TIME-PREDICTABLE SOFTWARE-DEFINED ARCHITECTURE WITH SDF-BASED COMPILER FLOW FOR 5G BASEBAND PROCESSING

Vanchinathan Venkataramani[⋆] *Bruno Bodin*[⋆†] *Aditi Kulkarni*[⋆] *Tulika Mitra*[⋆] *Li-Shiuan Peh*[⋆]

[⋆] School of Computing, National University of Singapore
[†]Yale-NUS College, National University of Singapore
{*vvanchi, bruno, aditi, tulika, peh*}*@comp.nus.edu.sg*

ABSTRACT

The advent of 5G networks motivates the need for high-performance, low-power, time-predictable hardware that can handle the aggressive real-time latency and throughput requirements of baseband processing. With newer generations like 5G, programmable hardware that can adapt readily to network specification updates becomes a critical requirement. We introduce a software-defined array-based many-core architecture, called SPECTRUM, that couples lightweight predictable hardware components with a compiler flow that orchestrates the on-chip hardware resources. This design, by construction, provides timing guarantees with a programmable architecture. Our architecture and compiler flow are designed to support basestation baseband processing computation represented using deterministic Synchronous Data Flow (SDF) model of computation. SDF is commonly used to represent signal processing applications and fits well with real-time systems requirements. We demonstrate substantial power savings with SPECTRUM compared to existing DSPs while meeting the performance requirements.

Index Terms— Time-predictability, many-core architecture, LTE/5G baseband processing, synchronous dataflow.

1. INTRODUCTION

The evolution towards 5G cellular network technology has brought considerable challenges in signal processing both at the user equipment and at the base stations. 5G is envisioned to achieve at least 10x throughput improvement compared to the 4G standards and sets exact latency requirements. While previous generations like Long Term Evolution (LTE) are designed to handle primarily downlink communications with mobile smartphones, 5G is expected to handle multiple use-cases: massive Machine Type Communication (mMTC), enhanced Mobile Broadband (eMBB) and Ultra-Reliable and Low Latency Communication (URLLC) [1]. Application scenarios that require URLLC include remote surgery and driverless cars, while mMTC is fueled by industrial automation and Internet of Things (IoT) [2, 3]. Uplink communications with guaranteed latency and throughput are crucial as these use-cases demand continuous data upload. We present an array-based processing architecture and the associated compiler flow for 5G baseband processing in the base stations.

Current base station deployments utilize custom DSP cores in conjunction with ASIC hardware accelerators to meet the performance requirements of baseband processing [4, 5]. However, these domain-specific architectures are hard to program, inflexible to evolving standards, and incur enormous Non-Recurring Engineering (NRE) costs [6]. Though FPGAs are flexible, their limited performance, energy-efficiency, and to some extent programmability challenges make them a less attractive alternative. Prior attempts towards baseband processing on GPU [7] or general purpose processor [8] were unable to provide real-time latency guarantees as the underlying hardware is composed of components with inherently unpredictable timing such as processors (with complex dynamic scheduling of instructions and speculation in out-of-order execution), hardware-managed caches, and networks-on-chip with dynamic routing. Additionally, the high Thermal Design Power (TDP) of these architectures in the order of hundreds of watts contributes to huge energy operating costs and prevents their usage in base station deployments [9]. These shortcomings of the existing architectures have generated increasing interest in novel alternative architectural designs that are flexible to changes, meet computation demands, and provide real-time guarantees.

We propose SPECTRUM [10], an array-based many-core architecture, targeted to meet the requirements of the uplink baseband processing at the base station by exploiting inherent thread-level parallelism of the computation. SPECTRUM is comprised of an array of lightweight processing elements, memories and networks that are fully exposed to the software for configuration. The high computational demands are satisfied by deploying a large number of simple in-order cores in parallel, each equipped with additional custom instructions [11] carefully chosen to accelerate the frequent computational patterns in baseband processing workloads. These simple cores with custom instructions are paired with on-chip per-core software-controlled scratchpad memories (SPM) [12] that are both amenable to easy software timing

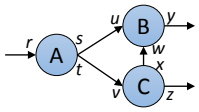

Fig. 1. Synchronous Data Flow (SDF) graph example.

analysis [13]. The communications among the array tiles are orchestrated by a completely software-scheduled, lightweight Network-on-Chip (NoC) leading to timing predictability of the entire SPECTRUM architecture. At the same time, the simplicity of the hardware enables low-power operations keeping the TDP of the design to only a few watts.

