Scatter-Gather DMA Performance Analysis within an SoC-based Control System for Trapped-Ion Quantum Computing

Tiamike Dudley[1,2], *Member, IEEE*, Jim Plusquellic[1], *Member, IEEE*, Eirini Eleni Tsiropoulou[1], *Senior Member, IEEE*, Joshua Goldberg[2], Daniel Stick[2], and Daniel Lobser[2]
[1]Dept. of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM, USA
[2]Sandia National Laboratories, Albuquerque, NM, USA

***Abstract*—Scatter-gather dynamic-memory-access (SG-DMA) is utilized in applications that require high bandwidth and low latency data transfers between memory and peripherals, where data blocks, described using buffer descriptors (BDs), are distributed throughout the memory system. The data transfer organization and requirements of a Trapped-Ion Quantum Computer (TIQC) possess characteristics similar to those targeted by SG-DMA. In particular, the ion qubits in a TIQC are manipulated by applying control sequences consisting primarily of modulated laser pulses. These optical pulses are defined by parameters that are (re)configured by the electrical control system. Variations in the operating environment and equipment make it necessary to create and run a wide range of control sequence permutations, which can be well represented as BD regions distributed across the main memory. In this paper, we experimentally evaluate the latency and throughput of SG-DMA on Xilinx radiofrequency SoC (RFSoC) devices under a variety of BD and payload sizes as a means of determining the benefits and limitations of an RFSoC system architecture for TIQC applications.**

***Index Terms*—trapped-ion, qubits, quantum computing, SoC-based FPGA control system**

I. Introduction

A trapped-ion quantum computer (TIQC) uses modulated optical [1] or RF/microwave [2] pulses to precisely control the quantum state of its trapped-ion qubits[1]. The sequence of pulses applied to an ion is referred to as a gate sequence, where gates represent the logic operations that together form a quantum algorithm. The magnitude, phase, and frequency of the laser pulses are commonly controlled using optical modulators with radiofrequency (RF) signal inputs, which can be generated using benchtop instruments like arbitrary waveform generators (AWGs) or direct-digital synthesizers (DDS), or using high-speed digital-to-analog converters (DACs) embedded within a device like a radiofrequency System-on-Chip (RFSoC). The programmable logic (PL) component of the RFSoC is capable of emulating AWG and DDS functionality, while providing a highly diverse mechanism to configure gate sequence pre-processing pipelines.

An important constraint to fully leveraging the flexibility of the PL for TIQC applications is ensuring the sufficient speed of data transfers between components of the memory hierarchy within the RFSoC system architecture. This was measured at a general level between the real-time processor (RPU) and PL on the Xilinx ZCU111 board in [3], but here we expand on previous work by analyzing communication in the context of scatter-gather dynamic-memory-access (SG-DMA). As described in this paper, SG-DMA provides a path to making practical TIQC control systems that have the flexibility to define pulses on-the-fly with latencies that do not cause significant delays in quantum algorithms, most importantly quantum error correction [4]. While errors must be decoded in real time, corrections may be applied as virtual Pauli-frame updates or as physical gates [5], the latter of which requires low-latency feedback to limit runtime impact. In addition to quantum error correction and related implementations of conditional operations based on mid-circuit measurements [6], repetitive recalibration and drift control [7] also motivate a control system capable of low-latency pulse reconfiguration.

The RF signals used to define quantum gates in a TIQC are derived directly from the gate sequence data. Therefore, SG-DMA performance is the limiting factor in meeting the timing requirements associated with synchronization and phase coherence, and in implementing feedback-based calibration within the TIQC system. An important overall goal of our system is to arbitrarily reconfigure gate parameters based on measurements made on feedback channels with a baseline latency that is on the order of the shortest gate time, $\approx 1\,\mu$s for a single-qubit gate [8]. This way the reconfiguration time would not

This work is supported by a collaboration between the US DOE and other Agencies. This material is based upon work supported by Quantum Systems through Entangled Science and Engineering through National Science Foundation Quantum Leap Challenge Institutes under Grant OMA-2016244. This material is also based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Quantum Systems Accelerator, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research Quantum Testbed Program.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

[1]Though both optical and RF/microwave pulses have been used to perform high fidelity gates, throughout this paper we will usually refer to them as laser pulses, which are used in the QSCOUT sytem [1]. However the same control system applies to both pulse types.

add substantially to the overall runtime. Although adjustments to the pulse parameters occur at longer time scales within a calibration loop [3], a high-speed data streaming interface is required to meet data delivery requirements associated with the inside loop.

The latency and throughput requirements for the memory-mapped DRAM to PL streaming interface implemented by the SG-DMA engine depends on several factors, including the size of the gate sequence data packets, the duration and complexity of the laser pulses and the periodic stalls needed to update pulse parameters based on calibration measurements. Although it is necessary for the SG-DMA engine to accommodate continuously changing gate sequences, it is common that a particular sequence is repeated multiple times before an update to the pulse parameters is necessary. The proposed control system incorporates several optimizations that reduce bandwidth in these cases.

The relationship between measured performance characteristics and TIQC system requirements is interleaved with the presentation of the main results of this work, which are:

- A performance analysis of a multi-processor system that utilizes APU, RPU, and PL. Performance trade-offs associated with using different architectural components for SG-DMA setup and operation are discussed, along with their respective strengths and weaknesses.
- A latency and throughput analysis (including statistical characterization) is carried out over a wide range of SG-DMA parameters, including BD ring size and BD block size. Architectural features impacting performance as they relate to SG-DMA parameters are discussed.

II. Related Work

Previous work investigated the use of SG-DMA for systems with similar requirements, characterized as high bandwidth transfers of data with different payload sizes and locations to and from main memory. The system described in [9] uses a customized DMA engine to transfer three-dimensional blocks of data for hyperspectral imaging applications. The CubeDMA system operates on a contiguous cube-shaped memory region and is capable of computing data addresses in real time. SG-DMA, in contrast, manages data transfers using a separate data structure, allowing transfers of discontiguous regions of memory at the cost of additional memory accesses corresponding to the reading and writing of the SG-DMA data structure.

The systems proposed in [10] and [11] measure and mitigate memory interference by running PS and PL benchmarks in parallel to reduce slowdowns caused by memory contention and inefficiencies related to DDR access. The former uses Controlled Memory Request Injection (CMRI) to increase the utilization of the DDR. The latter system uses scheduling policies within the PL to reduce memory stall times.

