Sentry-NoC: A Statically-Scheduled NoC for Secure SoCs

Ahmed Shalaby
ahmed.shalaby@nus.edu.sg
National University of Singapore
Singapore

Yaswanth Tavva
yaswanth@u.nus.edu
National University of Singapore
Singapore

Trevor E. Carlson
tcarlson@comp.nus.edu.sg
National University of Singapore
Singapore

Li-Shiuan Peh
peh@nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

SoC security has become essential with devices now pervasive in critical infrastructure in homes and businesses. Today's embedded SoCs are becoming increasingly high-performance and complex, comprising multiple cores, accelerators, and IP blocks interconnected with a Network-on-Chip (NoC). As these IPs can originate from diverse sources, they cannot be trusted to form the root of trust in SoCs. However, the NoC itself, being the communication backbone linking all IPs, is naturally positioned to be the basis for a secure SoC. Therefore, there is a need for an efficient solution that both meets the stringent requirements of modern embedded SoC designs, while maintaining a high level of security.

In this paper, we demonstrate how statically-scheduled NoCs inherently enforce traffic isolation and non-interference of communication. The time-division multiplexing (TDM) of NoC links across applications *provably* ensures that security properties are fulfilled. However, conventional TDM NoCs are still vulnerable to side-channel attacks. We thus propose temporal and data obfuscation schemes that can be embedded within static TDM NoCs, randomizing source-destination communication patterns and switching activity over the links. Our proposed statically-scheduled *Sentry-NoC* links up untrusted IP blocks to form a secure SoC. *Sentry-NoC* targets key security properties to effectively mitigate side-channel attacks with an extremely low overhead, reducing average temporal correlation by 81% and average data correlation by 91%.

ACM Reference Format:
Ahmed Shalaby, Yaswanth Tavva, Trevor E. Carlson, and Li-Shiuan Peh. 2021. Sentry-NoC: A Statically-Scheduled NoC for Secure SoCs. In *International Symposium on Networks-on-Chip (NOCS '21), October 14–15, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3479876.3481595

1 INTRODUCTION

More than 11 billion embedded SoCs are in use today and this number is expected to grow to 25 billion by 2025 [7]. The explosive growth of embedded SoCs and their potential for vulnerabilities make them a prime target for attacks. Life-threatening attacks on critical infrastructure have been reported [8, 25]; one such example, a power-grid control system hack, resulted in a power blackout for more than 200k customers [5]. Attacks have also been deployed against consumer embedded SoCs devices, with the recent identification of vulnerabilities in 50 commercial devices [4].

Embedded SoCs are pervasive and readily accessible. Physical and side-channel attacks have been deployed, where attackers can extract the secret information using physical parameters of the IC like power, time or electromagnetic emission. Yet, few embedded chips are secure by design [8]. Therefore, hardware security needs to be a fundamental design requirement alongside performance, power, cost, and usability specifications [20].

As ultra-low power consumption is a necessity for embedded SoCs, ultra-lightweight security mechanisms are necessary too. Nevertheless, embedded SoCs are becoming increasingly complex and high-performance, built with multiple cores, accelerators, and IP blocks, often interconnected with a NoC [1]. To accelerate the development process, and reduce the time-to-market, SoC designers can look to third-party IP vendors for critical system infrastructure. But, these IPs can be faulty or purposely misconfigured to undermine the security of the entire SoC, making it vulnerable to malicious exploits. The NoC is well-positioned to form the trusted backbone to connect other untrusted IPs. Hence, there is an essential need to develop a secure, low-overhead NoC that can maintain the SoC integrity in the presence of malicious IPs.

NoCs are broadly categorized into two flavors: dynamically and statically scheduled. Dynamic NoCs handle the on-chip traffic at run-time, aim to deliver high bandwidth at the expense of additional area and power consumption. Static NoCs, instead, manage the traffic based on a pre-defined schedule that guarantees real-time application constraints. Prior works have shown that theoretically-ideal latency and throughput can be achieved with static NoCs when prior knowledge of application traffic patterns is known, which often occurs in our target domain of embedded SoCs [26, 28].

Static NoCs are ultra light-weight, with no buffering, flow control, or routing logic within the NoC routers. Furthermore, static NoCs inherently impose traffic isolation and non-interference during communication; unauthorized access and denial of service attacks cannot occur. Therefore, they represent an ideal candidate to secure SoCs within embedded devices. However, all prior secure designs establish security mechanisms onto dynamic NoCs, ***adding massive overheads in terms of performance and power***

NOCS '21, October 14–15, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9083-5/21/10…$15.00
https://doi.org/10.1145/3479876.3481595

consumption. Instead, we propose building security mechanisms into static NoCs. ***To the best of our knowledge, no previous work has explored using statically-scheduled NoCs as the basis for a secure SoC***. Below, we list the key contributions of this work.

- We propose the use of statically-scheduled NoCs to secure SoCs, as they inherently ensure security properties, while preserving performance at extremely low overheads. We analyze the static NoCs' security properties, as all prior secure NoC designs were dynamic (Section 2).
- We propose Sentry-NoC, a secure statically-scheduled NoC that applies temporal and data obfuscation schemes to mitigate side-channel attacks (Section 3).
- Our detailed evaluation, using RTL and circuit-level simulations, shows that Sentry-NoC effectively achieves both temporal and data obfuscation. Sentry-NoC enables key security properties with an extremely low overhead, demonstrating an area reduction up to 98× compared to a RISC-Core security monitor [24] (Section 4).

2 MOTIVATING STATICALLY SCHEDULED SECURE NOCS

In this section, we introduce our threat model, clarify our assumptions, and survey how prior secure designs introduced security into dynamic NoCs. Next, we demonstrate how statically-scheduled NoCs can improve SoC security and how they are a good candidate for next-generation high-performance embedded and IoT devices.