The key to the success of SPECTRUM's software-defined architecture is the ability of the compiler to efficiently map diverse signal processing workloads on the platform. The toolchain accepts as input a specification of the array-based architecture including the array size, the SPM size, the NoC connection among the array tiles etc. The signal processing application (in this case the 5G baseband processing application) is specified in the form of a Synchronous Data Flow (SDF) [14] model of computation. The compiler takes care of the mapping of the computation and data as well as the scheduling of the communications. Experimental evaluations show that the software toolchain maps the uplink baseband processing application successfully, meets the throughput and latency requirements, and achieves 2.39x lower power than existing DSP cores based design with hardware accelerators.

2. SYNCHRONOUS DATA FLOW (SDF) FOR BASEBAND PROCESSING

Digital Signal Processing (DSP) applications typically contain a set of cooperating tasks that execute periodically. The Synchronous Data Flow (SDF) [14] model of computation suitably represent these applications. Static analysis of the SDF can generate a deterministic schedule of the tasks [15], making SDF a preferred model especially in real-time systems. A number of DSP and multimedia applications have been represented as SDF [16, 17].

Figure 1 shows an example of an SDF. The actors or vertices (A, B, C) in the SDF graph correspond to the computational kernels of the application. The edges are communication buffers between the actors. Buffers are labeled with the number of data packets (known as tokens in the SDF parlance) produced and consumed by each actor $(r, s, \ldots, z)$. An actor is fired (executed) when all the input data tokens required (values specified in the arrowhead of each incoming edge) are received in its incoming buffers. Once an actor is fired, it consumes the input data tokens and produces appropriate number of output data tokens denoted by the arrow tail of each outgoing edge. For example, in Figure 1, when actor A fires it consumes r packets and produces s, t packets that in turn will be consumed by the actors B and C, respectively.

Figure 2 shows the SDF corresponding to uplink baseband processing [8]. In uplink baseband processing, independently encoded data, called streams or layers (LAY) are transmitted by multiple User Equipments (UE) such as mobile or IoT devices. The radio receiver at the base station obtains combined signals from multiple UEs via receiver antennae (ANT), filters them, removes cyclic prefix, and performs FFT to convert back the signal from time to frequency domain. In the frequency domain, user extraction recovers individual UE signals and baseband processing is initiated. The baseband processing per UE has three main components: *Channel estimation*, *Data Demodulation*, and *Decoding*. It is customary to use ASICs for *Decoding* to achieve required performance [18] as these have fixed algorithms that rarely change.

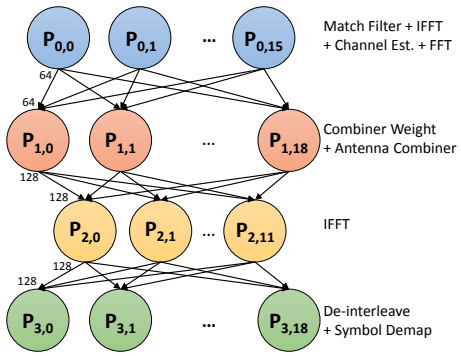

Fig. 2. SDF model of uplink baseband processing for 4x4 MIMO, 64 QAM, 1200 Subcarriers and 10 users.

Multiple actors executing the same computation but operating on different data in parallel define a phase in the application. We divide baseband processing into four phases consisting of 16, 19, 12 and 19 actors, respectively, to exploit task-level parallelism. We describe how an SDF application is mapped onto our SPECTRUM architecture later in Section 4. The execution time of each actor is obtained by Worst-case Execution Time (WCET) computation [19] (includes memory access latency) on the in-order core deployed in SPECTRUM.

3. SPECTRUM ARCHITECTURE

SPECTRUM is a time-predictable software-defined many-core architecture intended for streaming DSP applications including uplink baseband processing with real-time latency and throughput requirements. The input data flows through the chip and gets processed by the cores but never spills to the off-chip memory. In conventional shared memory multi-core architectures, hardware autonomously moves data between on-chip cache and off-chip memory at runtime creating uncertainty in execution time at the application level. Additionally, the NoC suffers from timing unpredictability arising from dynamic routing apart from increased traffic due to coherence messages. Finally, significant overheads and non-deterministic execution are caused by synchronization in parallel processing.

Figure 3 presents the high-level view of SPECTRUM architecture that employs software-defined, time-predictable

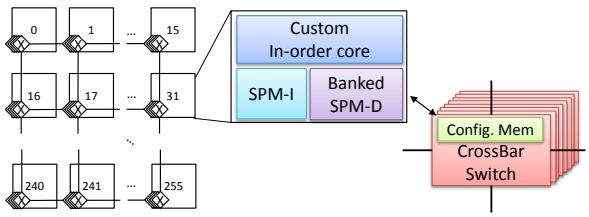

Fig. 3. High-level view of SPECTRUM architecture.

designs in all components. The current prototype consists of 256 tiles arranged in a 16x16 2D mesh connected by 8 switches per tile. We explain the different components and how time-predictability is achieved in SPECTRUM below.

