

Research Article

An Automatic Design Flow for Data Parallel and Pipelined Signal Processing Applications on Embedded Multiprocessor with NoC: Application to Cryptography

Xinyu Li and Omar Hammami

Laboratory of Electronics and Computer Engineering, ENSTA ParisTech, 32 boulevard Victor, 75739 Paris, France

Correspondence should be addressed to Xinyu Li, xin-yu.li@ensta.fr

Received 17 June 2009; Accepted 9 September 2009

Recommended by Andres Garcia

Embedded system design is increasingly based on single chip multiprocessors because of the high performance and flexibility requirements. Embedded multiprocessors on FPGA provide the additional flexibility by allowing customization through addition of hardware accelerators on FPGA when parallel software implementation does not provide the expected performance. And the overall multiprocessor architecture is still kept for additional applications. This provides a transition to software only parallel implementation while avoiding pure hardware implementation. An automatic design flow is proposed well suited for data flow signal processing exhibiting both pipelining and data parallel mode of execution. Fork-Join model-based software parallelization is explored to find out the best parallelization configuration. C-based synthesis coprocessor is added to improve performance with more hardware resource usage. The Triple Data Encryption Standard (TDES) cryptographic algorithm on a 48-PE single-chip distributed memory multiprocessor is selected as an application example of the flow.

Copyright © 2009 X. Li and O. Hammami. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

The International Technology Roadmap for Semiconductors (ITRSs) [1] predict that the future generation system on chip will experience an exponential increase in the number of processing engines and processors. Although parallel architecture and multiprocessors have been the focus of intensive research for decades [2, 3], multiprocessor system on chip [4] and emerging network on chip (NoC) issues [5] bring new challenge and constraints due to the embedded environment constraints such as area and limited in chip resources such as on chip SRAM. In addition submicron technology brings new trends by driving interconnect dominated designs. Although several commercial multiprocessors on chip devices have been proposed for signal processing applications [6–13], they are limited in their number of processors and they are not customizable. Image and signal processing applications in embedded systems are better served by customizable system on chip with appropriate system level design flow and possible design space exploration.

This helps to better tune embedded system and offer Pareto-like choices of configurations for system designers.

In this paper an automatic design flow is proposed for data parallel and pipelined signal processing application on embedded multiprocessor with NoC, and we implement block cipher TDES application on 48 cores signal chip distributed memory multiprocessor. The proposed flow explores first parallel software implementation through multi-FPGA emulation of the embedded multiprocessor designed for pipelined and data parallel application with emphasis on data locality. In a second phase the flow adds hardware accelerator connected as coprocessor to embedded multiprocessor through high level synthesis in order to finally propose a better performance solution to the designers.

The paper is organized as follows. Section 2 presents the related works. The proposed heterogeneous design workflow is described in Section 3, and design automation and emulation issues essential to efficient automatic design flow in Section 4. Section 5 introduces the multiprocessor

architecture used for embedded parallel software programming. In Section 6 the target application TDES algorithm is presented, then in Section 7 the implementation of the TDES into the platform and the results are discussed. Finally Section 8 draws conclusion and future work on the presented work.

2. Related Work

This proposed work is related to the embedded multiprocessor, system level synthesis design flow, and design space exploration of customizable embedded multiprocessors. A few examples of commercial multicore implementations have been proposed. They can be globally divided in 2 categories: (1) general purpose, (2) application specific. ARM ARM11MPcore [7], the MIPS MIPS32 1004 Core [8], and the Renesas/Hitachi SH-X3 [9] belong to the first category. Texas Instruments TMS320C6474/TMS320VC5441DSP [10–12], Freescale QorIQ P4080 [13], and the Toshiba Venezia multicore [14] are in the second category. The ARM11 MPcore is a classical shared memory 4 processors-based multiprocessor based on shared bus architecture with a snoopy cache coherency protocol (MESI). The MIPS32 1004 is a 1 to 4 multithreaded “base” cores (up to 8 hardware threads) with Coherence Management (CM) unit—the system “glue” for managing coherent operation between cores and I/O, I/O Coherence Unit (IOCU)-hardware block for offloading I/O coherence from software implementation on CPUs. Several multicore architectures are proposed by Texas Instruments. The Texas Instruments TMS320C6474 is a 3 DSP-based multicore architecture with switch central resource (SRC) as the interconnection between the 3 DSP and the memories. The 6474 device contains 2 switch fabrics through which masters and slaves communicate: (1) data switch (2) configuration switch. The data switch fabric is a high-throughput interconnect mainly used to move data across the system and connects masters to slaves via 128-bits data buses (SCR B) and 64-bit data buses (SCR A). The configuration switch is used to access peripheral registers. The Texas Instruments TMS320VC5441 is a 4-core multicore with shared bus between 2 cores and HPI for external accesses. The Freescale QorIQ P4080 is an 8-core multicore architecture with a Corenet coherency fabric. Each core is a high-performance Power Architecture e500mc cores, each with a 32-KByte Instruction and Data L1 Cache and a private 128-KByte L2 Cache. The CoreNet fabric is Freescale’s next generation front-side interconnect standard for multicore products. However, these devices have a limited number of processors and are not customizable.

Several embedded multiprocessor design flows have been reported in the literature. These flows are different on evaluation methods, system specification and application specification. The first is the evaluation methods on which the exploration is based. SoCDAL [15] and Daedalus [16] is SystemC simulation-based exploration for application to architecture mapping. Execution cycle is taken as the fitness of evolutionary algorithm. The Daedalus design flow is based on parallelization of a sequential program in C but do not consider hardware accelerators generation. System level

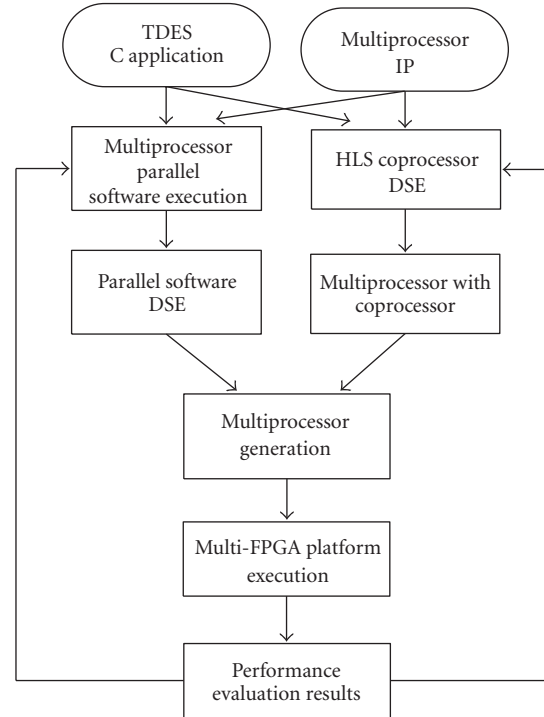


FIGURE 1: Automatic heterogeneous design flow.

architectural exploration is assured through the Sesame process and automated system-level synthesis through ESPAM. Finally the Xilinx Platform Studio tool is used for the generation of the FPGA-based system for prototyping and execution. SoCDAL uses a SystemC specification as input then designs space exploration through TF simulation and BCA TLM cosimulation. SystemCodesigner [17] is based on behavioral SystemC model. Hardware accelerators are generated using Forte Synthesizer and the hardware resource information reported from synthesis is added to the design space. The behavioral synthesizer goes through the exploration till rapid prototyping on FPGA. MOCDEX [18] is a multiobjective design space exploration flow based on multiprocessor emulation on FPGA platform but does not address parallel software mapping exploration. Most of the proposed flows have been implemented on limited number of IPs. In this paper the proposed methodology for data parallel and pipeline signal processing applications combines (1) multi-FPGA emulation for accurate and fast performance evaluation, (2) automatic multiprocessor generation up to 48 processors, synthesis, and execution, (3) accurate hardware NoC monitoring for accurate feedback for parallel program tuning, and (4) automatic synthesis and inclusion of hardware accelerators through High Level Synthesis (HLS).