The work in [12] presents an analysis of the throughput and latency of AXI port configurations for the ZCU102 and Ultra-96 boards. Their analysis considers AXI bus width, burst size, memory chip configuration, access patterns, and transaction frequency. An analysis of DMA engine performance using the ZCU102 is presented in [13]. Their work investigated the performance impact of DMA parameters including burst and memory stride sizes.

The FPGA-based qubit control systems proposed in [14], [15], [16] and [17] utilize low latency, high-bandwidth PL-side on-chip memory and simplified memory addressing. The authors of [18], [19] use off-chip DDR to store waveform data and/or parameters to greatly increase the amount of gate data the system can create and access on-the-fly.

The performance penalty associated with off-chip storage of gate sequence data is investigated in this work. We utilize the ZCU111 for performance characterization of SG-DMA operating in memory-mapped-to-streaming (MM2S) mode, and a multi-processor system architecture consisting of an APU, RPU, and the PL to investigate the performance impact of various SG-DMA parameters to represent different gate sequence structures.

III. Communication

In this section, the communication architecture as well as the relevant performance measurements of the RFSoC system architecture are described in the context of a TIQC system.

A. System Architecture

The memory hierarchy associated with the processor side of the Xilinx ZCU111 RFSoC used in this work is layered in a configuration commonly used in high-end microprocessors, with two 4 GB DDR4 (DRAM) defining the largest and slowest layer of the hierarchy, and processor caches representing the smaller and fastest layers. The RFSoC architecture additionally includes PL-side block RAM (BRAM/UltraRAM), lookup-table (LUT) RAM, and distributed RAM.

The large size of the DRAM components can be leveraged to accommodate a wide range of gate sequence definitions, which are configured in the proposed architecture by application processing units (APUs). A real-time processing unit (RPU) is charged with transferring APU-pre-configured gate sequences to the PL-side processing components by issuing commands to a dynamic memory access (DMA) engine, configured in SG mode. SG mode utilizes a set of buffer descriptors (BDs), arranged in a BD ring, with each pointing to an arbitrary memory region within the DRAM. The primary focus of this paper is on evaluating SG-DMA performance characteristics, i.e., latency and throughput, as a function of several BD ring parameters, including the number of BDs, their buffer size, and their location in DDR.

A block diagram showing the processing and interconnect components within the Zynq UltraScale+ RFSoC on the ZCU111 board is shown in Fig. 1. The SoC consists of a dual-core Cortex R5 RPU, a quad-core Arm Cortex A53 APU, a PL region, and two external DDR4 DRAM memories, interconnected through a complex point-to-point network of Arm Advanced eXtensible Interface (AXI) switches. The DDR memories are accessible to both RPU and APU for system instantiations in which the memory map for both DRAMs are configured with overlapping address spaces. Otherwise, the PL DRAM is not accessible by the 32-bit RPU when address-mapped above the 4 GB PS DRAM. Each core in the RPU

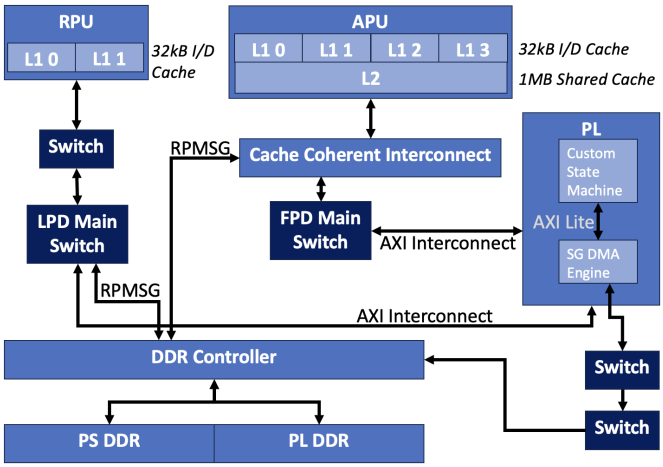


Fig. 1. Zynq MPSoC system architecture showing the microprocessors, programmable logic, DDR, and AXI channels that are used.

possesses a 32 kB L1 instruction and data cache, while each core in the APU possesses a 32 kB L1 instruction and data cache and a shared 1 MB L2 cache.

The inclusion of the processors and DDR enable the control system to network, compile gate sequences, and share a memory space with the PL without consuming PL resources. The gate sequence compilation activities are complex and are best handled within the context of software execution on the processor, which provides a higher level of abstraction. The PL, on the other hand, is capable of providing clock-cycle-specific delivery of gate sequence data to the modulators, which is required to maintain a coherent TIQC system. On-going work is focused on adding a quantum compiler with feedback [20], [21], [22] within the software framework running on the APU. Therefore, the tasks and requirements of a TIQC system are an ideal fit to the co-design-based approach proposed in this work.

The Vivado block diagram tool is used to create a system level diagram which utilizes the APU, RPU, block RAM (BRAM), DMA, GPIO, and PL DDR. The DDR is created using the memory interface generator (MIG), and is interconnected through AXI to other components of the system architecture. In the following experiments, a portion of the PL DRAM is mapped into the address space of the APU and RPU at an aperture given by 0x80000000 to 0x9FFFFFFF. This makes the PS and PL DDR accessible by both the APU and RPU. An AXI Smart-connect block is used to access the PL DDR, which is capable of arbitrating between multiple master IP interfaces.

The DMA engine possesses several input-output interfaces. The configuration and control channel of the DMA engine is connected to an AXI-lite interconnect and is controlled by the RPU in our experimental setup. The memory-mapped interfaces of the DMA engine connect to the PL DDR while the streaming interfaces are connected to a custom state machine (CSM) (described below in III-C). The CSM stores incoming data from the MM2S slave interface to a BRAM, and writes BRAM out-going data to the streaming-to-memory-mapped (S2MM) master interface. Two GPIO IP blocks are instantiated and are accessible by the APU, RPU, and SM. Although not shown in Fig. 1, a Linux kernel is created that enables the RPU and APU to communicate using an API defined by the libmetal and OpenAMP (*asymmetric* multiprocessing) standards called RPMsg. RPMsg utilizes on-chip tightly-coupled memory for code, stack, heap, etc., and the PL DDR for inter-processor communication.

B. Scatter Gather DMA (SG-DMA)

The DMA IP block provided by Xilinx [23] is implemented entirely in the PL, and can be configured to operate in either simple mode or scatter gather (SG) mode. In previous work, we investigated the latency and throughput of simple mode across a range of data transfer widths and PL clock frequencies [3]. This work presents a statistical characterization of latency and throughput with DMA configured in SG mode.