2.1 Threat Model

SoCs within embedded devices integrate multiple IPs. But, the aggressive time-to-market demands make security verification of all IPs infeasible. For instance, a third-party IP might have been modified to include a hardware Trojan or execute malicious software. Therefore, our threat model considers all IPs untrusted. The Trusted Computing Base (TCB) consists of the hardware and the software components that work together to ensure a set of security guarantees, as specified by the architecture. ***Our TCB is comprised of the NoC and light-weight secure engine.*** The secure engine enforces security policies, analyzes the NoC traffic, and updates the security state of the overall system once a violation has occurred [24]. Secure engines have been widely adopted to secure SoCs in industry products [2] and academic research [14, 22]. The underlying threats include the following attacks:

- Confidentiality attacks aim to extract secret information, like the encryption key, through reading, writing, or observing the content of the confidential information while it is communicated through the NoC.
- Integrity attacks aim to corrupt the packet content or alter the NoC's components' functionality. Data corruption and function alteration can lead to performance degradation.
- Availability attacks aim to exhaust the resources, degrade the performance and consume additional power. Example attacks in this category include: Reply, Flood, Incorrect path, Deadlock attack, Livelock, and Bandwidth-denial.

It should be highlighted that ***our threat model is limited to secure SoC communications,*** where attacks are visible to the NoC. The threat model does not consider the security requirements of operations within each IP, such as restricted read-access to defined memory segments. ***Attacks based on IP access control constraints are out of the scope of this work***, as each IP has distinct features that result in a different set of security requirements.

2.2 Secure Dynamic NoCs

Confidentiality, integrity, and availability are the three cornerstone security properties needed for secure systems [20]. Here, we introduce each of these properties as adopted by the state-of-the-art dynamic NoCs.

2.2.1 ***Confidentiality.*** Preventing untrusted users from reading, writing, or observing the content of information to be protected is termed confidentiality. Encryption is one method commonly used to enable confidentiality and to guarantee information security during NoC communication. *Stream* ciphers have been proposed to secure dynamic NoCs, like XOR [9] and the Linear Feedback Shift Register (LFSR). XOR and LFSR are both low-overhead in terms of power consumption and latency, but they come at the expense of vulnerabilities [20]. On the other hand, *Block* ciphers, like the popular Advanced Encryption Standard (AES) [18], provide a higher level of security at the expense of additional complexity, latency and power consumption. To use these ciphers within dynamic NoCs, the cipher has to be implemented at each Network Interface (NIC), to encrypt each packet as they are injected at run time, which translates to high area, power and performance overheads.

2.2.2 ***Confidentiality—Side Channel Resilience.*** Confidentiality ensures data protection. Side-channel resilience supports non-observability by obscuring private data. Prior works have investigated ways to mitigate side-channel attacks such as timing, power consumption, and electromagnetic emissions. Node obfuscation [9] increases resilience against side-channel attacks through routine migration of the running applications between nodes to induce a temporal variation in their power consumption. Within the NoC, packet route randomization [15] has been proposed to increase the NoC's resilience against timing attacks. Over the link, data obfuscation was proposed to prevent hardware Trojan triggering with logic added to routers to obfuscate each flit by shuffling, inverting, and scrambling the bits [10].

2.2.3 ***Integrity.*** Packet integrity is to verify that that packet contents have not been altered. The packet tag, firewall, and security zone approaches have been proposed to track the packets and report any violation that occurs. Using a security tag is the most common approach, where a tag is attached to the packet tail. In [9], a tag of a 16-bit unique identifier is dynamically generated for each node. More complex tagging methods have been proposed to detect modifications in the payload and in the headers. In [11], this tag is calculated by Cyclic Redundancy Check (CRC) and Algebraic Manipulation Detection (AMD) schemes. In [18], Galois Hash (GHASH) function is utilized to provide the authentication tag. Adding such tags consume a substantial portion of the communication bandwidth and are costly in terms of area, power and latency.

2.2.4 ***Availability.*** Availability ensures the reliability and trustworthiness of providing services. Secure dynamic NoCs guarantee

the prevention of denial-of-service (DoS) attacks such as deadlock, livelock, flooding, reply, and bandwidth attacks by monitoring traffic at runtime, adding counters, traffic limiters, and probe units into NICs and routers. In [18], a traffic counter is implemented at the NIC, and after a pre-defined threshold (MAX-number of packets are sent), the data transmission is halted for a configurable number of cycles. In [30], the NoC is partitioned into different domains, and communication is scheduled between the different domains to achieve timing non-interference. In [12], a probe unit observes the NoC signals, monitoring throughput, latency, resources' utilization, message characteristics, and packets' statistics.

Summary. Prior security solutions built upon dynamic NoCs need to handle their traits: run-time multiplexing and resources sharing across multiple packets. ***Such dynamism leads to solutions that come at the expense of higher complexity, power consumption, and performance.*** Next, we introduce statically-scheduled NoCs and motivate their use as the basis for secure NoCs.

2.3 Statically-Scheduled NoCs

In a statically-scheduled NoC, unlike a dynamically-routed NoC, the links are allocated by a predetermined schedule that guarantees packets reach the correct destinations. A statically-scheduled NoC ensures contention-free routing and flow control, and removes the need for runtime routing and arbitration. Thus, router microarchitecture is simplified to a crossbar switch with no need for buffers, flow control nor routing logic. Buffers can contribute to more than 55% of the dynamic power and 65% of the area [17]. Noticeably, a statically-scheduled NoC is inherently ultra light-weight compared to dynamically-routed NoC. But it comes at the expense of the performance in terms of packet latency and bandwidth, since a packet needs to wait for its allocated time slot before it can move according to Time Division Multiplexing (TDM) schedule.

The TDM schedule bounds statically-scheduled NoC performance. For instance, the maximum throughput is 0.2 flit/cycle/node for TDM NoCs (mesh-64 nodes) [13]. In general, for a k-ary n-mesh network with uniform random traffic, throughput is $1/(k^n$-1), while the theoretical throughput is $k/4$ for the ideal dynamic NoC [17]. However, fully utilized schedules have been proposed for application-based solutions [26] and near-optimal schedules for traffic-based solutions [28], where statically-scheduled NoCs can match the latency and bandwidth of dynamically-routed NoCs.