3. Heterogeneous Design Flow

The heterogeneous design flow used for this study is described in Figure 1. The inputs of the flow are the TDES application provided as an ANSI C code and a small-scale multiprocessor (SSM) IP which will serve as the basic

TABLE 1: EVE Zebu-UF4 platform details.

Modules	Descriptions
FPGA	4 Virtex-4 LX200
DRAM	512 MBytes
SSRAM	64 MBytes
ICE	Smart and direct

component for the parallelization process. All performance results will be achieved through actual execution on a multi-FPGA platform. At first step, software parallelization is explored through direct execution on multi-FPGA platform to find out the best data parallel and pipelined configuration. During the exploration, data are partitioned into different groups and distributed to pipelined groups of processors; pipelined application is mapped onto available processors. The execution on the multiprocessor will be based on a fork-join multiple-data multipipeline exploiting both the data parallel and the pipelined feature of the TDES algorithm.

Having achieved maximum design space exploration through parallel programming, the second step is to explore coprocessor-based TDES parallel execution by incrementally adding TDES C-based synthesis generated coprocessor. Final step will compare both paths to select the most appropriate implementation. If the performance of software parallelization meets the system design objective, the parallelized application is executed on the multiprocessor. If not, coprocessors are added to multiprocessor to improve system performance with using more hardware resources.

4. Design Automation and IP Issues

The automatic design flow approach requires a tight and efficient combination and integration of EDA tools. The Zebu-UF multi-FPGA platform is used and different EDA tools from EVE [19], Xilinx [20], Arteris [21], and ImpulseC [22] are combined for a automatic and fast design flow.

4.1. Emulation: EVE Zebu-UF Platform. Zebu-UF platform is based on 4 Xilinx Virtex-4 LX200, built as an extended PCI card via a motherboard-daughter card approach.

The 4 FPGA-based system can emulate the equivalent of up to 6 million ASIC gates in a single card. Zebu-UF also includes on-board memory capacity based on 64 MBytes of SSRAM and 512 MBytes of DRAM memory chips via an additional memory board, which plugs into the PCI motherboard. Table 1 gives the details of this multi-FPGA card. As shown in Figure 2, there are 3 FPGA chips at one side of the card and the last one is at the other side.

4.2. EDA Tools Combination. Design automation tools of 4 commercial companies are combined together to generate the multi-FPGA MPSoC and the parallelized execution files for emulation as described in Figure 3.

The Xilinx EDK tool is used to generate the Small-Scale Multiprocessor (SSM), which is described in Section 5. Once

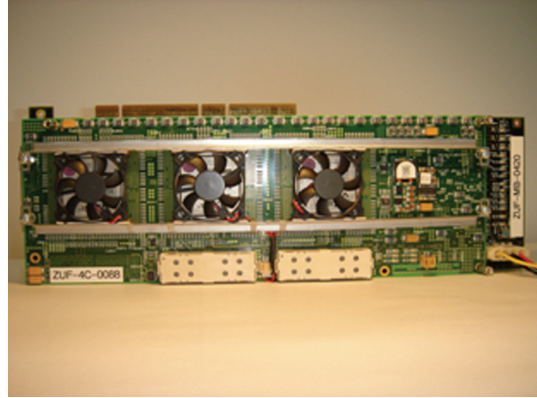


FIGURE 2: Eve Zebu-UF4 Platform.

the RTL files of SSM are generated, they are reused for the multi-FPGA large-scale multiprocessor synthesis, which can largely reduce system design time. Different NoC files are synthesized for each SSM on different FPGA chips of Zebu platform by changing the generic route-table according to the XY routing algorithm and the SRAM addresses on each FPGA. These NoC RTL files are generated by Arteris NoCcompiler tool, which allows the export of NoC using the Arteris Danube Library. Sequential C code of application can be synthesized to RTL codes by ImpulseC High Level Synthesis (HLS) tools to generate coprocessor to SSM IPs. Eve Zebu compiler takes the EDIF files converted by Xilinx synthesis tools for the implementation. Different SSM IPs are analyzed and distributed onto FPGAs. User constraints can be used to accelerate this incremental process. Finally Xilinx place and rout tools are used to generate the download bit files of FPGA. This phase can be parallelized to reduce the turnaround time. Application is partitioned and programmed into parallel C codes for each processor, compiled with Xilinx EDK tools to generate execution file of each processor. Finally system bit file is downloaded to multi-FPGA platform and application is executed.

4.3. High Level Synthesis: C-Based. Hardware accelerators are blocks of logic that are either automatically generated or manually designed to offload specific tasks from the system processor. Many math operations are performed more quickly and efficiently when implemented in hardware versus software. Adding powerful capability to FPGAs, hardware accelerators can be implemented as complex, multicycle coprocessors with pipelined access to any memory or peripheral in the system. They can utilize FPGA resources (such as on-chip memory and hard-macro multipliers) to implement local memory buffers and multiply-accumulate circuits. Using as many master ports as necessary, they can initiate their own read and write operations and access any I/O pin in the system. Hardware accelerators are a great way to boost performance of software code and take full advantage of the high-performance architecture of FPGAs. The design and software simulation of a hardware accelerator used a CAD tool called ImpulseC. This software

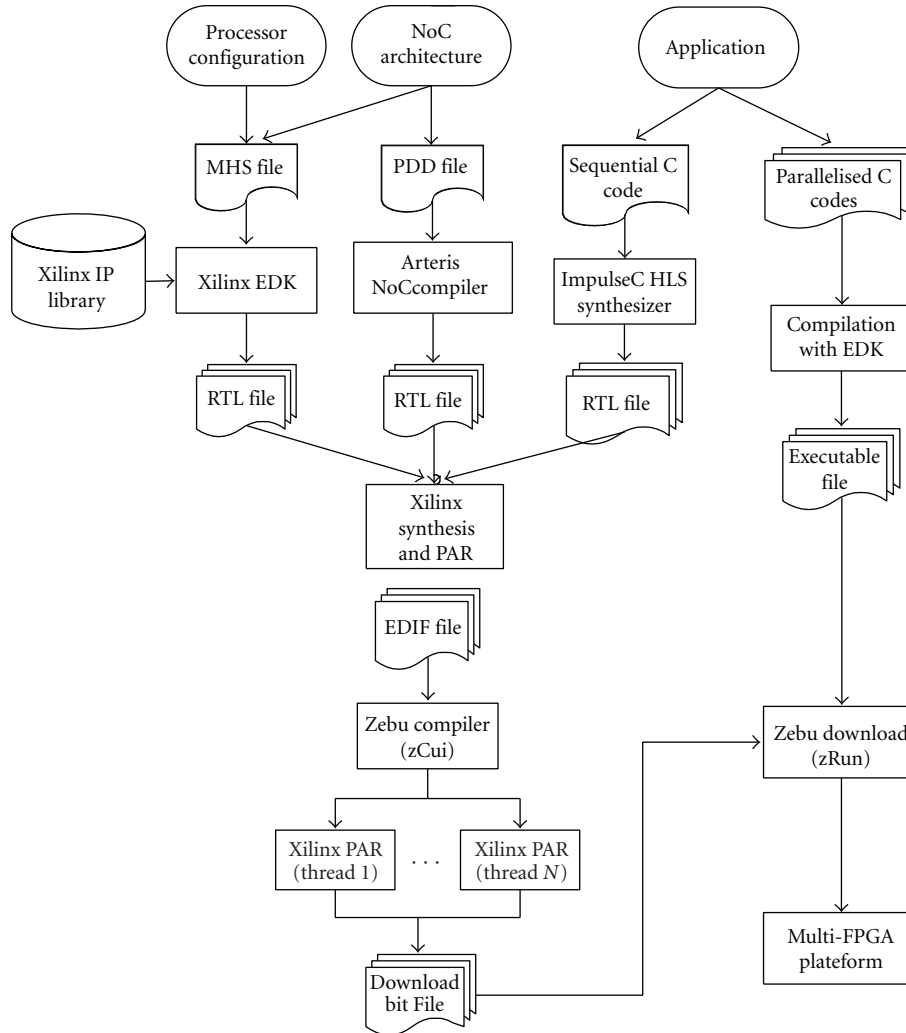


FIGURE 3: Workflow of multi-FPGA MPSoC.

allows the creation of applications intended for FPGA-based programmable hardware platforms similar to [23] and design space exploration at the HLS level [24, 25].