Memory: We utilize software-controlled Scratchpad Memory (SPM) as the on-chip memory for instructions and data. SPM is explicitly managed by the compiler [12, 20] or programmer (using domain knowledge). Thus, for a given mapping, the access latency is deterministic. We statically map the entire kernel code in the instruction SPM as we have sufficient capacity. Next, we obtain the data requirements for the worst-case input. The private/stack variables are mapped in each tasks local SPM. As the shared variables within a phase for this application have only read-only accesses, we create separate copies in the data SPM of each core accessing a shared variable.

Compute: The in-order cores include one-off customization for accelerating common computational patterns in baseband processing using specialized functional units for power-performance efficiency. We execute one task per core on the single-issue in-order cores. Additionally, the data and instruction accesses have deterministic latency as explained earlier. Thus, we are able to obtain the WCET of each computation kernel accurately and precisely using [13, 19].

Communication: The bufferless software-controlled NoC routers are essentially comprised of just a crossbar switch, whose connections are set according to the configuration memory associated with each switch each cycle. Since there is no runtime routing or flow control logic, we need to set the configuration memory ensuring that the packets reach the correct destination without encountering any network link contention along the way. We obtain the switch configurations assuming worst-case baseband processing traffic using the SPECTRUM compiler flow explained in Section 4. As the router behavior is determined statically to handle all possible workloads, we ensure time-predictability.

Thus, SPECTRUM is designed to handle the worst-case workload, ensuring that the latency and throughput constraints are always satisfied.

4. SDF-BASED COMPILER TOOLCHAIN

SPECTRUM is supported by a software toolchain that automates task/NoC scheduling. Our earlier toolchain proposed in [10] was specific to the uplink baseband processing application. The four different phases were executed in a pipelined fashion with the number of tasks (per phase) determined to match the performance requirements [21]. All tasks within a phase synchronize on the system clock after computation, then the data packets are sent to destination tasks using precomputed NoC switch configurations. Finally, the system throughput was decided based on the critical phase of the pipeline. This pipelined nature leads to system under-utilization as the phases exhibit different execution times.

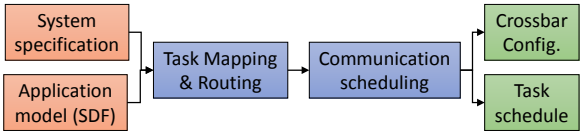

Fig. 4. SDF-based compiler toolchain for task and communication scheduling.

Instead, we propose a generic compiler toolchain that takes as input the baseband processing application code, its SDF representation, and a specification of the architecture. The toolchain then generates the task mapping and the communication schedule (Figure 4). SDF-based scheduling can capture execution time variations better as it does not require synchronized execution of the phases; the tasks can execute independently and concurrently as long as the data dependencies are satisfied.

In our toolchain shown in Figure 4, the task-to-core mapping and communication routes are iteratively obtained for each SDF actor, similar to the approach proposed in [22]. As multiple paths exist between the source and destination cores, we chose the route with the minimum routing cost, defined as a weighted sum of (i) the latency between communicating tasks and (ii) the number of shared links.

Once the task mapping and communication routes are set, we need to schedule packet transfers, i.e., determine a valid time to send the packets. The SDF model can be used to generate a finite list of task execution and precedence constraints known as the expansion graph [23]. This information is sufficient to produce a valid schedule that ensures that link contentions do not happen and the data dependencies between tasks are met. We model this problem using an Integer Linear Programming formulation with minimization of total execution time as the objective where task executions are constrained by data dependencies and link utilization.

Finally, the packet sending time obtained from solving the ILP formulation is utilized to create the switch configurations and added to the software code. At run-time, the configuration memory is automatically loaded ensuring that the packet transfers happen accurately.

5. EXPERIMENTAL EVALUATION

In 5G, multiple use-cases with different workload specifications and/or performance requirements are also expected to be handled by the same base station [1]. The specifica-

tions for massive Machine Type Communication (mMTC), enhanced Mobile Broadband (eMBB) and Ultra-Reliable and Low Latency Communication (URLLC) can differ in terms of MIMO (1x1, 2x2 or 8x8) as well as modulation scheme (QPSK, 64QAM, 256QAM). For each of these use-cases, if the workload processing time does not meet this deadline, it is dropped leading to substantial packet loss that cannot provide the ultra-reliable connectivity. Thus, we statically map the different use-cases onto isolated regions of the SPECTRUM chip automatically using our compile toolchain, such that the latency and throughput requirements can be successfully met for the worst-case workload.