Unlike simple mode, SG mode allows data blocks to be distributed across the DDR at non-sequential addresses. Data blocks are described using BDs, which are stored in a ring as a linked-list data structure. Each BD is 64 bytes in size and contains a 64-bit base address, a 26-bit size field (for up to 64 MB payload sizes), a pointer to the next BD and several control and status fields. An example of a BD ring is shown in Fig. 2.

The proposed TIQC system uses the APU to construct gate sequences in PS and/or PL DDR. The linked list data structure associated with the BD ring provides a fast and flexible mechanism for creating heterogeneous gate sequence orderings in real time during system operation, that can be tuned based on feedback from instrument sensors. After the creation of each custom sequence by the APU, the base address of the BD ring is transferred to the RPU, using either RPMsg or GPIO. The RPU is tasked with configuring the SG-DMA engine with the base address and then starting the SG-DMA engine, which occurs when the tail descriptor is transferred to the SG-DMA engine. SG-DMA configuration registers are controlled using a 32-bit AXI4-lite [23] data bus between RPU and the SG-DMA engine. Given the tight timing requirements of TIQC systems, this type of dual APU-RPU architecture enhances performance characteristics over single-CPU or soft-core alternatives by providing parallelism and lower latency through the use of a dedicated processor optimized to respond to real-time events.

The RFSoC resources used to implement several SG-DMA configurations are given in Table 1, which were obtained by synthesizing a Vivado block diagram containing only the MPSoC, dual-channel SG-DMA, AXI Interconnect, AXI SmartConnect, and Processor System Reset IP blocks. This set of IP blocks represents the minimum set required to implement the SG-DMA engine. Although bus-width is configurable up to 1024 bits, at the expense of additional PL resources, the 256-bit version is the best match to the spline-based data path specification defined within our TIQC system. The SG-DMA engine instantiated in our experiments is configured with a 256-bit data bus, which enables one complete spline segment to be encoded and transmitted as one word (spline encoding and other artifacts are described in Section IV).

Data Width (b)	LUTs Utilized	% of total	LUTRAM Utilized	% of total	FFs Utilized	% of total	BRAM Pages Utilized	% of total
32	10489	2.47	1970	0.92	15523	1.83	3 (90kb)	0.28
64	10639	2.50	1967	0.92	15792	1.86	5 (162kb)	0.46
256	13598	3.20	2695	1.26	20033	2.36	9 (306kb)	0.83
512	16120	3.79	4158	1.95	25420	2.99	9 (306kb)	0.83
1024	22864	5.38	7109	3.33	35543	4.18	9 (306kb)	0.83

TABLE I
UTILIZATION OF A DUAL-CHANNEL SG DMA ON THE ZCU111 RFSOC AS A FUNCTION OF DATA BUS WIDTH. A MINIMAL FUNCTIONAL IMPLEMENTATION REQUIRES AN INSTANCE OF THE MPSOC, AXI INTERCONNECT, AXI SMARTCONNECT AND THE PROCESSOR SYSTEM RESET IP BLOCK.

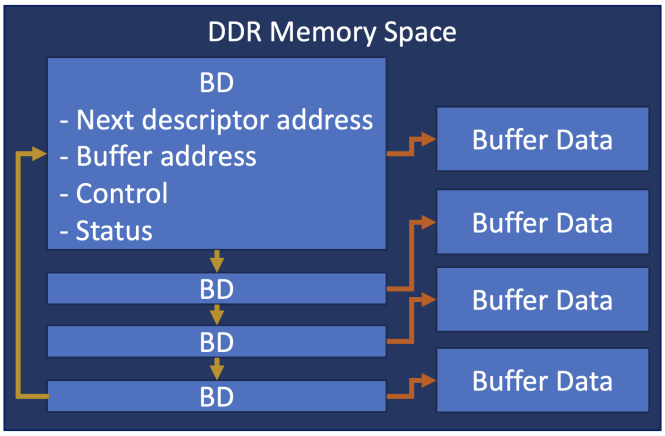


Fig. 2. Structure of BD ring in DDR.

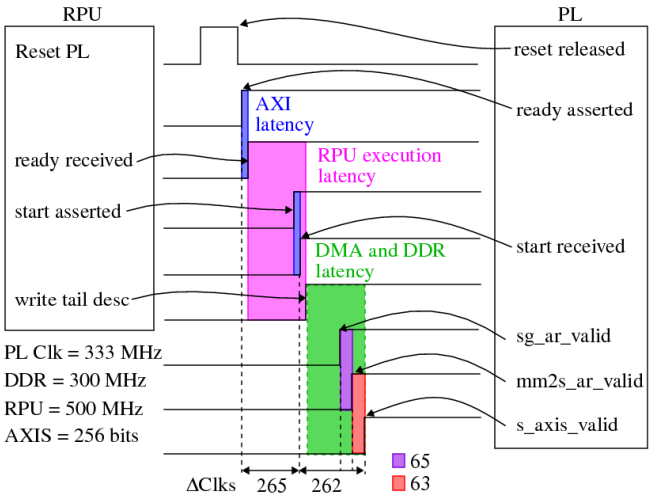


Fig. 3. Latency components of the MM2S channel of SG-DMA associated with the first iteration. The PL side clock frequency is 333 MHz, and the DMA engine configured with a bus width of 256-bits.

C. Customized State Machine (CSM)

The latency and throughput characteristics of the TIQC system architecture is measured using a customized state machine (CSM). An illustrative example showing the types of timing information that can be measured using the CSM is shown in Fig. 3. This example shows the interaction of the RPU with the SG-DMA and CSM components in the PL, and the corresponding counter values recorded within the CSM. The counter values reflect the latency of two components of the data transfer operation, namely 1) execution time of the RPU and interactions across the GPIO interface associated with starting the CSM and 2) latency of the SG-DMA measured between the start event and the arrival of the first word on the streaming interface in the PL. The sequence of operations are shown from top-to-bottom, while time (measured in clock cycles) is shown along the horizontal axis. The PL clock frequency in this example is 333 MHz.

The data transfer operation begins when the RPU issues a reset to the CSM through the GPIO and the first of two counters begins to increment after the release of the reset. The RPU blocks and waits for the PL to assert a GPIO ready signal. Once received, the RPU asserts a GPIO start signal. The PL blocks (and increments the first counter) until the start signal is received. Once received, the first counter is stopped and the second counter is started. The reported 265 clock cycles accounts for the delays of the GPIO signal transmissions and RPU execution time. The PL then blocks and waits for the first word to be delivered through the streaming interface. This second time interval is broken down further by routing several AXI signals, namely *sg_ar_valid*, *mm2s_ar_valid*, and *s_axis*, from the MM2S SG-DMA bus to the CSM. These signals reflect the instant in time when the SG-DMA issues the BD address to DDR, the time instance when the SG-DMA engine issues the buffer address to DDR and when PL is notified that the first word from the BD buffer is available on the streaming interface, respectively.

