005.

[4] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel*

Architectures and Compilation Techniques, PACT '08, pages 72–81, New York, NY, USA, 2008. ACM.

[5] C. Bobda and A. Ahmadinia. Dynamic interconnection of reconfigurable modules on reconfigurable devices. *Design Test of Computers, IEEE*, 22(5):443–451, 2005.

[6] Eric Chien and Péter Ször. Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses. *Virus*, 1, 2002.

[7] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin. Noc-centric security of reconfigurable soc. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 223–232, 2007.

[8] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano. Secure memory accesses on networks-on-chip. *Computers, IEEE Transactions on*, 57(9):1216–1229, 2008.

[9] Leandro Fiorin, Gianluca Palermo, Slobodan Lukovic, and Cristina Silvano. A data protection unit for noc-based architectures. In *Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '07, pages 167–172, New York, NY, USA, 2007. ACM.

[10] Leandro Fiorin, Gianluca Palermo, and Cristina Silvano. A security monitoring service for nocs. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '08, pages 197–202, New York, NY, USA, 2008. ACM.

[11] C. H. Gebotys and Y. Zhang. Security wrappers and power analysis for soc technologies. In *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '03, pages 162–167, New York, NY, USA, 2003. ACM.

[12] Joel Hestness, Boris Grot, and Stephen W. Keckler. Netrace: Dependency-driven trace-based network-on-chip simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, NoCArc '10, pages 31–36, New York, NY, USA, 2010. ACM.

[13] Joel Hestness and Stephen W Keckler. Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation. Technical Report TR-10-11, The University of Texas at Austin, Department of Computer Scienc, May 2011.

[14] Jingcao Hu and R. Marculescu. Energy- and performance-aware mapping for regular noc architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(4):551–562, April 2005.

[15] T. Huffmire, B. Brotherton, T. Sherwood, R. Kastner, T. Levin, T.D. Nguyen, and C. Irvine. Managing security in fpga-based embedded systems. *Design Test of Computers, IEEE*, 25(6):590–598, 2008.

[16] Georgios Kornaros and Dionisios Pnevmatikatos. A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Trans. Des. Autom. Electron. Syst.*, 18(2):17:1–17:38, April 2013.

[17] S. Lukovic, P. Pezzino, and L. Fiorin. Stack protection unit as a step towards securing mpsocs. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–4, 2010.

[18] Slobodan Lukovic and Nikolaos Christianos. Enhancing network-on-chip components to support security of processing elements. In *Proceedings of the 5th Workshop on Embedded Systems Security*, WESS '10, pages 12:1–12:9, New York, NY, USA, 2010. ACM.

[19] Slobodan Lukovic and Nikolaos Christianos. Hierarchical multi-agent protection system for NoC based MPSoCs. In *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems*, S&D4RCES '10, pages 6:1–6:7, New York, NY, USA, 2010. ACM.

[20] Shufu Mao and T. Wolf. Hardware support for secure processing in embedded systems. *Computers, IEEE Transactions on*, 59(6):847–854, 2010.

[21] Open Core Protocol 3.0 Specification, 2013. `http://accellera.org/downloads/standards/ocp/`.

[22] J. Porquet, A. Greiner, and C. Schwarz. Noc-mpu: A secure architecture for flexible co-hosting on shared memory mpsocs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–4, March 2011.

[23] M. Rahmatian, H. Kooti, I.G. Harris, and E. Bozorgzadeh. Hardware-assisted detection of malicious software in embedded systems. *Embedded Systems Letters, IEEE*, 4(4):94–97, 2012.

[24] Patrick Schaumont and Anand Raghunathan. Guest Editors' Introduction: Security and Trust in Embedded-Systems Design. *Design Test of Computers, IEEE*, 24(6):518–520, 2007.

[25] Radu Stefan and Kees Goossens. Noc security using multipath routing. In *20th Workshop on Circuits, Systems and Signal Processing (ProRISC 2009)*, pages 522–525. Citeseer, 2009.

[26] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305, 2002.

[27] H.M.G. Wassel, Ying Gao, J.K. Oberg, T. Huffmire, R. Kastner, F.T. Chong, and T. Sherwood. Networks on chip with provable security properties. *Micro, IEEE*, 34(3):57–68, May 2014.

[28] Xilinx: Power Efficiency, 2013. `http://www.xilinx.com/content/xilinx/en/products/technology/power/`.