As our target is embedded SoCs, where security, area, and power overheads are more critical than bandwidth requirements, we see statically-scheduled NoCs as an attractive basis for secure SoCs as long as the device can keep up with real-time computation requirements. The essence of our proposal is that statically-scheduled NoCs start with secure designs at conception. It inherently supports confidentiality, integrity, and availability by creating end-to-end isolated non-interference channels, while avoiding overhead in terms of buffering, flow-control, and arbitration. To be more concrete, we detail how TDM NoCs ensure various security properties below.

2.3.1 ***Confidentiality.*** Statically-scheduled NoCs inherently support confidentiality. Read and write operations are only permissible at predefined time-slots when the specific IP can access the NoC, so untrusted users cannot see the messages passed by other sources and destinations. There is no need to even encrypt the packets.

Proof sketch: A TDM schedule $\boldsymbol{S}(\boldsymbol{R};\boldsymbol{T})$ is defined as a set of routes $\boldsymbol{R}\{r_1, r_2, ..., r_m\}$ assigned to a set of time slots $\boldsymbol{T}\{s_1, s_2, ..., s_t\}$ in a specific order with period t. Such that $\forall(sr,dt)$, a packet can traverse the network from a source sr to a destination dt using one specific route $r_i \in$ R at one particular time slot $s_j \in$ T.

Assume a node n_c can access a packet as it moves from n_x to n_y ($n_x \neq n_y$) in a given route r_i and a time slot s_j. By definition of TDM, the set $\{n_x,n_y,n_c\}$ has exactly two elements, so $n_c \in \{sr, dt\}$: n_c is either the source sr or destination dt. So no malicious node can read or write to a route $r_i \in$ R while another pair (sr, dt) is occupying the route r_i.

2.3.2 ***Integrity.*** Similarly, a statically-scheduled NoC ensures no malicious IP can modify the packets of another IP. Attacks like data corruption and message alteration are blocked as the message can only be accessed by its source and destination.

Proof sketch: Similar to confidentiality, a TDM schedule authorizes a node to access route $r_i \in$ R at one particular time slot $s_j \in$ T. If a packet moves from n_x to n_y, route r_i is confined for (n_x, n_y) communication at time slot s_j, hence no malicious node can access the routers/links along route r_i during this timeslot s_j, and consequently cannot modify the packet as it travels on the route r_i.

2.3.3 ***Availability.*** Availability guarantees the prevention of denial-of-service attacks. The statically-scheduled NoC inherently prevents DoS attacks as the TDM manages the packets' injection based on predefined slots. Hence, no detection is necessary.

Proof sketch: For flooding, reply, and bandwidth DoS attacks, the malicious application tries to exhaust the NoC resources and deplete the bandwidth to prevent other applications from communicating. The probability of depleting the bandwidth is proportional to the packet injection rate *ir*. $ir = 1/c$, where c is the time slots between two consecutive packets.

Assume a malicious application attempts a DoS attack, it works to magnify $ir \gg$. However, the TDM schedule limits the injection rate ir to $1/n$ for traffic-based NoC (all-to-all) or w_i/t for application-based NoC, where n is the number of nodes, w_i is the weight of application a_i in terms of time slots, and t is the schedule period. In both cases, the injection rate is predefined and bounded, so the malicious application cannot increase ir to exhaust the NoC resources. Hence, flooding, reply, and bandwidth DoS attacks cannot succeed.

Caveat: Confidentiality—Side Channel Resilience.
Statically-scheduled NoCs inherently ensure security properties, impose traffic isolation and non-interference communication, as proved. Besides, timing side-channel attacks are foiled by static NoCs as there is no leakage of timing information due to fixed schedules that enforce isolation. But, it comes with a caveat – ***the static behavior of TDM makes static NoCs predictable and readily vulnerable to observation and side-channel attacks.*** Side channel attacks such as power and electromagnetic probing can be particularly effective with predictable static NoCs. To close this gap of side channel resilience, we need to obfuscate NoC traffic within the framework of statically scheduled NoCs, reconciling the two seemingly conflicting goals of predictability and obfuscation. We propose Sentry-NoC to preserve the provably-secure properties of statically-scheduled NoCs, while ensuring obfuscation for side channel resilience, at low implementation overheads.

3 SENTRY-NOC

Sentry-NoC takes a statically-scheduled TDM NoC as its base, inherently supporting confidentiality, integrity, and availability by creating end-to-end isolated non-interference channels, while avoiding overhead in terms of buffering, flow-control, and dynamic arbitration. To reduce static NoC side-channel vulnerabilities, ***Sentry-NoC incorporates encryption, temporal and data obfuscation techniques into its base TDM NoC***, all the while mindful of ensuring ultra-low overheads.

3.1 Temporal Obfuscation

Sentry-NoC implements temporal obfuscation by randomly, and periodically, changing the TDM schedule of the statically-scheduled NoC. It randomly selects from multiple schedules that carry out the same traffic pattern. The TDM schedules are generated by changing the order of assigning routes or by inserting empty time-slots. The challenge lies in leveraging unused time slots in links to ensure minimal performance impact. We thus first study the link utilization for mesh and torus NoCs at different network sizes for worst-case traffic. We simulated an all-to-all traffic load scenario for various network sizes utilizing near-optimal schedule [28]. As shown in Fig. 1, link utilization increases with network size N^2, overwhelming the greater path diversity of larger NoCs. Yet even a 5x5 mesh still has 63% link utilization, or 37% unused. Moreover, torus NoCs have lower utilization as they have more path diversity, with their additional wrap-around links. ***These available timeslots and links present unused capacity that can be leveraged by multiple TDM schedules for obfuscating traffic at little performance impact.***

Fig. 2 illustrates our obfuscation scheme, where time is divided into obfuscation sessions (P), each session is divided into i temporal obfuscation sessions with a period (T), where different TDM schedules are assigned randomly. T is divided into j encryption sessions with a period (E), where different encryption keys are assigned. The E interval is divided to (t) which equals the TDM schedule period. Because of this, the difficulty level (number of traces required for a successful side-channel attack) is multiplied by $i \times j$, significantly boosting side-channel resilience. Encryption sessions thus reduce the number of temporal obfuscation sessions needed, which in turn lowers the timing overhead. We follow the AES-CTR recommendation to change the key after encrypting data 2^{12} times [16]. Hence, latency and throughput overhead of temporal obfuscation is negligible (see Section 4.3).