The ImpulseC compiler translates and optimizes ImpulseC programs into appropriate lower-level representations, including Register Transfer Level (RTL) VHDL descriptions as in Figure 4(b). Impulse CoDeveloper is an ANSI C synthesizer based on the Impulse C language with function libraries supporting embedded processors and abstract software/hardware communication methods including streams, signals, and shared memories. This allows software programmers to make use of available hardware resources for coprocessing without writing low-level hardware descriptions. Software programmers can create a complete embedded system that takes advantage of the high-level features provided by an operating system while allowing the C programming of custom hardware accelerators. The ImpulseC tools automate the creation of required hardware-to-software interfaces, using available on-chip bus interconnects. The FSL (Fast Simplex Link) interfaces of MicroBlaze processor are generated and used

for the interconnection between HLS accelerators and the processors as shown in Figure 4(a).

Concurrency Model. The main concurrency feature is pipelining. As pipelining is only available in inner loops, loop unrolling becomes the solution to obtain large pipelines. The parallelism is automatically extracted. Explicit multiprocessing is also possible to manually describe the parallelism.

Types. ANSI C types operators are available like int and float as well as hardware types like int2, int4, int8. The float to fixed point translation is also available.

Timing Specification. The only way to control the pipeline timings is through a constraint on the size of each stage of the pipeline. The number of stages of the pipeline and thus the throughput/latency is tightly controlled.

Memory. All arrays are stored either in RAM or in a set of registers according to a compilation option.

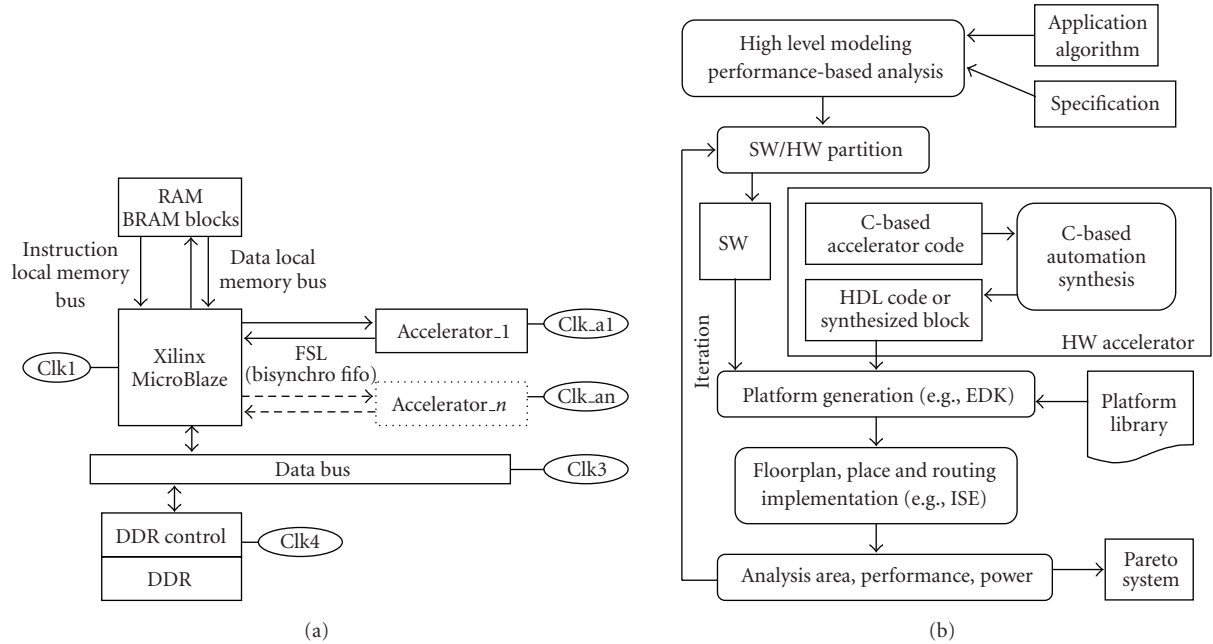


FIGURE 4: (a) Block diagram of accelerator connection forms, (b) C-based HW accelerated system design workflow.

An automatic design flow for C-based hardware accelerator based on the preciously described tool has been proposed and applied [25].

5. The Multiprocessor Architecture

Data parallel and pipelined applications require architectures with close communication between processing elements and locality memory resource for computation and data access. Mesh-based architectures are suitable for data parallel and pipelined applications. However, most mesh-based architectures have signal processing element per-switch while a cluster-based architecture allows more concurrency on local data.

5.1. Small-Scale Multiprocessor IP: SSM. The architecture of the small-scale multiprocessor, presented in Figure 5, is based on a mesh-based network on chip connecting 12 processors organized as with 3 processors and 2 SRAM on chip memories per switch. The network on chip topology is mesh for a better multi-FPGA implementation. This small-scale multiprocessor represents a 12-processor single-chip multiprocessor which is representative of current small-scale multiprocessor on chip.

The distributed memory SSM IP is composed of several IPs described in Table 2.

The design is OCP-IP [26] compliant which implies that the processor IP can be replaced by any other OCP-IP compliant processor IP while leaving the overall design identical. The OCP-IP protocol is used for the communication between the processors and Network on Chip (NoC). MicroBlaze [20] is a 32-bit 3-stage single issue pipelined Harvard style embedded processor architecture provided

by Xilinx as part of their embedded design tool kit. The MicroBlaze processor is flexible and gives the user control of a number of features such as the cache sizes, interfaces, and execution units like: selectable barrel shifter (BS), hardware multiplier (HWM), hardware divider (HWD), and floating point unit (FPU). A network interface is designed in order to make MicroBlaze processor compatible to OCP-IP protocol. This interface gets data from MicroBlaze through FSL link and transfers the data to the Network on Chip under OCP-IP protocol. Each MicroBlaze communicates with a 32-KB local memory via LMB bus on one side and with the Network Interface (OCP-IP) on the other side.

5.2. Arteris NoC. The design of the network on chip is based on Arteris Danube Library. The Arteris Danube library [21] includes the switch generator which is an essential building block of the NoC interconnect system. The main features of Arteris switch are (1) fully synchronous operation, (2) internal full crossbar: up to one data word transfer per MINI port and per cycle, (3) full throughput arbitration: up to one routing decision per input port and per cycle, wormhole routing to reduce latency, (4) freely cascading connection, supporting any loop-less network topology. It is possible to select the arbitration type of the switch among 4 possible values: Round-Robin, LRU, Random, and FIFO with default value Round-Robin. Several optional registers can be added in order to pipeline the switch.

Arteris switch uses worm-hole routing for packet transfer. Packets are combined with header, necker (request only), and data cells. Each field in cells carries special information. Most of the fields in header cells are parameterizable, and