The reported values are the mean counter values associated with this RPU-bus-PL transaction. We report the mean, minimum, and maximum values converted to nanoseconds in the reminder of this paper. For example, the second counter value of 262 is equivalent to 787 ns of latency using a PL frequency of 333 MHz.

D. Experiment Overview

The objective of this analysis is to determine the performance characteristics, i.e., the average, best, and worst case data transfer characteristics, of a multi-processor architecture consisting of APU, RPU, and PL components. The partitioning of the tasks, namely, gate sequence management, BD ring creation, control, and execution, across these three components, enables the system to leverage the inherent parallelism that exists in the RFSoC platform. The following analysis reports on the performance characteristics and elaborates on the speed-limiting components as they relate to the timing constraints of a TIQC system.

The interaction between the RPU and PL components, i.e., the SG-DMA engine and CSM, defines the inner loop of

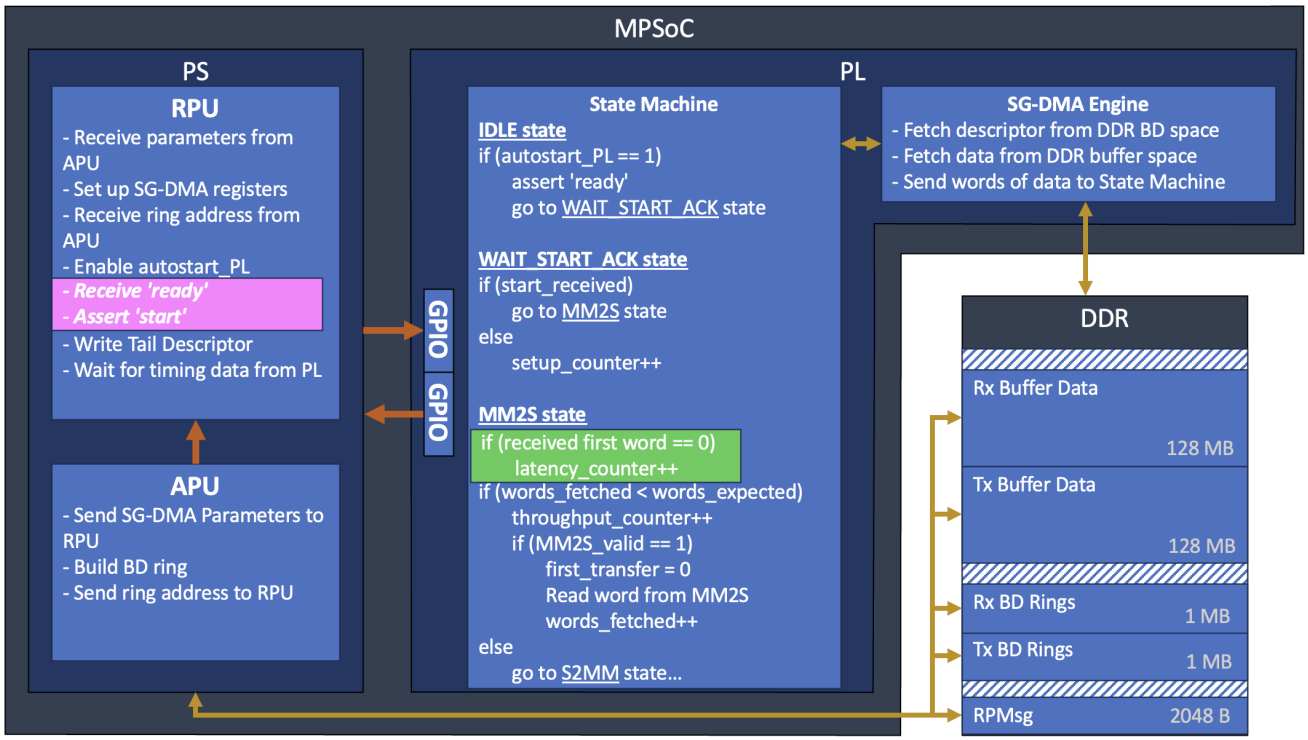


Fig. 4. Multiprocessor SG-DMA setup and state machine processes. The green and pink regions represent the timing regions in Fig 3. The pink timing region can be wrapped around any code running on APU or RPU to measure PS side execution time. Note that the RPU may build the BD ring in experiments measuring RPU execution time.

the TIQC system, and therefore possesses the tightest timing constraints. The analysis of this subsystem in section IV explores parameters including the size of the BD ring, i.e., number of BDs and the size of the buffer regions associated with each BD (payload size). We also examine the impact of sequential versus random address locations of the BD payloads in PL DDR as a means of determining whether the DDR memory controller provides any look-ahead optimizations.

Section IV also contains an analysis of the performance of the BD ring configuration process. The mean setup time and variability, i.e., minimum and maximum, are reported for different ring configurations. The performance of the BD ring creation process when executed on the APU is compared with the time taken when the same task is executed on the RPU, while accounting for the synchronization and communication latencies that are present when the APU creates the rings and hands-off to the RPU to schedule execution with the SG-DMA and CSM engines.

A block diagram illustrating the tasks and communication channels utilized in the experiments performed is shown in Fig. 4. The PL side remains the same across all experiments and consists of a CSM and SG-DMA engine. The timing events shown on the right side of Fig. 3 are associated with the IDLE, WAIT_START_ACK, and MM2S states in the CSM. The three counters in the CSM, namely, *setup_counter*, *latency_counter*, and *throughput_counter* are used to measure the corresponding performance characteristics of the data transfer operations. All components in the PL run at the maximum allowable frequency of 333 MHz. The PL DDR memory interface controller is also implemented in the PL and runs at 300 MHz.

The CSM is connected to the SG-DMA engine component through a 256-bit wide streaming AXIS interface. The bandwidth of this interface determines the shortest average gate time that can be continuously streamed. RPMsg utilizes shared memory in the PL DDR to enable data to be transferred between the APU and RPU. Alternatively, the APU and RPU can use the GPIO registers to exchange data, and given the GPIO are implemented in the PL, they are also accessible by the CSM, providing a three-way communication and synchronization channel between APU, RPU, and the CSM. The GPIO register can also be used within the CSM to obtain clock-cycle accurate timing measurements of communication latency between APU and RPU. However, the bandwidth of the APU-RPU-CSM communication and synchronization channel does not have a significant impact on overall system performance. This is true because control information related to the BD ring base addresses is represented using a small number of words and updates occur at relatively slow rates, in comparison to gate sequence transfer operations occurring between DDR and the SG-DMA engine. Therefore, the choice of using RPMsg or GPIO is driven by other factors, including the extent of the address space accessible by the RPU, resource utilization, ease of update by the APU, etc. Given these constraints, RPMsg is better suited as a communication mechanism, and is used in our system architecture for this purpose, while GPIO is used only for control and access to timing measurements.