3.2 Data Obfuscation

Side channel attacks, typically based on differential or correlation power analysis [29], are correlated with the dynamic power consumption or switching activity of a chip. Messerges et al. [23] showed that voltage magnitude is directly proportional to the number of bit transitions (state changes). Similarly, ElectroMagnetic (EM)-based side-channel attacks are correlated to the dynamic state of the transistor: variations in the current is directly proportional to the EM field emission, which attackers can utilize to extract information [31]. Hence, obscuring the data on link interferes with its switching activity can mitigate power and EM side-channel

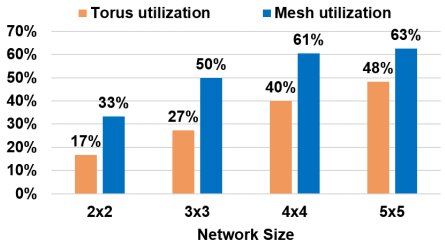


Figure 1: Average link utilization of statically-scheduled meshes/ torus, with all-to-all traffic – ($n \times n$-1), XY routing.

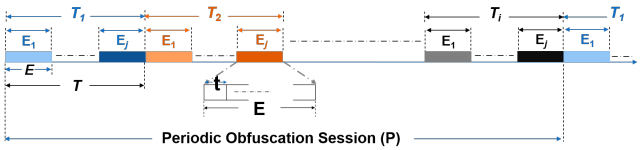


Figure 2: Temporal Obfuscation Scheme.

attacks. Thus we propose to implement data obfuscation alongside temporal obfuscation to boost side-channel resilience.

3.2.1 ***Data Inversion.*** Sentry-NoC obscures power and EM leakage by obfuscating the switching activity over the links utilizing data inversion. We utilize data inversion to change the number of data transitions over the link; thus, the switching activities are correlated to the data or inverted data, foiling side-channel attacks.

3.2.2 ***Encryption.*** As a statically-scheduled NoC ensures confidentiality by design, encryption is not required to protect against man-in-the-middle attacks. Sentry-NoC does not require encryption for the protection of confidentiality or integrity. ***Instead, Sentry-NoC uses encryption sessions*** [16], ***to enhance the NoC's immunity against side-channel attacks by securing raw data on links.*** This reduces the encryption overhead compared to a secure dynamic NoC. While a dynamic NoC needs a unique encryption key to secure the communication between each IP pair, a statically-scheduled NoC can use only one encryption key per session for all communication between IPs. Malicious IPs cannot access NoC resources due to the limitation of traffic isolation and non-interference communications. So, ***the ciphering is simplified to utilize one encryption key rather than n encryption keys for n IPs in dynamic NoCs.***

We propose the use of the Advanced Encryption Standard Counter Mode (AES-CTR). AES-CTR is recommended as one of the most secure modes [16]. In addition, it supports a parallel implementation unlike AES chaining modes. To reduce overhead and improve NoC performance, we divide the AES implementation to software and hardware components, with the secure engine running software that generates and delivers keys to network interfaces, while the hardware NoC implements XOR to encrypt the plain data with the encryption keys. The key generation and delivery is performed rarely, and hence can be done in software with little performance impact, while the hardware can be kept simple, using only XORs. Table 1 summarizes the differences between the state-of-the-art secure dynamic NoC solutions and our proposed Sentry-NoC design.

Table 1: Secure Dynamic NoC vs. Sentry-NoC.

	Attack	Secure Dynamic NoC	Sentry-NoC
Confidentiality	Unauthorized read, write or reconfiguration	• Encryption and Access Control Schemes • Firewall • Security Zone	Inherently Implemented
	Side-Channel Attacks (Run-time Observation)	• Task Migration • IP Mobility • Route Randomization • Scrambler, Shuffle, Inverting • Operation Redundancy	• Temporal Obfuscation • Data Obfuscation - Data Inversion - Encryption
Integrity	Data Corruption and Alteration	• Packet Tag - Hash function - Path ID Tag - CRC/ AMD Tag - Power Profile Tag	Inherently Implemented
Availability	• Deadlock • Livelock • Flooding • Replay • BW Denial	• Traffic Monitor and Analysis • Non-interference Communications	Inherently Implemented

4 NOC-LEVEL SETUP AND EVALUATION

In this section, we detail the microarchitecture and experimental setup, evaluate the effectiveness of Sentry-NoC against side-channel attacks, and assess its performance and area and power overheads.

4.1 Microarchitecture

The centerpiece of Sentry-NoC is the network interface that houses the TDM slot table, as Sentry-NoC schedules at the granularity of source-destination flows, and realizes temporal and data obfuscations. ***Exclusively, the secure engine communicates with the network interfaces to update the encryption key and TDM schedule. The secure engine is trusted*** (threat model, Section 2.1). ***Therefore, it is difficult to attack the TDM tables.*** The NoC routers remain low-overhead, with no logic or buffering, comprised of multiplexer switches and wires that are set according to the TDM schedule. Thus hardware overhead is minimal.

4.1.1 ***Network Interface.*** The network interface microarchitecture is shown in Fig. 3. The input buffer unit is where packets are received and stored to be injected into the NoC. The slot-table unit stores the TDM schedule. The output unit applies encryption and data inversion to the incoming and outgoing packets. The data inversion unit inverts the switching activity on the data link by XORing the encrypted packet with the data-inversion flag bit set by the secure engine. The control unit receives and reads the control packets then sets the control signals. The timer synchronizes the start of the operation state at all network interfaces.