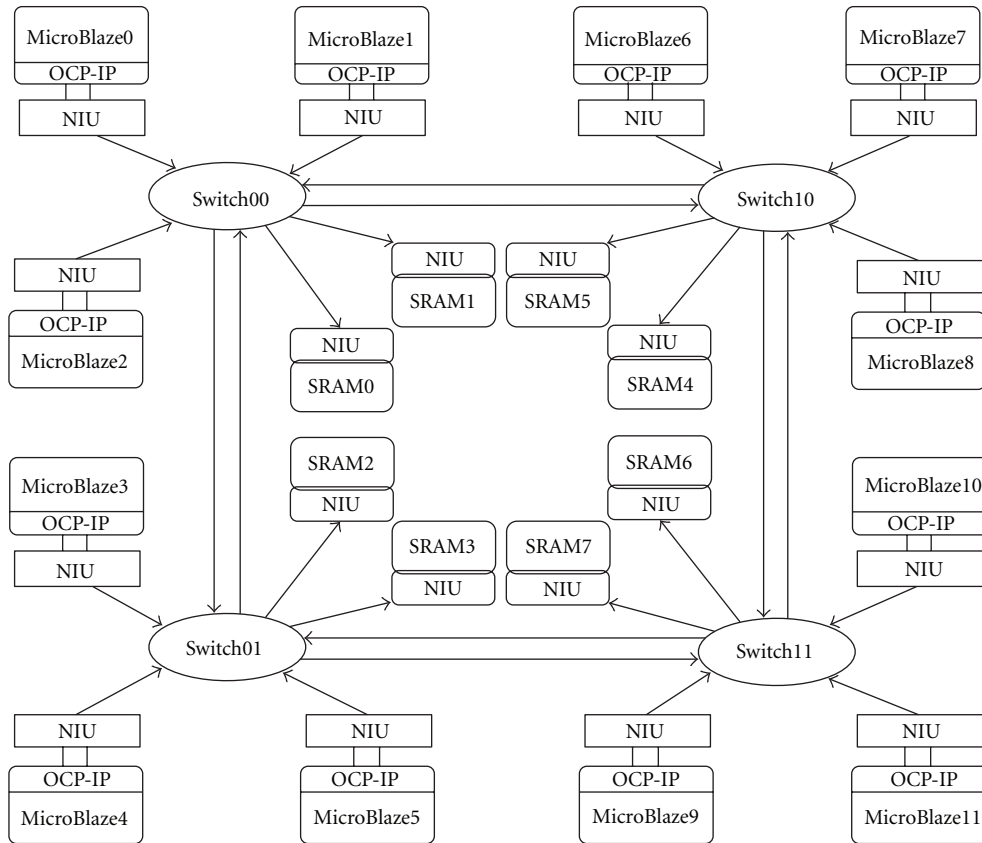


FIGURE 5: Small-scale multiprocessor IP.

TABLE 2: IPs of SSM multiprocessor.

IP component	Description	Source	Version	Qty
Processor	Soft core IP	Microblaze Soft core IP Xilinx	5.00 b	12
Memory	Soft core IP	Xilinx Coregen 96 KB	v.2.4.	8
Network on chip switch	Soft core IP	VHDL Arteris Danube library	1.10	4
Interchip	Soft core IP	VHDL Arteris Danube library	1.10	1

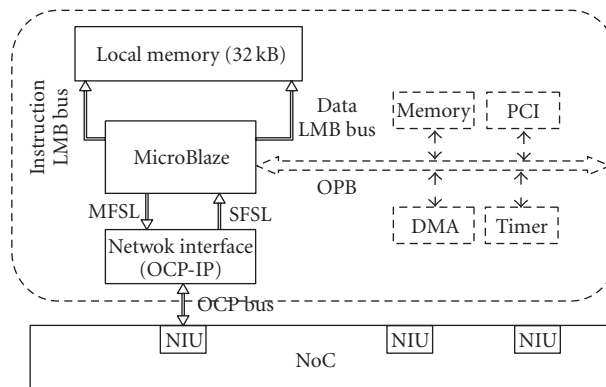


FIGURE 6: Processor tile.

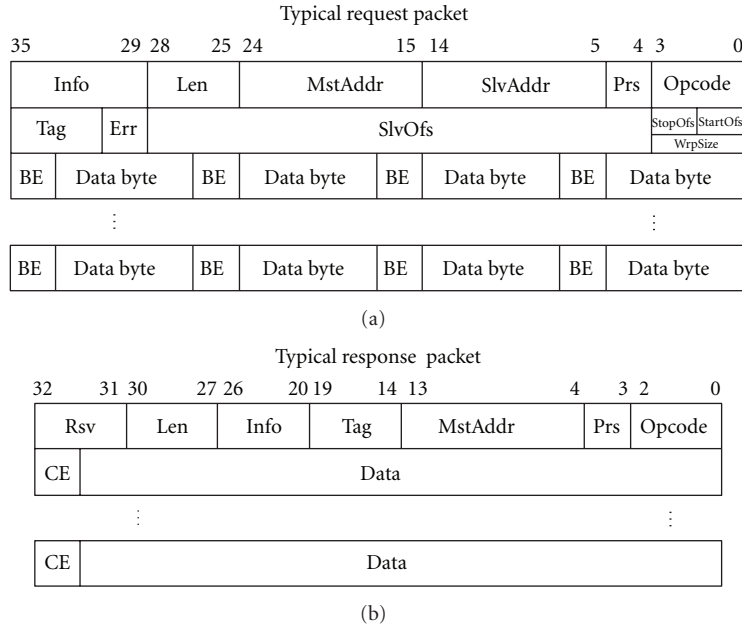


FIGURE 7: Arteris typical request and response packet.

some are optional. Typical request and response packet structure are shown in Figure 7. For request packets, a data cell is 32 or 36 bits wide. In the experiment, the width is fixed at 32 bits. For response packets, a data cell is always 33 bits wide. The multiprocessor is homogeneous. The NoC works with the same frequency with processing and memory cores.

5.3. Automatic Extension through Composed SSM: Overall Architecture. The previous SSM IP can be easily extended and composed up to a 48-processor multiprocessor organized as a 4×4 mesh of clusters. Each cluster includes 3 processor elements and 2 local memories. The architecture is symmetrical which makes it a good candidate for design modularity and IP-based design and reuse.

Due to its large size and prohibitive simulation time, emulation on FPGA platform is targeted for the performance evaluation of this multiprocessor architecture as well as for accurate area data. The emulation platform is the Eve Zebu-UF4 Platform.

5.4. NoC Monitoring. Network on chip traffic is monitored through hardware performance monitoring unit. The overall concept is to probe the Network Interface Unit (NIU) from initiator and target.

Statistics collector is added for each group of 3 processors and 4 on-chip SRAM and collected through a dedicated performance monitoring network on chip. The monitoring unit and performance monitoring network on chip use dedicated hardware resources and thus do not interfere of affect the measurement. There is a tradeoff between emulation frequencies and hardware monitoring area which is significant.

6. The TDES Algorithm

TDES algorithm was introduced as a security enhancement to the aging DES, a complete description of the TDES, and DES algorithms can be found in [27–29].

6.1. The Algorithm. The TDES is a symmetric block cipher that has a block size of 64 bit and can accept several key sizes (112, 168 bits) which are eventually extended into a 168 bit size key; the algorithm is achieved by pipelining three DES algorithms while providing 3 different 56 bits keys (also called key bundle) one key per DES stage. The DES is a block cipher with 64 bit block size and 54 bit key.

The TDES starts by dividing the data block into two 32 bits blocks which are passed into a Forward Permutation (FP) then criss-crossed in what is known as Feistel scheme (or Feistel Network) while being passed into a cipher Feistel function F , as illustrated in Figure 11, this operation is repeated for 48 rounds followed by one Inverse Permutation (IP). The F function expands the 32 bits half-block input into 48 bits that is mixed with a round key that is 48 bits wide, the result is divided into 8 blocks 6 bits each which in turn are passed into 8 Substitution Box (S-Box) that returns only 4 bits each making an output of 32 bits. Round keys are calculated for each round by expanding the 56 bits key through a specific key scheduling process, then dividing the result into 48 bits key. Figure 10 shows the Feistel F function.

The TDES algorithm clearly processes data on a pipelined mode in both the encryption and the decryption mode.

6.2. Operation Mode. Block cipher algorithms have different operation modes; the simplest is called Electronic Code Book (ECB) in this mode and the block cipher is used directly as illustrated in Figure 12.