The left side of Fig. 4 shows the tasks carried out by the RPU and APU under the scenario where the APU creates the rings and defines the contents of the buffers, while the

RPU only handles the BD ring scheduling operations for execution by the SG-DMA engine. As shown, once the BD ring is created, the APU passes the base address of the ring to the RPU. The process of creating the BD ring data structure involves the following steps:

- APU: Allocate memory space for BD ring and buffer data
- APU: Write sequence data into the buffer data region
- APU: Create BD ring linked list with buffer address and status information, e.g., first BD and last BD, etc.
- APU: Perform sanity checks on ring, flush cache, transfer base address to RPU
- RPU: Configure SG-DMA control registers
- RPU: Write to SG-DMA tail descriptor register

The last step, which starts the DMA transfer operation, is always performed by the RPU in our experiments. The RPU is also tasked with controlling the timing measurements which are carried out within the CSM. Once the transfer operation is complete, the CSM notifies the RPU and transfers the counter values through the GPIO register interface. The RPU forwards the counter values to the APU using RPMsg and then to a host for off-line analysis.

Note that carrying out DMA in SG mode requires the DMA engine to access DDR three times, once to fetch the BD, a second time to fetch the buffer data, and a third time to update the BD status fields. Therefore, SG-DMA has a larger overhead than simple mode, and corresponding performance penalty. However, SG mode offers several advantages, including the ability to quickly construct customized gate sequences that leverage pre-computed gate sequences distributed across the DDR memory space.

Moreover, the utilization of DDR itself, in contrast to smaller embedded memory resources, e.g., PL BRAM, enables additional scalability and flexibility in the TIQC control system. The proposed system can easily leverage technology improvements to DDR transfer speed and capacity as a means of improving the throughput of the control system as well as the number and variety of gate sequences that the control system can access.

E. Memory Map of the Experimental System

The PS and PL DDRs are utilized by the APU, RPU, and SG-DMA engine, with the APU running Linux on top of a virtual, paged memory system, while RPU and SG-DMA access physical memory directly, with the former running a bare metal application. As indicated, RPMsg also uses a shared DDR memory region for data exchange and synchronization.

The organization of the DDR memories utilized in the experiments performed is shown on the right side of Fig. 4. The physical memory accessed by RPU and SG-DMA is excluded from the paged, virtual memory system of the APU by adding a reserved memory region to the Linux device tree in the 4 GB aperture at the address range between 0x80000000 and 0x9FFFFFFF. The APU constructs BD rings and defines BD buffers in this 512 MB region. Given the aperture is located within the 32-bit address space of the RPU, this strategy enables performance comparisons related to BD ring creation to be made between the APU and RPU. A more attractive arrangement in which the PL DDR is memory-mapped above the PS DDR would allow a TIQC system to fully utilize the 8 GB of DDR, but would only allow updates to the PL DDR by the APU.

IV. Experimental Results

SG mode provides a great deal of flexibility in creating a complex sequence of data transfer operations. The user controls several BD ring parameters including the number, size, and location of distinct buffer spaces. In our experiments, we investigate a range of values related to the number of BDs transferred, from 2^0 to 2^{13} BDs and the size in bytes of each BD, from 2^5 to 2^{13}. Given the maximum clock frequency of the PL fabric is 333 MHz, it follows that the maximum achievable bandwidth is 10.6 GB/s.

In order to fully explore the transfer characteristics, several additional experiments were carried out. An SG-DMA mode referred to as *cyclic mode* is configured to evaluate the impact of repeatedly transferring the same BD buffer data, in contrast to the above experiments where distinct data is referenced in each BD. In all evaluations, no difference in the latency and throughput were observed between transfers using *n*-cycles in *cyclic mode* and an equivalent size BD ring in *non-cyclic mode*, so the performance statistics associated with *cyclic mode* are omitted. In a last set of experiments, the performance of the BD ring creation process is evaluated when carried out by the RPU, and compared with the performance of the APU.

A. Application to TIQC

Control pulses in our TIQC system are variable in size. Pulses can incorporate spline-based parameter modulation which supports either continuous or discrete updates for waveform parameters. Specialized gate definitions may incorporate various degrees of modulation on one or more waveform parameters for a variety of compensated gate schemes or for bandwidth limiting (such as with Gaussian amplitude profiles). Each waveform parameter takes 32B words, which represents a spline segment (even for square pulses, to keep the architecture consistent). Each channel supports 8 waveform parameters, amplitude, frequency, phase, and virtual phase, for two independent tones.

A simple square pulse requires at least one 32B word for all 64 endpoints, giving a minimum of 2kB per control pulse. However, duplicate parameter data, which typically arises for parameters that are unused and equal to zero, can be sent to multiple endpoints with a single word. For a simple (timed) NOP sent to all 64 endpoints (8 channels, with 8 parameters each), the only non-zero value is duration, and data is identical across all endpoints and can be represented with a single word. The 4GB DDR4 used in this work has the capacity for on-the-order of 10^7 unique simple gates. For a square pulse sent to a single channel, unique values will usually be used for all parameters except one phase and one virtual phase, and this can be represented as 8 32B words in total, or 256B. Some

gates require multiple concatenated control pulses, for example those used for dynamical decoupling (e.g. BB1 or SK1 gates).

The mapping of BDs to control pulses and gates depends on the software implementation. There are varying degrees of granularity that can be imposed, in which a BD could be used to represent a circuit, a subset of a circuit, a single gate, a pulse within a gate, or a subset of a pulse. Pulses comprise multiple independent data words for individual waveform parameters, e.g. frequency, phase, amplitude, and a virtual phase. For a gate which shares multiple parameter values, such as X and Y gates which only differ by phase, shared data can be packed into a single buffer and the different phases into a second buffer. This way, updates to the shared data for a gate can be made in a single buffer and any BD that references that buffer is instantly updated. For simplicity, we are assuming that a single step in a circuit is captured by a BD. This single BD scenario represents individual gates and multiple gates run in parallel, and also captures NOP data that is sent to unused channels. It can also be used to pad out extra time if gates run in parallel have different total durations.