4.1.2 ***Router.*** The TDM scheduler handles the routing and ensures contention freedom. Thus, the router microarchitecture is simplified to a crossbar. We implemented a multiplexer-based crossbar, whose input and output select signals are driven by the slot table module. At boot time, the network interface sets the router to an initial setup phase where the network interface can receive the control packets from the secure engine. After completing the obfuscation and encryption programming, the router operates based on the TDM schedule.

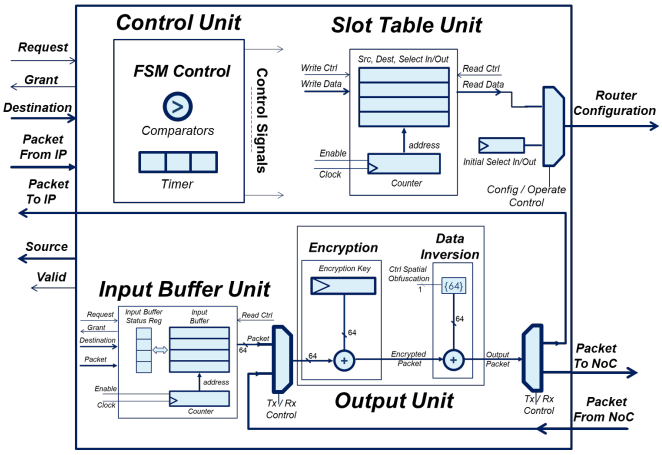


Figure 3: Sentry-NoC: Network Interface Microarchitecture.

4.1.3 ***Secure Engine.*** The Sentry-NoC secure engine is much simpler than those used with dynamic NoCs [24], as its task is to mitigate side-channel attacks only, not to enforce security policies nor monitor traffic at runtime. ***At boot time, the secure engine programs network interfaces. It delivers TDM schedules and encryption keys to all nodes.*** Then, according to the communication rate of the applications, E and T are defined (Fig. 2). E and T are inversely proportional to the communication rate to counterbalance the side-channel resilience.

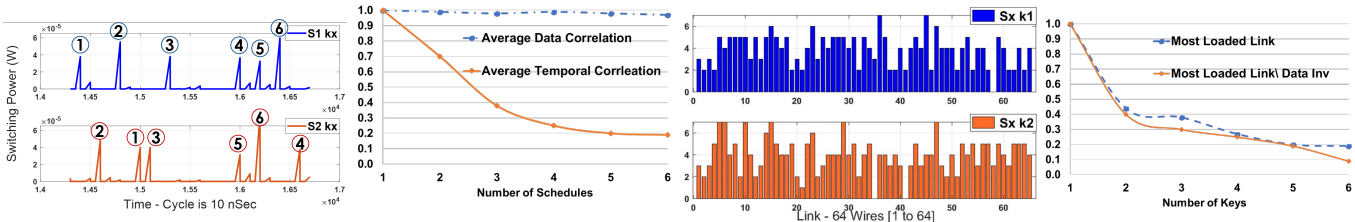
In our implementation, Sentry-NoC's secure engine is realized as a FSM. The states are: 1) **Configure**: sets the communication path to each node, and the flag-bit of the data inversion session, 2) **Encrypt**: handles the encryption sessions, generates and delivers encryption keys to network interfaces, 3) **Obfuscate**: manages the temporal obfuscation session, creates and provides the TDM slot tables to network interfaces, and 4) **Operate**: initializes the NoC to start the operation.

4.2 Side Channel Resilience of Sentry-NoC

For our 4×4 Sentry-NoC implementation, we inject synthetic traffic to mimic an Embedded SoC scenario (details, Fig.4[*]). We apply various TDM schedules, encryption keys with and without data inversion, and calculate the traces' correlation as an evaluation for obfuscation effectiveness. Similar to attack scenarios, traces are collected for a certain period related to the input data to uncover the encryption key.

4.2.1 ***NoC-based Switching Power Activities.*** To generate the switching power activity traces, we apply the same methodology adopted in [29]. We perform functional simulation on the Sentry-NoC gate-level netlist, analyze the waveforms, obtain the power consumption data on time-based mode by PTPX (power traces), and apply power analysis using MATLAB.

In a power correlation attack, the attack is premised upon on collecting power traces and uncovering the correlation coefficients between the actual power and the hypothetical power, in order to guess the encryption key. Sentry-NoC introduces temporal obfuscation to obscure the switching power activity, thus reducing the temporal correlation between the traces.



(a) Temporal Obfuscation. (b) Average Temporal/ Data Correlations. (c) Encryption/ Data Obfuscation. (d) Average Data Correlation.

Figure 4: Sentry-NoC: (a,b) Network-Switching Power Activities / (c,d) Link-Hamming distance transitions.

*Mesh-4×4, Link: 64 bits, TDM: 30 slots, Traffic: hot-spot:*n*-to-1/ 1-to-*n*: periodic injection (a sensor sends packets periodically).

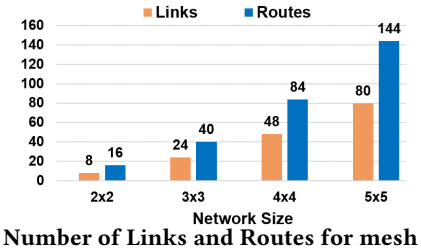


Figure 5: Number of Links and Routes for mesh topology.

Fig. 4a shows the switching power activity (in Watts) with two different schedules but with the same encryption key and data inversion mode. We observe that Sentry-NoC's temporal obfuscation shifts traffic events to different time points, thus obscuring the switching power profile temporally, disturbing the attacker's ability to uncover the correlation between input data and the switching power activity. An improved attack mitigation can be seen with lower correlation values. As observed in Fig. 4b, the average temporal correlation reduces as the number of schedules increase, demonstrating the effectiveness of temporal obfuscation in lowering the correlation, ensuring high resilience to side-channel attack. On the other hand, utilizing various encryption keys with or without data inversion along with the same TDM schedule affects data transitions. It leads to a difference in the magnitude of power values for the same event. In this example, the events still occur at the same time points with no effect on the temporal correlation. This result matches our intuition, as encryption and data inversion are data obfuscation techniques, not temporal obfuscation techniques.