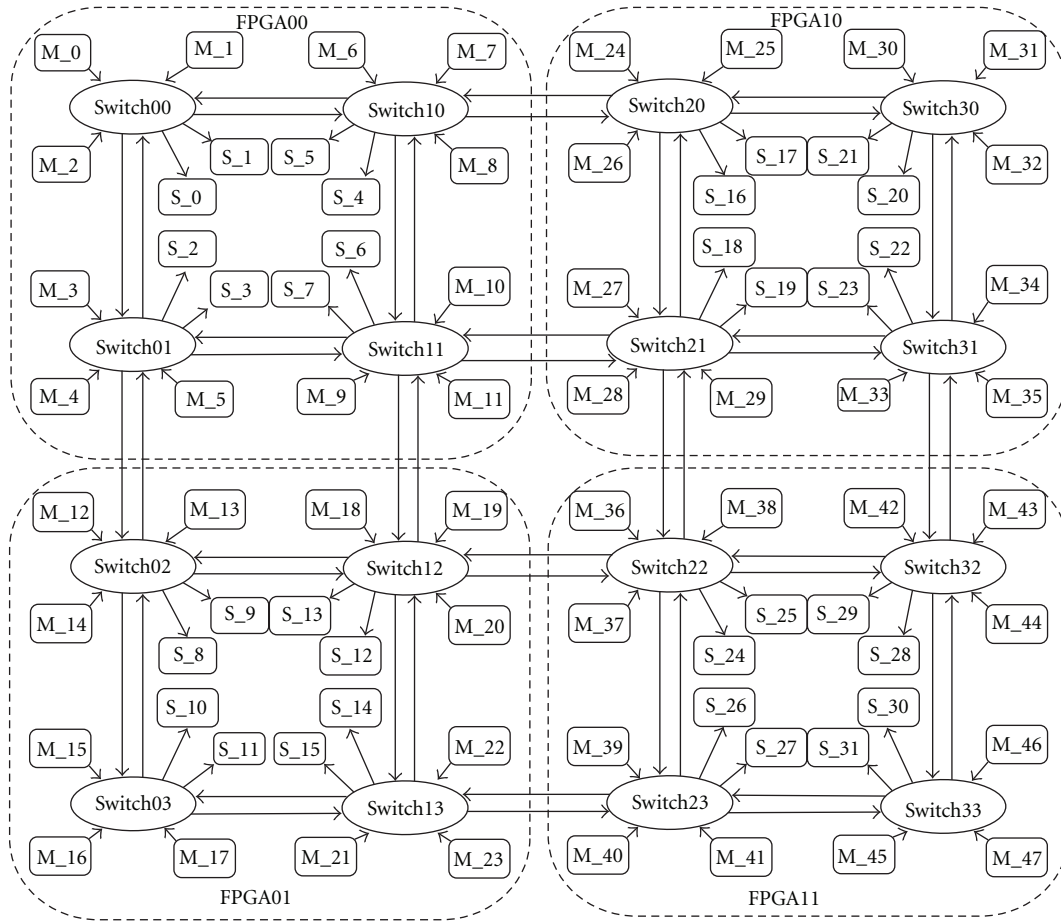


FIGURE 8: 48 processors multiprocessor architecture.

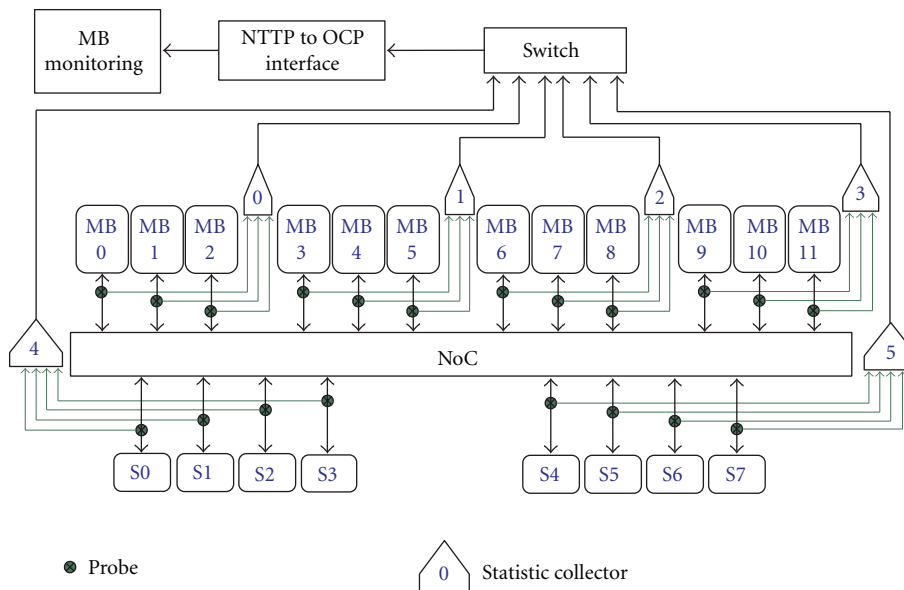


FIGURE 9: Performace monitoring network on each FPGA.

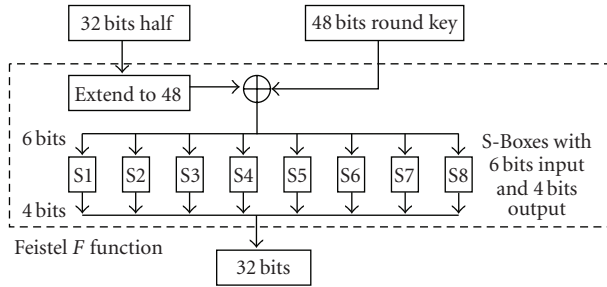


FIGURE 10: Feistel function F (S-Boxes).

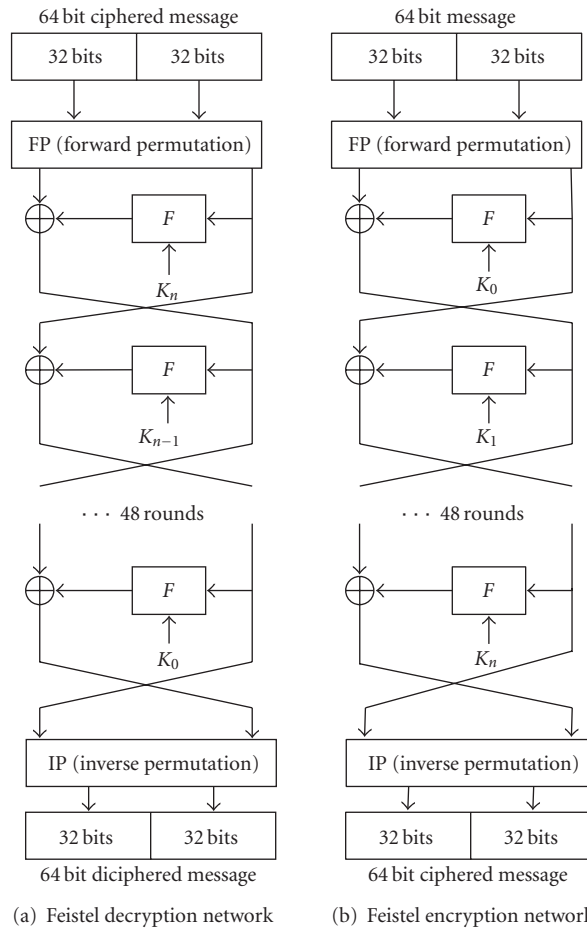


FIGURE 11: TDES encryption and decryption schemes (Feistel network).

The problem with ECB is that encrypting the same block with the same key gives an identical output. As a counter measure to revealing any information, blocks are chained together using different modes like Cipher Feed Back (CFB), Output Feed Back (OFB), and Cipher Block Chaining (CBC) illustrated in Figure 13.

The TDES represents a data parallel and pipelined signal processing application. So far TDES has been implemented as a hardware IP core [30–32]. To the best of our knowledge, no parallel software implementation on embedded multiprocessor has been reported.

7. Performance Evaluation and Comparison

Based the C implementation from National Institute of Standards and Technology (NIST) [29], the sequential TDES encryption C code consists of a Forward Permutation (FP), 48 calls to an F macro that executes both F boxes and the Feistel network, and finally there is an Inverse Permutation (IP), the C Code is a lookup table- (LUT-) based implementation with combined SPBox tables meaning all arithmetic operations are precalculated and stored in tables.

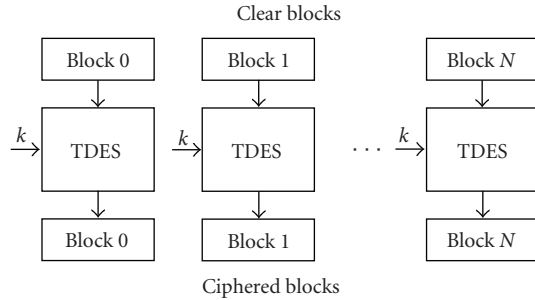


FIGURE 12: ECB operation mode for the TDES block cipher.