Here we consider a realistic scenario where raw data is directly streamed via SG-DMA and each BD is assigned to a particular gate. Fig. 5 portrays the throughput of a BD ring as a function of buffer length, plotted with transfer time in microseconds against the number of spline words. For example, a BD ring composed of 256 spline words (8192B), i.e. 32B per spline word, requires a payload transfer time of 0.77 μs .

The system architecture is set up to allow broadcasts of identical data to multiple endpoints, and constrains the smallest usable payload size to 32B, or 1 word. Common quantum gate usage scenarios include:

- Timed wait with all parameters set to 0: 1 word transfers.
- Implementation of an Rz gate on a single channel: 3 unique words (non-zero virtual phase, zero values for remaining parameters on target channel, zero values for all parameters on all other channels).
- Implementation of a square pulse gate on a single channel (assuming a co-propagating Raman pulse): typically takes 8 words (2 amplitudes, 2 frequencies, 1 non-zero phase, and 1 virtual phase for AC Stark shift, 1 zero-valued word that is broadcast for the other phase and virtual phase, and 1 zero-valued NOP to pad the other channels).
- Implementation of multiple square pulse gates on all channels in parallel with unique parameters for each channel: Typically takes 56 words (64 if both phases and virtual phases are used).
- Implementation of a square pulse with a single sinusoidal period of amplitude modulation where deviations from a perfect sinusoid are below the resolution of the DACs: Requires 28 words (22 amplitude words, a fixed amplitude word, 2 frequencies, 1 non-zero phase, 1 non-zero virtual phase (neglecting time-dependent Stark shifts), 1 NOP word for unused phase and virtual phase, and 1 NOP for all other channels).

All of these gates can have durations under ∼250 ns, despite the variation in the data size and throughput, which is still well below the target limit for 1 μs gates, and in most cases

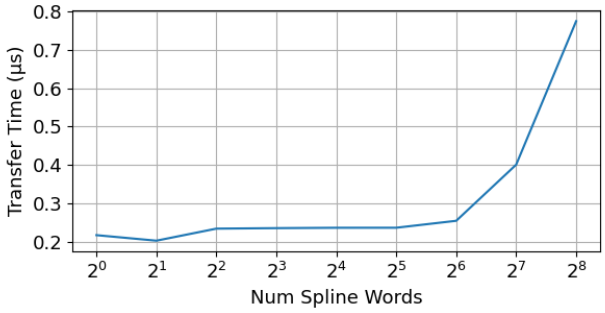


Fig. 5. Throughput measurements of the SG-DMA engine.

the minimum gate time (for a constant feed) is independent of the number of data words unless a significant amount of modulation is required during a short pulse. For the sinusoidal amplitude modulation, a bandwidth of roughly 4 MHz can be achieved. These sequences can be run back-to-back for a long continuous sine wave without impacting per-sequence transfer time. Moreover, packing more data into a single gate will only improve throughput, meaning back to back pulses with the same duration can be used to further improve throughtput.

If multiple non-identical sinusoidal amplitude modulation gates are needed in parallel on all channels, then 27*8=216 words are required, approaching 750 ns and a bandwidth of 1.3 MHz for constantly fed data. However, amplitude modulation bandwidth requirements are typically <100 kHz for simultaneously addressing multiple modes, and is not typically applied to all channels. If higher-order harmonics of motional modes are required, it may be desirable to drive amplitude sidebands on 4 tones, which requires a 2.5 MHz bandwidth. This is slightly above the typical radial secular mode frequencies for ytterbium ions, and 2nd harmonics can be driven by further detuning of the frequency inputs. Depending on the amount of data that needs to be fed where higher bandwidth is needed, these limits can be locally exceeded if the circuit is punctuated with slower gates or gates without the same data requirements since the FIFOs can be packed while a slower gate is being executed. Individual spline engine FIFO depths are 256, with another 256 depth FIFO that feeds a broadcaster for each channel (so the FIFO depth is effectively 512 in cases where one parameter is heavily modulated and the others are fixed).

The resources provided by a large DRAM are leveraged in our system architecture to satisfy this wide range of gate sequence configurations, and SG mode of DMA enables gate sequence definitions of various lengths and complexity to be dynamically configured and updated on-the-fly by the APU in parallel with gate sequence execution. In particular, the PL DRAM is mapped to an address space above the lowest 32-bit aperture used by the PS DRAM. The reasons for this are twofold: it allows full utilization of the entire PS DRAM (which is 4 GB on the ZCU111/ZCU102) and PL DRAM address space, where PL DRAM is strictly dedicated to hosting gate data and associated BDs. Second, the 64-bit address space of the APU can read and write the PL DRAM directly. Note, however, that the RPU is restricted to a 32-bit address

space, and therefore transfers between PL and PS DRAM (via CDMA) are required to enable the RPU with full access. As an alternative, the RPU could be charged with only configuring the SG-DMA engine with 64-bit base addresses of BD rings, which enables the SG-DMA engine to access any portion of PS or PL DRAMs.

The computational complexity associated with implementing gate sequences on-the-fly using feedback channel information, and the high throughput, low latency requirements of the modulated laser pulse control system is best implemented as a co-design-based system architecture, as discussed earlier. Architectures that attempt to implement the entire system as state machine logic would result in over-utilization of the PL-side resources and would make the tasks of configurability and maintainability much more difficult, and would significantly limit scalability.

As proposed, nearly all of the complexity of creating BD rings is performed by the APU, which hosts a compiler to convert quantum assembly down to raw pulse instructions, mainly to avoid network overhead for feeding back on gate definitions that often include non-linear parametrization of calibration parameters and may employ spline-based modulation. While the details of the compiler are provided in [22], two key capabilities of the compiler are:

- The ability to express high-level functional definitions of gates that employ more complicated mathematical representations, as well as methods to efficiently fit splines with parameters that exceed 32 bits.
- The creation and management of persistent metadata associated with gates and circuits for updating gates in-situ during the state preparation stage.

Fitting splines requires matrix methods, in which the number of spline knots affects matrix size, and this (to our knowledge) cannot be implemented efficiently in PL without limiting the number of spline knots supported in a pulse or requiring significant resource utilization in the PL fabric. Moreover, spline fitting requires some non-trivial elements for mapping into efficient PL side interpolators. While we do support fast register-based updates to control pulse parameters if they do not incorporate spline-based modulation, the methodology does not encapsulate the full complexity that might arise. For instance, changes in pulse amplitude will likely yield different AC Stark shifts, which are typically corrected using a spline-modulated virtual phase. However, the relationship between pulse amplitude and virtual phase is not trivial because of non-linearities in the AOMs and amplifiers. Moreover, the actual light shift can change day to day because of carbon buildup on the AOMs, which is a common issue when using high-power UV lasers. A key take-away here is that these kinds of problems cannot be easily addressed by a PL-based system.