4.2.2 ***Link Transition Hamming Distance.*** Power and electromagnetic attacks are based on the dynamic power and switching activity which are directly related to data transitions on the link. Hamming distance, therefore, can be a useful metric reflecting link data transitions which directly affect the relative power consumption of the CMOS circuit [21, 29]. With our cycle-level RTL simulations, link activity is faithfully captured. For every two consecutive cycles, we calculate the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions per link as: $HD(l_i) = \sum_{t=0}^{q}(b_i(t+1) \oplus (b_i(t))$, where b is the bit value (0 or 1), i is the wire number, q is the time period of the trace.

We plot the Hamming distance of the 64 wires for the most-loaded link within the period q utilizing two different encryption keys with the same TDM schedule and data inversion. In Fig. 4c, we observe how different encryption keys change the Hamming distance of the wires, since encryption changes the bits values. However, the number of data transitions remains fixed for various schedules, as the different schedules shift traffic events to different time points, but do not change the data transitions over the link. Utilizing multiple TDM schedules is a temporal obfuscation technique, not a data obfuscation technique. Fig. 4d shows the average data correlation for varying number of encryption keys with and without data inversion for the most-loaded link. As observed, the average data correlation reduces as the number of keys increases.

These experiments show Sentry-NoC effectively achieving both temporal and data obfuscation. Utilizing six schedules, six keys, and data inversion, ***Sentry-NoC reduces average temporal correlation and the average data correlation by 81% and 91%, respectively.*** Hence, Sentry-NoC effectively mitigates side-channel attacks.

Scalability. Fig. 5 illustrates the number of links and routes for a mesh topology with different network sizes. The number of links and routes increases as network-size scales up, giving the capacity to create more TDM schedules. As the number of TDM schedules increase, the difficulty level (number of traces required for a successful side-channel attack) is multiplied by $i \times j$, as discussed in Section 3.1, significantly boosting side-channel resilience. Hence, the obfuscation efficiency will improve with larger network-size.

4.3 Sentry-NoC's Performance

Sentry-NoC is implemented in Verilog, synthesised on a commercial 22nm process. For 2×2-mesh with 64-bits links, Sentry-NoC NIC+Router consume 1.69mW at 1.2GHz with 2,745μm^2. Compared to baseline insecure static NoC, the overhead is just the output data obfuscation unit (534μm^2, 0.49mW), and additional FSM states (79μm^2, 0.11mW). This overhead is reasonable for 2×2 size, which is the worst case. As the network size scales up, the slot table and input buffer modules (Fig. 3) area increase, thus the overhead ratio reduces.

Also, we compare Sentry-NoC's area footprint against prior secure NoCs. We selected prior designs that reported absolute area numbers, then translate the area into transistor count based on the transistor density of each technology node [6] (see Table 2). All prior works assume a dynamic NoC and ***these area overheads do not include the additional footprint of a dynamic NoC over static NoCs.***

We see that Sentry-NoC shows extremely low overhead, while addressing the most important security concerns compared to previous solutions [12, 24]. Efficiency improves by 4× (compared to a limited SPI security wrapper) and by 98× for NoC-ARK's RISC-Core security monitor [24]. NoC-ARK has a significant overhead compared to other designs as the security wrapper analyzes and

detects many attack and violation events. Events differ between IPs based on functionality and activity. In addition, the wrapper tracks and stores all violations to inform the secure engine.

These results confirm our claim that dynamic NoCs are inherently complex and costly for securing SoCs because they maintain high run-time flexibility in terms of communications and shared resources. Whereas starting with a TDM static NoC, Sentry-NoC can cover almost all security attributes with a lightweight design. Sentry-NoC secures SoC communication, providing extensive protection against confidentiality and integrity breaches, and complete protection against availability breaches. Moreover, Sentry-NoC defends against side-channel attacks by applying temporal and data obfuscation techniques.

Frequency and Critical-Path. Sentry-NoC's simplicity leads to a shorter critical path. It can run at a maximum frequency of 1.28GHz, 1.37× DynamicNoC-Insecure's maximum frequency of 935MHz [19], assuming single-cycle routers.

Latency-throughput. Sentry-NoC matches the average latency of the insecure baseline static NoC before saturation, but has slightly lower saturation throughput than dynamic NoCs (Fig. 6). Sentry-NoC has shorter critical path (1.28GHz), which NoC-level simulations cannot capture. Therefore, there can be additional performance wins over dynamic NoCs when cycle time is considered (see SoC-level case study 4.4).

TDM configuration overhead. It takes $m \times (t+x)$ to reconfigure the TDM tables, where m is the number of nodes, t is the schedule period, and x is the average number of cycles to set the path to node n_i. This overhead is negligible compared to temporal session **T** (in the range of millions of cycles, Fig (2)).

Bandwidth Efficiency. DynamicNoC utilizes header bits to establish communication between various nodes. The header size is 17 bits [19], adding 26% overhead of 64 link-bandwidth. However, Sentry-NoC utilizes the full link bandwidth as there is no need for headers, with the routers set according to the TDM tables.

4.4 SoC-Level Setup and Evaluation

We evaluate Sentry-NoC within a 9-component SoC, containing accelerators, memories and I/O IP blocks running a gesture-based face detection application. In this application, a gesture (hand wave) activates face detection and checks for the possible presence of face in the captured image. If detected, the detected face and coordinates are encrypted and saved. The goal of this experiment is to compare the system characteristics (side channel resilience, application performance, power and area) of Sentry-NoC against a conventional dynamically routed NoC baseline without additional security features built-in.