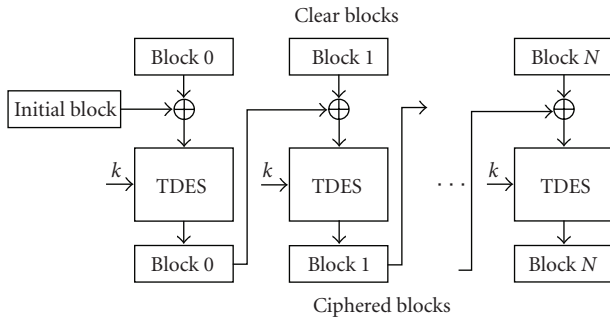


FIGURE 13: CBC operation mode for the TDES block cipher.

Two different implementation methods are compared on the 48-PE multiprocessor: multiple-data multipipeline software parallel implementation and coprocessor data parallel implementation.

7.1. Software Parallel

7.1.1. Fork-Join Model. To fully use the 48-PE multiprocessor, two parallelisms, data and task parallelism, are studied and combined together to achieve a best performance. Data parallelism means that different data are distributed across different processing nodes. These nodes process their received data in parallel. In task parallelism, the 48 calls to an F macro are cut averagely into several groups; each of these groups is mapped onto one MicroBlaze PE sequentially. Each MicroBlaze processor gets input data, calculates its mapped group of several F macros, and sends the results to the following MicroBlaze processor. In this way, the MicroBlaze processors mapped with all the 48 F macros work together as pipelined group.

The two parallelisms are combined together to work as a Fork-Join model showed in Figure 14. In order to measure the exact execution time and verify the result validity, two MicroBlaze processors are reserved as the source and destination; in a real case, the source and destination are the memory blocks, where original and worked data are stocked. The source MicroBlaze processor generates the original data, does the FP function, and sends one-by-one to the first MicroBlaze processors of different pipelined groups. Each pipelined group of MicroBlaze processor calculates their received data in parallel and finally sends the results to the

destination. The destination MicroBlaze processor gets the data from the last MicroBlaze processors of pipelined groups, does the IP function, and stocks them to the memory.

As described in the architecture section, the target architecture is a 48-PE multiprocessor organized as a 4×4 mesh of clusters fully interconnected through a network-on-chip (NoC). Each cluster includes 3 local MicroBlaze processors and 2 local memory blocks connected to an NoC switch. One local memory block is used for data communication between MicroBlaze processors and the other one is used for synchronization service between MicroBlaze processors in the same pipelined group. Separation of data and synchronization memory location will avoid unnecessary communication conflicts and improve system performance. MicroBlaze processors in the same cluster communicate using local memory blocks to minimize the communication time; in each memory block, addresses have been reserved for intercluster communication.

7.1.2. Results Example on 24 Cores. One mapping example result is given in Figure 15. Only 24 cores at most among 48 PEs are used for data and task mapping. As mentioned before, MicroBlaze 42 is used as source and MicroBlaze 24 as destination.

- (i) 24 MicroBlaze PEs are chosen for implementation.
- (ii) All data are divided into 4 blocks and the 24 MicroBlaze PEs are divided into 4 groups correspondingly.
- (iii) To encrypt each data block, each pipelined group has $24/4 = 6$ MicroBlaze PEs.
- (iv) In each pipelined group, each MicroBlaze PE will calculate $48/6 = 8 F$ macro calls.

At the first step, only one pipelined group is used to map the TDES application. Different size of data is sent by the source: from 10 packets up to 400 000 packets as in Figure 16. MicroBlaze processors in the pipelined group augment from 1 PE to 24 PEs at most. As the size of packets increasing, the average cycle to treat one packet of all the pipelined groups converges to a constant, which is in the fact the calculation time of one MicroBlaze processor in the pipeline ($48/N * \text{calculation time of } F$). The average cycle for one packet is almost the same between 4000 packets and 400 000 packets traffic cases. For the next step of experiment, in total 96 000 packets traffic is used for data and task parallelization, at most 24 pipelined group will get 4000 packets for each group. The observation of stable average cycle for one packet will ensure that the variety of packet number will not impact the result precision in the next step of experiment.

Another important observation is that small size of packets treated by large number pipelined group will use more time than the one treated by small number pipelined group, for an extreme example 10 packets treated by 24-PE pipeline uses almost the same time as 10 packets treated by 1-PE pipeline. So task parallelism is suitable for large-scale data application.

At the second step, task parallelism and data parallelism are combined to find out a good tradeoff between the

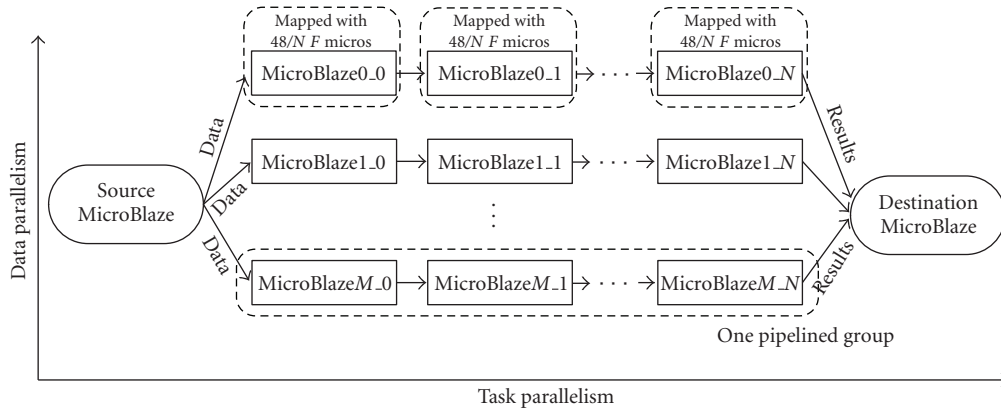


FIGURE 14: Fork-Join Model of data and task parallelism.

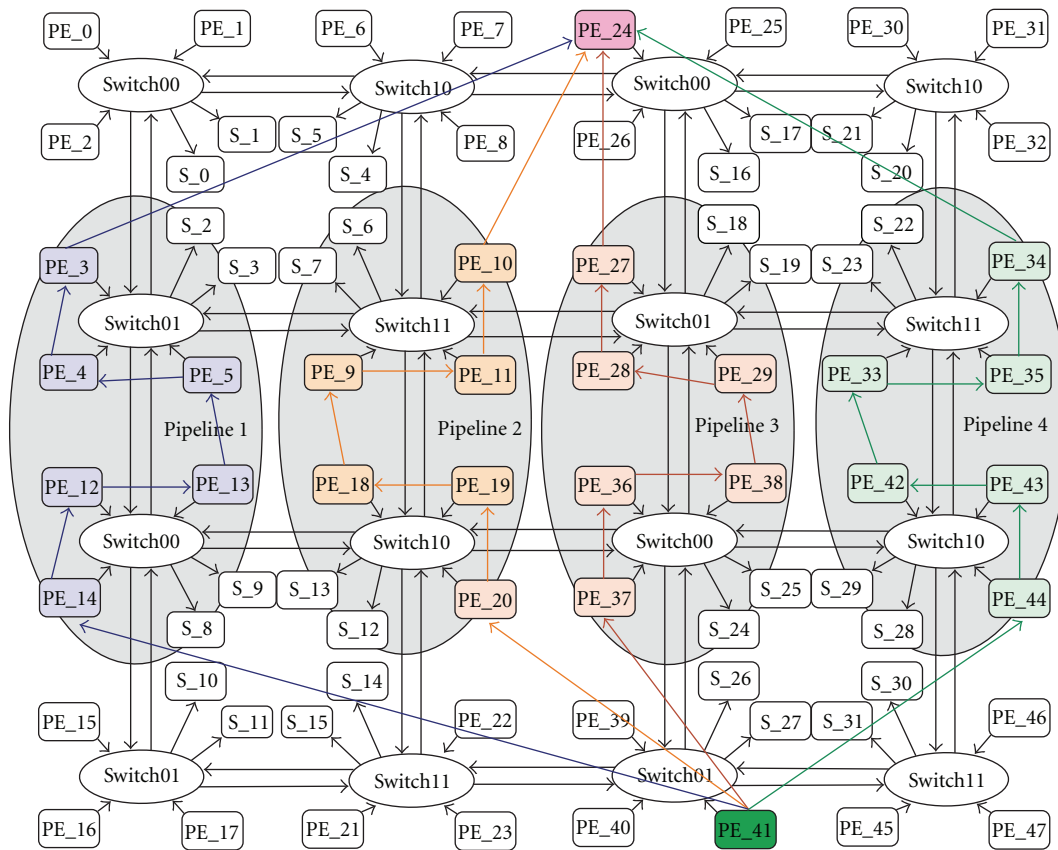


FIGURE 15: Fork-Join Model mapped to 48-PE multiprocessor.