Benefit of the SDF compiler toolchain: When compared to the previous work on SPECTRUM [10], the SDF toolchain improves system utilization. The benefit comes mainly from the fine-grained, interleaved schedule of task executions and communications, and thus a reduced total execution time of the application. This compact schedule enables using a 800MHz clock frequency instead of 900MHz obtained previously (Table 1). Thus, application mapped using the SDF compiler toolchain on SPECTRUM achieves 1.13x power reduction. Detailed explanation and evaluation of various aspects of SPECTRUM can be found in [10].

Comparison with Existing Hardware Architectures: In this work, we present evaluation for eMBB use-case with 4x4 MIMO and 64QAM (worst-case workload with highest data throughput of 233.6 Mbps [21]), as the power/performance numbers on proprietary hardware platforms were only available for this specification. For SPECTRUM, we obtain application performance using modified *gem5* cycle-accurate micro-architectural simulator [24] and power consumption from RTL synthesis. Table 1 states the performance of SPECTRUM, Intel Xeon Gold 6126 (Skylake-SP) many-core CPU and DSP+accelerator platform [5, 25]. Skylake-SP experiences significant drop rate of 33% with average power of 215W due to timing unpredictability and thermal throttling of the chip due to the intensive workload. Though DSP+accelerator platform [25] meets the performance requirements, they rely heavily on custom hardware accelerators and also consume 2.39x higher power than SPECTRUM.

Table 1. Power-performance comparison for 4x4 MIMO

	SPECTRUM [10]	SPECTRUM + SDF Toolchain	C66x DSP Cores + Accelerators	Skylake-SP
Drops	0%	0%	0%	33%
Avg. Power	6.1 W	5.4 W	12.9 W	215 W

RTL Design, Synthesis and Layout: The SPECTRUM architecture has been implemented in RTL. Figure 5(a) shows our floorplan comprising 256 tiles arranged in a 16x16 grid. Each tile (Figure 5(b)) consists of an Amber25 processor core [26], 16KB SPM for instruction, 32KB SPM for data and 8 software-scheduled network switches where two networks are connected to each of the four SPM banks of data. Both the Amber core and the NoC switches are synthesized, carried to layout, with support for up to 1.5 GHz system frequency. A SPECTRUM tile consumes 82.1 mW power at 800MHz on commercial 40nm process with power breakdown as: Data SPM 58%, Instruction SPM 33%, Core 4%, NoC Switches 3%, and others 2%.

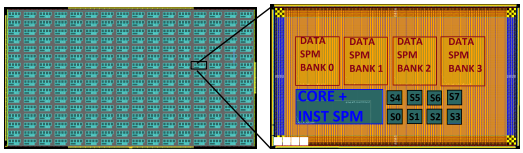

Fig. 5. *SPECTRUM* [10] (a) 16x16 floorplan (b) Tile layout (Area per tile is 1.4mm x 0.7mm)

6. RELATED WORK

There are existing time predictable single-core (e.g., PRET [27]) and many-core architectures (e.g., Kalray [28], T-CREST [29]). LTE-specific architectures have been stated earlier in Section 1. Among many-core designs, Kalray [28] requires sophisticated software timing analysis using network calculus for its interconnect. T-CREST [29] contains caches in addition to SPM supported by software-based coherence leading to difficulty in programmability. SPECTRUM is designed with scalable predictable light-weight components that are easy to program including task, data placement and NoC communication simplifying timing analysis.

Software-scheduled NoCs have been proposed for efficiency and quality of service. The AEthereal NoC [30] realizes timing guarantees with slot tables that implement time-division multiplexing (TDM) across different traffic flows. SPECTRUM, with its domain-specific knowledge, can employ fully software-scheduled NoCs.

A number of frameworks have been developed to compile dataflow models and perform task mapping on many-cores and heterogeneous architectures [16, 31]. In this work we additionally optimize for communications to obtain the configuration of the programmable routers.

7. CONCLUSION

We proposed a sophisticated SDF-based compiler toolchain that automatically maps the LTE/5G baseband processing application on SPECTRUM, a time-predictable array-based many-core architecture. We show that our toolchain for SPECTRUM meets the real-time performance guarantees of the baseband processing and delivers 2.39x power reduction with respect to DSP cores with ASIC hardware accelerators.

8. ACKNOWLEDGMENT

This work was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Industry-IHL Partnership Grant NRF2015-IIP003.