Case Study Experimental Setup. Our modeled SoC is built to enable low-power gesture-based face detection. It contains two embedded in-order Shakti E-class cores (C0, C1) [3] with 3 pipeline stages, with respective scratch-pad memories (M0, M1) of size 136KB each (8KB for Boot and 128KB for Code memory), a memory controller (M2) to off-chip storage, two accelerators (A0: face-detection, A1: AES-128), and two peripheral units for sensor input data (P0: inertial measurement unit (IMU), P1: camera). The CMOS color image sensor, which is capable of capturing an 8-bit grayscale RAW image at a resolution of 80px×60px, transfers 4,800 bytes per image (30fps sends 141kB). Fig. 7 lists the communication flows between the various IP blocks.

Table 2: Comparison: Sentry-NoC vs. Secure NoCs.

Security modules	Security Coverage C + I + A + R	Tech(nm) / area μm^2	# Tr	Over-head%
Sentry-NoC	✓ ✓ ✓ ✓	**22 / 613**	**9195**	**1.0×**
CRC-32 [11]	✗ ✓ ✗ ✗	45 / 2800	8696	0.9×
AMD [11]	✗ ✓ ✗ ✗	45 / 12800	39752	4.3×
OCP + XOR-Cipher [9]	✓ ✓ ✗ ✓	45 / 5384	16721	1.8×
OCP + Security Domains [27]	✓ ✓ ✗ ✗	45 / 6982	21684	2.3×
Non-interference Communication [30]	✓ ✗ ✓ ✗	45 / 22120	70413	7.7×
OCP + Access Rights [12]	✓ ✓ ✓ ✗	130 / 156000	97344	10.5×
SPI Security Monitor [24]	✓ ✓ ✓ ✗	32 / 5456	38971	4.2×
RISC-Core Security Monitor [24]	✓ ✓ ✓ ✗	32 / 290496	902161	98.1×

C:Confidentiality, I:Integrity, A:Availability, R:Side-Ch Resilience.
Tech (nm)|Tr Density (MTr/mm) 130|0.624 45|3.1 32|7.1 22|15.0

Table 3: Area, power and latency results of the MPSOC

NoC- 3x3	Area (um^2)	Power (mW)	Latency (cycles)
Dynamic-NoC (1-VC)	139,418	11.36	3462
Dynamic-NoC (2-VC)	199,987	16.44	2862
Static-NoC	37,203	2.20	3256
Sentry-NoC	**45,302**	**2.82**	**3267**

MPSoC system, synthesized at 100MHz in 22nm, with Sentry-NoC consumes 13.51mW with area 1,582,413 μm^2 while the MPSoC system with the Dynamic NoC consumes 22.05mW with area 1,676,529 μm^2

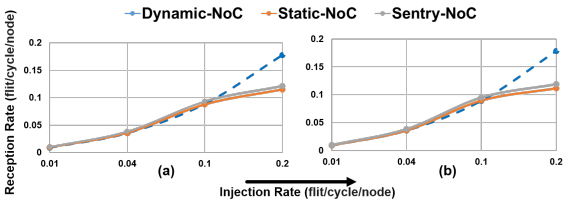


Figure 6: Reception rate with (a) uniform-random and (b) bit-complement traffic patterns, Mesh NoC-size: 3×3.

NoC architecture. The baseline dynamic NoC is implemented with the BSV OpenSMART NoC generator [19]. It is configured with a 3×3 mesh, wormhole flow control, and 4 flit buffers per Virtual Channel (VC). The NICs interfaces of both NoCs contain input buffers of depth 8. Sentry-NoC uses a 32-slot TDM schedule tailored to this application, with 25 slots allocated to the P1-A0 flow (TDM Schedule in Fig. 7).

Results. To examine the system performance, we compare Sentry-NoC with Static and Dynamic NoCs in terms of power, area, and latency at 100MHz (details at Table 3, the frequency is set to 100MHz

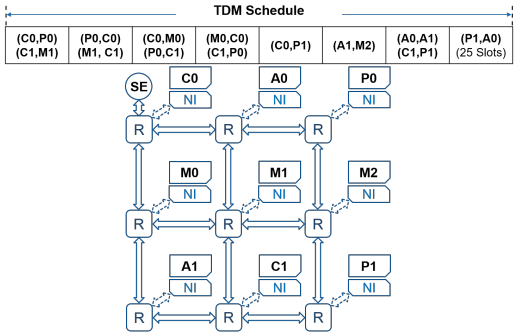


Figure 7: MPSoC system and its communication pattern.

Comm.: P0→C0; P0→C1; C0→P0; C1→P0; C0→P1; C1→P1; P1→A0; A0→A1; A1→M2; C0→M0; M0→C0; C1→M1; M1→C1

due to Shakti E-class core limitation [3]). We measure the application execution latency as the time taken from the triggering IMU (P0) message to the last packet received at the memory controller (M2). The latency of our secure Sentry-NoC is almost identical to StaticNoC-Insecure with no penalty and 5.6% faster than the DynamicNoC-Insecure 1-VC (due to the network congestion). While DynamicNoC-Insecure 2-VC is 13.4% faster but at the expense of 4× area and 5.8× power. Sentry-NoC achieves this level of security (confidentiality, integrity and availability) at 1.48× lower power and 5.6% less area. In comparison, previous works [24] would require between 2× to 98× of the NoC area, significantly increasing system costs (see Table 2).

Obfuscation evaluation. We evaluate the side channel resilience of Sentry-NoC within this SoC. Sentry-NoC (4 schedules, 4 keys) reduces average data correlation by 76%.

5 CONCLUSION

Unlike prior secure NoC designs that are based off dynamic NoCs, Sentry-NoC uses statically-scheduled NoCs as the base, and thus only has to mitigate side-channel attacks. Sentry-NoC provides a secure platform for communication among malicious IPs. Sentry-NoC offers extensive protection against confidentiality and integrity attacks, complete protection against availability attacks. Moreover, it provides protection against side-channel attacks by applying temporal and data obfuscation techniques.

The goal of Sentry-NoC is to enable side-channel resilience while maintaining high performance, energy efficiency, and low overhead compared to previous works. Our results demonstrate that through its low overhead, strong security properties and the ability to maintain performance, Sentry-NoC can be a key enabler for next-generation secure SoCs.