2 parallelisms. In this example, at most 24 PEs are used, different combination of data and task parallelism is listed in Table 3.

Figure 17 shows the comparison between single pipelined group (only task parallelism) and multiple pipelined groups (task parallelism combined with data parallelism) both treating 96 000 packets. Single pipelined group means that only one pipeline with different number of PE is used to encrypt the 96 000 packets. Multiple pipelined groups fully use all the 24 PEs. Results show that multiple pipelined group mapping is much quicker than single pipelined group

mapping, which is obvious. And a large number PE (24 PEs) pipelined group are not a good choice. The results of the other 7 multiple pipelined groups are zoomed in Figure 18 to find out the best tradeoff between data and task parallelism.

In Figure 18, it is clear that the combination of 8 pipelined groups, each with 3 MicroBlaze processors per group, is the mapping of TDES with minimum execution time. So it is the best tradeoff between data and task parallelism. The principle of PE grouping is to make the most nearby MicroBlaze processor together, meaning that use inner-cluster MicroBlaze processor as most as possible,

TABLE 3: Combination of data and task parallelism (24 cores case).

Number of pipelined group	Number of PE in 1 pipelined group	Number of F micro mapped to 1 PE	Packets for 1 pipelined group
24	1	48	4000
12	2	24	8000
8	3	16	12 000
6	4	12	16 000
4	6	8	24 000
3	8	6	32 000
2	12	4	48 000
1	24	2	96 000

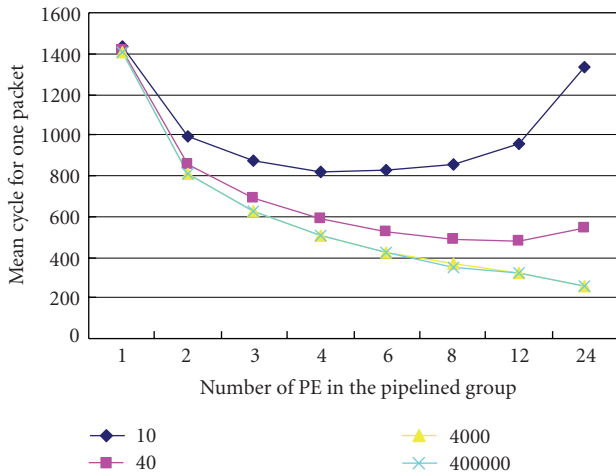


FIGURE 16: Results of one pipelined group with different size of data.

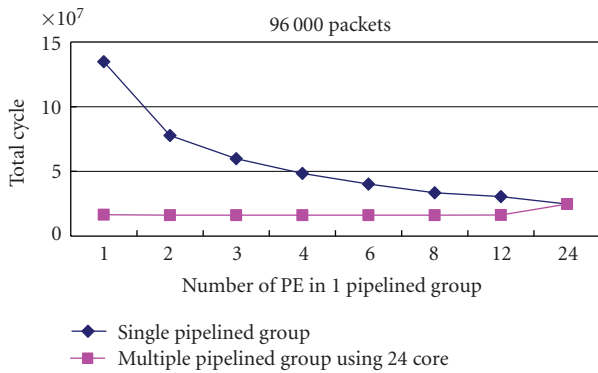


FIGURE 17: Single pipelined group versus multiple pipelined groups.

inter cluster MicroBlaze processor as close as possible. This principle can explain the best tradeoff combination of 8 pipelined groups each with 3 MicroBlaze processors.

Hardware performance monitoring network records the latency of each packet sent by the 24 MicroBlaze processors. The latency of packets is shown in Figure 19. The latency of both data and synchronization packets is important as it effects the pipelined execution. As the data and synchronization communication are separated and most nearby PE grouping principle is applied, there are not many routing conflicts on the NoC and the packets latency does not change

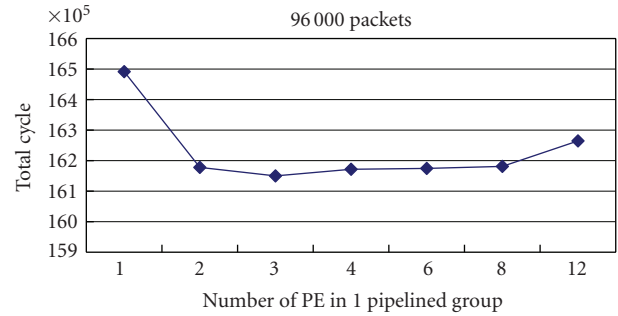


FIGURE 18: Tradeoff between task and data parallelism (24 core limited case).

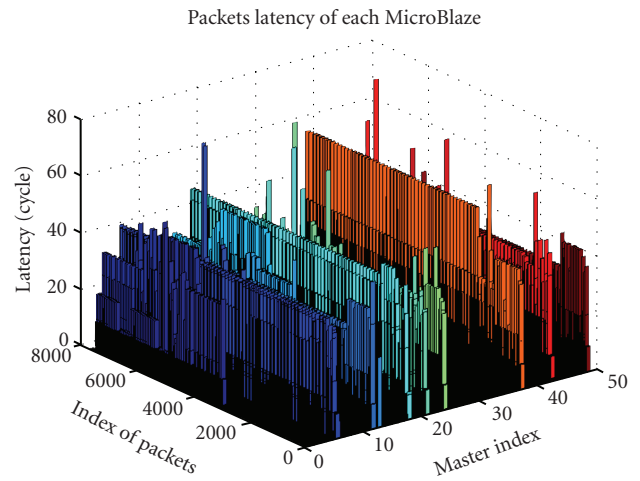


FIGURE 19: NoC monitoring.

too much; for the PEs in the middle of pipelined group, for example, processor 5, the packets latency changes little. Most of the conflicts are found at the connection from source processor 41 or destination processor 24, which is the bottleneck of system communication. From the performance results and real traffic monitoring, the Fork-Join model and pipelined data and task parallelization avoid the congestion problem of mesh topology. With hardware performance monitoring network, the system communication situation can be analyzed to find out the limits of system to easily improve performance.

TABLE 4: HLS-based TDES IP versus optimized IPs.

	Helicon	Xilinx	HLS (RAM)	HLS (LUT)
Slices	467	16181	2877	4183
Max frequency (MHz)	196	207	170	162
Throughput at 100 MHz	255.6 Mbps	6.43 Gbps	305 Mbps	305 Mbps



FIGURE 20: 5 Stage pipeline TDES.

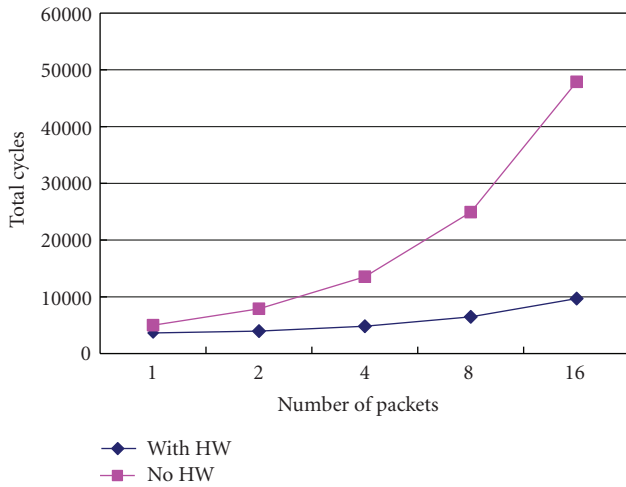


FIGURE 21: Parallel software versus coprocessors on a 48-PE multiprocessor.