B. SG-DMA Performance Analysis

Latency is defined in our experiments as the interval of time associated with the writing of the tail descriptor and the arrival of the first word on the AXI MM2S streaming interface in the CSM. An example is shown in Fig. 3, where the second interval labeled 262 (787 ns) between events *start received* and *s_axis_valid* represents latency. Multiple repeated experiments are performed to obtain the median, minimum, and maximum values of latency in our experiments. Similarly, the median, minimum, and maximum values of throughput are reported in MB/sec, where throughput is computed by excluding the latency of the first transfer. Non-parametric statistical metrics, i.e., median, minimum and maximum, are used instead of the mean and standard deviation because the statistical characteristics of the measurements are not Gaussian.

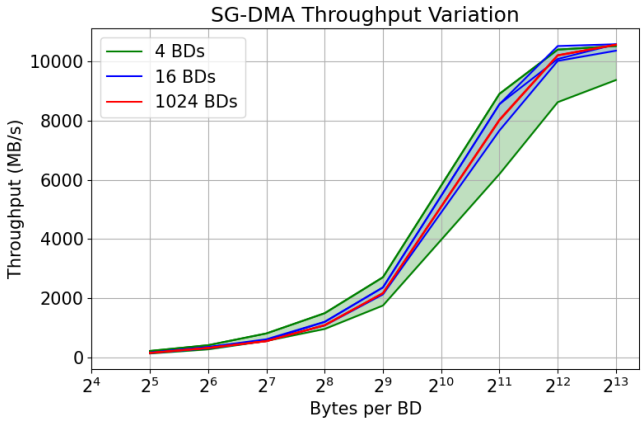


Fig. 6. SG-DMA throughput as a function of the number of bytes per BD and BD ring size. The shaded regions highlight the variability and median throughput of the SG-DMA engine decreasing for larger BD rings.

Our measurements reveal that the latencies associated with the set of RPU-to-SG-DMA experiments are independent of the BD-ring parameters. In particular, the statistical characteristics of the latency measurements are well characterized using a minimum latency of 850 ns, a median latency of 988 ns, and a maximum latency of 1516 ns. Variations in latency are largely due to stalls in the fetch-decode-execute-writeback sequence carried out by the RPU, which includes delays introduced by access contention and refresh cycles in the DDR, cache misses, and AXI bus contention. For example, the DDR4 device used on the ZCU111 (MT40A512M16JY-075E) specifies that a refresh row stall is required every 7.8 μs. We observed DDR4 row refresh stalls using Xilinx ChipScope, which showed stalls occurring for approximately 210 ns in duration, and asynchronously with program execution.

The corresponding results for throughput are shown in Fig. 6, with bytes per BD plotted along the x-axis against throughput in MB/s on the y-axis. Unlike latency, throughput is impacted by the BD ring size. The graph superimposes the throughput results for BD rings of size 4, 16 and 1024. The variability in throughput is largest for the BD ring of size 4, and decreases linearly as the total amount of data transferred increases. Given the DDR stalls occur at regular time intervals, the impact that they have on the throughput associated with larger rings is amortized over the longer data transfer time interval, reducing their impact on variability. Moreover, the negative performance impact of thrashing between DDR memory locations corresponding to the BD ring and buffer addresses also becomes a smaller fraction of the

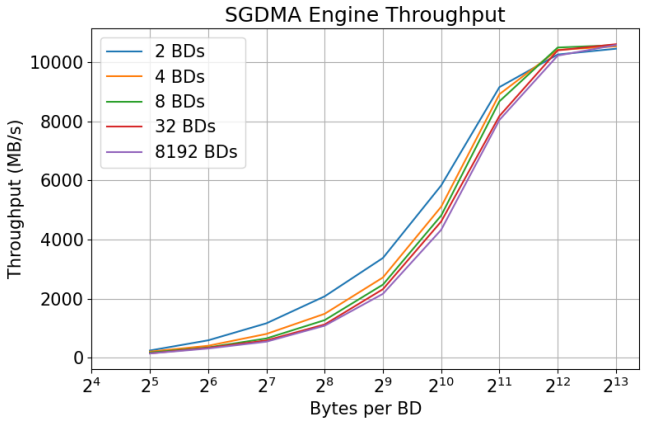


Fig. 7. Throughput of the SG-DMA engine suffers a diminishing penalty as the number of BDs in the ring increases. Throughput increases linearly until saturating at the AXI data bus's maximum bandwidth of 10.6 GB/s.

overall transfer time for larger transfer sizes. The median throughput results for all tested BD ring sizes and buffer sizes is shown in Fig. 7. The number of cycles has the same effect on throughput as the number of BDs, indicating buffer data can be stored discontiguously in memory without degrading throughput.

C. BD Ring Creation Performance Analysis

The BD ring creation process described earlier can be executed by either the APU or RPU, and consists largely of instantiating an SG-DMA data structure for managing BD rings and maintaining status. The SG-DMA IP block defines a suite of registers that Linux or bare metal API functions utilize for control and status operations. Both the APU and RPU access the physical memory address space of these registers directly. The C code running on the APU accomplishes the task of bypassing the virtual memory system by opening, reading and writing to the device in /dev/mem under Linux. The C code associated with the BD ring creation process is otherwise identical on both the APU and RPU.

In order to provide an apples-to-apples performance comparison, the CSM is used to measure the Δt associated with the creation of the BD ring in both cases. From the timing diagram shown in Fig. 3, the counter which is sandwiched between GPIO signals *Reset PL* and *start asserted* can be used to measure execution time between any two points encapsulated by the C code which writes these two GPIO signals. The entire sequence of steps outlined in Section III-D is the target of our timing operation. Note that the C code which writes to the buffer spaces, e.g., with gate sequences, is not included in the BD ring creation process, and instead, is handled by separate routines before this sequence is executed.

The results of the performance comparison are shown in Figs. 8 and 9. The execution time plotted on the y-axis is divided by the size of the BD ring, so the y-axis shows the average time to create a single BD in the ring. Although the APU has a 5-9x speedup over RPU in the best-case scenarios, APU suffers significantly longer stalls of up to 30 μs during execution of the user-space program when running a Linux

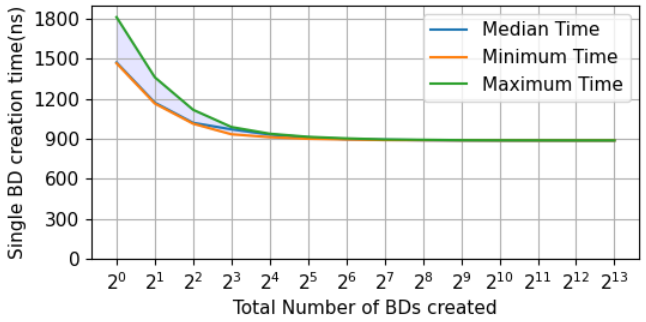


Fig. 8. Time taken for RPU to create a single BD.