9. REFERENCES

[1] ITU, "Setting the Scene for 5G: Opportunities & Challenges," 2018, https://bit.ly/2MO2Swv.

[2] Imtiaz Parvez, Ali Rahmati, Ismail Guvenc, Arif I Sarwat, and Huaiyu Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, 2018.

[3] Philipp Schulz, Maximilian Matthe, Henrik Klessig, et al., "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," 2017.

[4] "Alcatel-Lucent 9926 digital 2U eNodeB baseband unit," Alcatel-lucent product brief, 2009.

[5] Raguram Damodaran, Timothy Anderson, Sanjive Agarwala, et al., "A 1.25 ghz 0.8 w c66x dsp core in 40nm cmos," in *VLSID*, 2012.

[6] Huawei, "Base station operation increases the efficiency of network construction," https://bit.ly/2GtCd6N.

[7] Qi Zheng, Yajing Chen, , et al., "Using graphics processing units in an lte base station," in *JSPS 2015*.

[8] Magnus Själander, Sally A McKee, Peter Brauer, et al., "An lte uplink receiver phy benchmark and subframe-based power management," in *IEEE ISPASS*, 2012.

[9] "Temperature Control Solution of Communication Base Station," 2011, https://bit.ly/2Bpa9jH.

[10] Vanchinathan Venkataramani, Aditi Kulkarni, Tulika Mitra, et al., "Spectrum: A software defined predictable many-core architecture for lte baseband processing," in *LCTES 2019*.

[11] Pan Yu and Tulika Mitra, "Scalable custom instructions identification for instruction-set extensible processors," in *CASES 2004*.

[12] Vanchinathan Venkataramani, Mun Choon Chan, and Tulika Mitra, "Scratchpad-memory management for multi-threaded applications on many-core architectures," in *TECS 2019*.

[13] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, et al., "The worst-case execution-time problemoverview of methods and survey of tools," in *TECS 2008*.

[14] Edward A Lee and David G Messerschmitt, "Synchronous data flow," in *Proceedings of the IEEE 1987*.

[15] Edward Ashford Lee and David G Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE TC 1987*.

[16] José Luis Pino, Soonhoi Ha, Edward A Lee, et al., "Software synthesis for DSP using Ptolemy," in *Journal of VLSI signal processing systems for signal, image and video technology 1995*.

[17] Sander Stuijk, Marc Geilen, and Twan Basten, "SDFˆ 3: SDF for free," in *ACSD 2006*.

[18] Sandro Belfanti, Christoph Roth, Michael Gautschi, et al., "A 1Gbps LTE-advanced turbo-decoder ASIC in 65nm CMOS," in *VLSIC*, 2013.

[19] Xianfeng Li, Yun Liang, Tulika Mitra, et al., "Chronos: A timing analyzer for embedded software," *Science of Computer Programming*, 2007.

[20] Vanchinathan Venkataramani, Anuj Pathania, and Tulika Mitra, "Unified thread- and data-mapping for multi-threaded multi-phase applications on spm many-cores," in *DATE 2020*.

[21] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," Technical Specification (TS) 36.213, 3rd Generation Partnership Project (3GPP), 03 2017, Version 14.2.0.

[22] Stephen Friedman, Allan Carroll, Brian Van Essen, et al., "SPR: an architecture-adaptive CGRA mapping tool," in *FPGA 2009*.

[23] Sundararajan Sriram et al., *Embedded Multiprocessors: Scheduling and Synchronization*, 2nd edition, 2009.

[24] Nathan Binkert et al., "The gem5 simulator," *ACM SIGARCH 2011*.

[25] Silexica, "Multi-core software design for an lte base station, white paper," 2016, https://bit.ly/2TyE7sx.

[26] "Amber ARM-Compatible Core," 2010, https://bit.ly/32DNRWO.

[27] Stephen A Edwards and Edward A Lee, "The case for the precision timed (pret) machine," in *DAC*, 2007.

[28] B. D. de Dinechin, "Consolidating High-Integrity, High-Performance, and Cyber-Security Functions on a Many-core," in *DAC 2019*.

[29] Martin Schoeberl, Sahar Abbaspour, Benny Akesson, et al., "T-crest: Time-predictable multi-core architecture for embedded systems," *JSA*, 2015.

[30] Kees Goossens, John Dielissen, and Andrei Radulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE D & T of Computers*, 2005.

[31] Maxime Pelcat, Karol Desnos, Julien Heulot, et al., "Preesm: A dataflow-based rapid prototyping framework for simplifying multicore dsp programming," in *EDERC 2014*.