7.2. Data Parallelism with Coprocessor. Data parallelism with coprocessor is another method to realize TDES application parallelization onto multiprocessor. Coprocessor is used to execute complex math operation, which can greatly improve system performance. In this case, each MicroBlaze processor of the 48-PE multiprocessor has a coprocessor to do the whole TDES functions; the MicroBlaze processor is only in charge of communication: they get the original data from the source, send them to coprocessor, wait until coprocessor sends back the results, and finally send the results back the destination.

ImpulseC tools are used to generate the TDES coprocessor directly from the C code to VHDL source, which greatly improves the design productivity.

7.2.1. Coprocessors Generation Using ImpulseC

(1) TDES Coprocessor Generation. The TDES coprocessor is designed for the Xilinx MicroBlaze processor using an FSL interface. The Triple DES IP was synthesized for a Virtex-4 platform by Xilinx ISE. The maximum frequency is reported by the synthesis. The generated 5-stage pipeline implementation uses 2877 slices and 12 RAM blocks with a maximum frequency of 169.75 MHz. The occupation for the same IP using LUTs instead of RAM blocks is 4183 slices.

The maximum frequency for this version is 162.20 MHz. 12 instances of the generated IP were successfully implemented on an Alpha Data XRC-4 platform (Virtex-4 FX-140) using 12 MicroBlaze processors within a Network-on-Chip.

The chosen architecture for the Triple-DES hardware accelerator is the 5-stage pipeline shown in Figure 20.

This IP was synthesized for a Xilinx Virtex-4 LX-200 FPGA. RAM blocks can be saved by using LUTs, if necessary. Table 4 compares the surface and performance of the generated IP with other commercial IPs. Xilinx's implementation [32] uses a fully pipelined architecture, which allows a very high throughput. But it is impossible to reach such a throughput on an NoC. Helion's architecture [31] uses a simple loop, which saves a lot of slices. The IP is generated by ImpulseC whereas Helion's is coded directly in VHDL/Verilog. This is the main reason why the IP generated by ImpulseC is not as efficient. The performances of these 2 commercial IPs are picked up from the documents to compare them with ImpulseC generated IP in Table 4.

(2) Parallel Software versus Coprocessors. In this case, all the 48 MicroBlaze processors do the TDES application in parallel. When they finish all the packets, they will send a finish flag to the synchronization memory. MicroBlaze 0 will verify that all the PEs finish their jobs and records the execution time. Results of TDES application using coprocessor on the 48-PE multiprocessor are shown in Figure 21. The results of software parallel are used to illustrate the acceleration with hardware coprocessor. It can improve system performance by 5 times.

8. Conclusion

High performance embedded applications based on image processing and single processing will increasingly be moved to embedded multiprocessors. An automatic design flow is proposed for data parallel and pipelined signal processing applications on embedded multiprocessor with NoC for cryptographic application TDES. The proposed flow explores through execution on multi-FPGA emulation for parallel software implementation with task placement exploration and task granularity analysis. A hardware-based network on chip monitoring drives the task placement process to reduce communication bottlenecks. In the second phase, high level synthesis generated hardware accelerators are added to explore the tradeoff in area performance while still privileging multiprocessor basis for the implementation. Future work will add reconfigurability management of dedicated area for hardware accelerators as well as automatic parallelization.

References

- [1] <http://www.itrs.net/>.
- [2] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, Boston, Mass, USA, 4th edition, 2006.
- [3] D. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, Boston, Mass, USA, 1999.
- [4] A. A. Jerraya and W. Wolf, *Multiprocessor Systems-on-Chip*, Morgan Kaufman, Boston, Mass, USA, 2004.
- [5] L. Benini and G. De Micheli, *Networks on Chips: Technology and Tools*, Morgan Kaufmann, Boston, Mass, USA, 2006.
- [6] M. Ito, T. Hattori, Y. Yoshida, et al., "An 8640 MIPS SoC with independent power-off control of 8 CPUs and 8 RAMs by an automatic parallelizing compiler," in *Proceedings of IEEE International Solid State Circuits Conference (ISSCC '08)*, pp. 90–598, San Francisco, Calif, USA, February 2008.
- [7] "ARM 11 MPCore," <http://www.arm.com/products/CPUs/ARM11MPCoreMultiprocessor.html>.
- [8] "MIPS32® 1004K™ Core," <http://www.mips.com/products/processors/32-64-bit-cores/mips32-1004k>.
- [9] S. Shibahara, M. Takada, T. Kamei, et al., "SH-X3: SuperH multi-core for embedded systems," in *Proceedings of the 19th Symposium on High Performance Chips (Hot Chips '07)*, Stanford, Calif, USA, August 2007.
- [10] "Texas Instruments Multicore Fact Sheet SC-07175".
- [11] "Texas Instruments TMS320C6474 Multicore DSP SPRS552," October 2008.
- [12] "Texas Instruments TMS320VC5441 Fixed-Point DSP data manual SPRS122E," October 2008.
- [13] "QorIQ™ P4080 Communications Processor," <http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=0162468rH3bTdG25E4>.
- [14] T. Miyamori, "Venezia: a scalable multicore subsystem for multimedia applications," in *Proceedings of the 8th International Forum on Application-Specific Multi-Processor SoC*, Aachen, Germany, June 2008.
- [15] Y. Ahn, K. Han, G. Lee, et al., "SoCDAL: system-on-chip design accelerator," *ACM Transactions on Design Automation of Electronic Systems*, vol. 13, no. 1, pp. 17.1–17.38, 2008.
- [16] H. Nikolov, M. Thompson, T. Stefanov, et al., "Daedalus: toward composable multimedia MP-SoC design," in *Proceedings of the 45th Design Automation Conference (DAC '08)*, pp. 574–579, Anaheim, Calif, USA, June 2008.
- [17] C. Haubelt, T. Schlichter, J. Keinert, and M. Meredith, "SystemCoDesigner: automatic design space exploration and rapid prototyping from behavioral models," in *Proceedings of the 45th Design Automation Conference (DAC '08)*, pp. 580–585, Anaheim, Calif, USA, June 2008.
- [18] R. Ben Mouhoub and O. Hammami, "MOCDEX: multiprocessor on chip multiobjective design space exploration with direct execution," *EURASIP Journal of Embedded Systems*, vol. 2006, Article ID 54074, 14 pages, 2006.
- [19] <http://www.eve-team.com/>.
- [20] <http://www.xilinx.com/>.
- [21] <http://www.artemis.com/>.
- [22] <http://www.impulsec.com/>.
- [23] M. O. Cheema, L. Lacassagne, and O. Hammami, "System-platforms-based systemC TLM design of image processing chains for embedded applications," *EURASIP Journal of Embedded Systems*, vol. 2007, Article ID 71043, 14 pages, 2007.
- [24] O. Hammami, Z. Wang, V. Fresse, and D. Houzet, "A case study: quantitative evaluation of C-based high-level synthesis systems," *EURASIP Journal of Embedded Systems*, vol. 2008, Article ID 685128, 13 pages, 2008.
- [25] Z. Wang and O. Hammami, "C-based hardware-accelerator coprocessing for SOC an quantitative area-performance evaluation," in *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems (ICECS '08)*, pp. 522–525, Saint Julians, Malta, August-September 2008.
- [26] "Open core protocol specification, release 2.2," OCP-International Partnership, 2008.
- [27] FIPS 46-3, "Data Encryption Standard (DES)," National Institute of Standards and Technology, October 1999.
- [28] FIPS 197, "Advanced Encryption Standard (AES)," National Institute of Standards and Technology, November 2001.
- [29] "Recommendation for block cipher modes of operation-methods and techniques," NIST Special Publication 800-38A 2001 Edition.
- [30] P. Kitsos, S. Goudevenos, and O. Koufopavlou, "VLSI implementations of the triple-DES block cipher," in *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS '03)*, vol. 1, pp. 76–79, Shariqah, United Arab Emirates, December 2003.
- [31] "High performance DES and triple-DES cores for Xilinx FPGA," Helion Technology, 2003.
- [32] V. Pasham and S. Trimberger, "High-speed DES and triple-DES encryptor-decryptor," Xilinx Inc., August 2001.