ACKNOWLEDGMENTS

The authors acknowledge the support from the National Research Foundation, Singapore (Grants NRF-RSSS2016-05 and NRF2018NCR-NCR002).

REFERENCES

[1] [n.d.]. https://greenwaves-technologies.com/.
[2] [n.d.]. https://www.dovermicrosystems.com/.
[3] [n.d.]. https://shakti.org.in/.
[4] 2015. https://www.symantec.com/connect/blogs/iot-smart-home-giving-away-keys-your-kingdom/.
[5] 2017. https://www.bbc.com/news/technology-38573074/.
[6] 2017. https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/03/Ruth-Brain-2017-Manufacturing.pdf/.
[7] 2019. www.ericsson.com/en/mobility-report/reports/november-2019/.
[8] 2019. https://www.csa.gov.sg/news/publications/iot-security-landscape/.
[9] Dean Michael Ancajas, Koushik Chakraborty, and Sanghamitra Roy. 2014. Fort-nocs: Mitigating the threat of a compromised noc. In Proceedings of the 51st Annual Design Automation Conference. 1–6.
[10] Travis Boraten and Avinash Kodi. 2018. Mitigation of hardware trojan based denial-of-service attack for secure nocs. J. Parallel and Distrib. Comput. 111 (2018), 24–38.
[11] Travis Boraten and Avinash Karanth Kodi. 2016. Packet security with path sensitization for NoCs. In 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 1136–1139.
[12] Leandro Fiorin, Gianluca Palermo, and Cristina Silvano. 2013. A configurable monitoring infrastructure for NoC-based architectures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22, 11 (2013), 2438–2442.
[13] Salma Hesham, Jens Rettkowski, Diana Goehringer, and Mohamed A Abd El Ghany. 2016. Survey on real-time networks-on-chip. IEEE Transactions on Parallel and Distributed Systems 28, 5 (2016), 1500–1517.
[14] Dongil Hwang, Myonghoon Yang, Seongil Jeon, Younghan Lee, Donghyun Kwon, and Yunheung Paek. 2019. Riskim: Toward complete kernel protection with hardware support. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 740–745.
[15] Leandro Soares Indrusiak, James Harbin, Cezar Reinbrecht, and Johanna Sepúlveda. 2019. Side-channel protected MPSoC through secure real-time networks-on-chip. Microprocessors and Microsystems 68 (2019), 34–46.
[16] Darshana Jayasinghe, Roshan Ragel, Jude Angelo Ambrose, Aleksandar Ignjatovic, and Sri Parameswaran. 2014. Advanced modes in AES: Are they safe from power analysis based side channel attacks?. In 2014 IEEE 32nd International Conference on Computer Design (ICCD). IEEE, 173–180.
[17] Natalie Enright Jerger and Li-Shiuan Peh. 2009. On-Chip Networks, Synthesis Lectures on Computer Architecture. Morgan & cLaypool publishers (2009).
[18] Hemangee K Kapoor, G Bhoopal Rao, Sharique Arshi, and Gaurav Trivedi. 2013. A security framework for noc using authenticated encryption and session keys. Circuits, Systems, and Signal Processing 32, 6 (2013), 2605–2622.
[19] Hyoukjun Kwon and Tushar Krishna. 2017. OpenSMART: Single-cycle multi-hop NoC generator in BSV and Chisel. In 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 195–204.
[20] Ruby B Lee. 2013. Security basics for computer architects. Synthesis Lectures on Computer Architecture 8, 4 (2013), 1–111.
[21] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. Power analysis attacks: Revealing the secrets of smart cards. Vol. 31. Springer Science & Business Media.
[22] Dan Meng, Rui Hou, Gang Shi, Bibo Tu, Aimin Yu, Ziyuan Zhu, Xiaoqi Jia, and Peng Liu. 2018. Security-first architecture: deploying physically isolated active security processors for safeguarding the future of computing. Cybersecurity 1, 1 (2018), 1–11.
[23] Thomas S Messerges, Ezzat A Dabbish, and Robert H Sloan. 2002. Examining smart-card security under the threat of power analysis attacks. IEEE transactions on computers 51, 5 (2002), 541–552.
[24] Atul P Nath, Srivalli Boddupalli, Swarup Bhunia, and Sandip Ray. 2019. ARK: Architecture for Security Resiliency in SoC Designs with Network-on-Chip (NoC) Fabrics. Technical Report. University of Florida Gainesville United States.
[25] Maire O'Neill et al. 2016. Insecurity by design: Today's IoT device security problem. Engineering 2, 1 (2016), 48–49.
[26] Tomás Picornell, José Flich, Carles Hernández, and José Duato. 2019. DCFNoC: A Delayed Conflict-Free Time Division Multiplexing Network on Chip. In Proceedings of the 56th Annual Design Automation Conference 2019. 1–6.
[27] Joël Porquet, Alain Greiner, and Christian Schwarz. 2011. NoC-MPU: A secure architecture for flexible co-hosting on shared memory MPSoCs. In 2011 Design, Automation & Test in Europe. IEEE, 1–4.
[28] Martin Schoeberl, Florian Brandner, Jens Sparsø, and Evangelia Kasapaki. 2012. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip. IEEE, 152–160.
[29] Weiwei Shan, Xin Chen, Bo Li, Peng Cao, Jie Li, Gugang Gao, and Longxing Shi. 2013. Evaluation of correlation power analysis resistance and its application on asymmetric mask protected data encryption standard hardware. IEEE Transactions on Instrumentation and Measurement 62, 10 (2013), 2716–2724.
[30] Hassan MG Wassel, Ying Gao, Jason K Oberg, Ted Huffmire, Ryan Kastner, Frederic T Chong, and Timothy Sherwood. 2013. SurfNoC: a low latency and provably non-interfering approach to secure networks-on-chip. ACM SIGARCH Computer Architecture News 41, 3 (2013), 583–594.
[31] Oskar Westman. 2018. Electromagnetic analysis of AES-256 on Xilinx Artix-7. (2018).