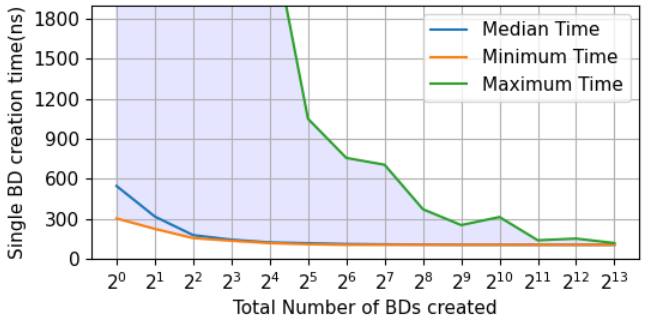


Fig. 9. Time taken for APU to create a single BD.

5.4.0 kernel created with PetaLinux 2020.2. These longer stalls in APU must be considered when attempting to meet timing requirements in a TIQC system. Fig. 10 shows the ranges of APU execution time to create a single BD and their observed probabilities of occurrence.

V. Application to Quantum Computing

The objective of this research was to develop and test SG-DMA for its ability to satisfy communication requirements for the control system of a trapped-ion quantum computer. While SG-DMA is slower than simple-mode DMA, it has advantages for higher level sequencing of data streams where segments

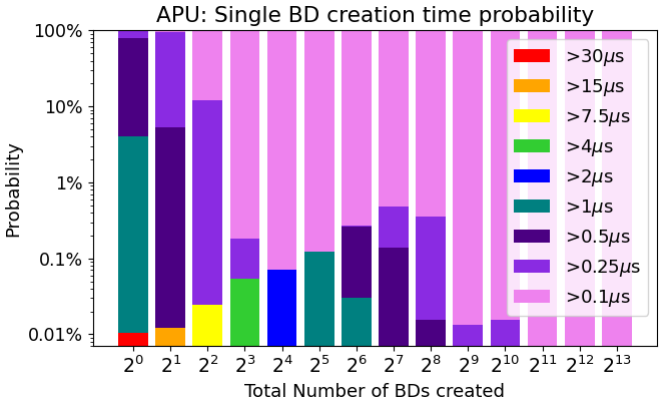


Fig. 10. Probabilities for single BD creation time after a total of 100,000 data points. The observed best-case performance is a single BD taking an average of 0.1 to 0.25 μs to be created, while the absolute worse case could take over 30 μs. BD creation time was not observed to smoothly transition between the range of all reported values.

of the underlying data are reused. This applies to both the case where the segments addressed by BDs are small but frequently reused or large and only occasionally reused. These conditions are particularly applicable to quantum computing where electronic waveforms for quantum gates or shuttling ions may require large amounts of data but are frequently repeated, with only small changes needed to recalibrate them. Recalibrating gates using SG-DMA can be acheived efficiently by updating the data in the buffer which carries through to any gates that reference that BD.

This type of scheduling offers more flexibility when large amounts of data are required, and overall cuts down on data communication requirements in the control system. In this work the dependence of throughput on BD size was measured and it was found that SG-DMA can match the performance of simple-mode DMA when buffer sizes exceed 2^{12} bytes, showing that SG-DMA is a useful way to coordinate complicated but recycled buffer sequences (e.g. gate sequences or ion shuttling waveforms). This framework is not just useful for trapped-ion quantum computing control systems, but can be applied to other quantum technologies like reconfiguring neutral atom arrays or performing composite pulse sequences in a superconducting quantum computer. Much faster gate times (down to 10's of nanoseconds) in these technologies, however, may require additional techniques to achieve real-time reconfigurability.

VI. Conclusion

This paper describes the experimental evaluation of latency and throughput of SG-DMA, implemented on a Xilinx ZCU111 RFSoC, as well as the benefits and limitations as they relate to TIQC control systems. The analysis of overall system performance considers communication between elements of a multi-processing system, including the APU, RPU, and PL, using different workloads for the SG-DMA engine. The conclusions are summarized as follows:

- The latency of the RPU to initiate SG-DMA engine operations can be characterized as having low variability and does not depend on the parameters of the transfer.
- The SG-DMA engine is most efficient when transferring large buffer regions, and has diminishing levels of variability in performance when processing large BD rings. The SG-DMA engine, though implemented in PL, does not have a fixed level of performance due to AXI bus and DDR contention.
- Repeatedly cycling through a single BD ring has the same effect on performance as increasing the number of BDs.
- The nature of SG-DMA to store both BD rings and data buffers in DDR provides high levels of flexibility at the cost of reduced performance due to the additional memory accesses required to fetch BDs.
- With respect to application of the proposed architecture to TIQC systems, the SG-DMA engine's measured throughput is sufficient to meet gate sequence transfer requirements, and is in fact much better than the $\approx$1 μs lower-bound target speed.
- The APU provides a 5-9x median execution time speedup over RPU when creating BD rings. However, APU has significantly slower worst-case performance for small numbers of BDs than RPU primarily due to Linux interrupt handling. The trade-off of high performance or low variability motivates the usage of APU and RPU for tasks that best suit their respective strengths.
- APU is better equipped for complex calculations by virtue of its L2 cache, 64-bit architecture and larger number of cores, while RPU is more appropriate for timing-critical portions of the control system such as reading/writing data to and from the PL.

The proposed multi-processor system architecture leverages parallelism across the APU, RPU and PL state machine computational units to effectively improve the performance, flexibility and scalability of TIQC control systems. Although the analysis carried out in this work focuses on the application of the RFSoC to TIQC system architectures, the results are relevant for other QC control systems and alternative RFSoCs and MPSoCs.

References

[1] S. M. Clark, D. Lobser, M. C. Revelle, C. G. Yale, D. Bossert, A. D. Burch, M. N. Chow, C. W. Hogle, M. Ivory, J. Pehr, B. Salzbrenner, D. Stick, W. Sweatt, J. M. Wilson, E. Winrow, and P. Maunz, "Engineering the quantum scientific computing open user testbed," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–32, 2021.

[2] M. A. Weber, M. F. Gely, R. K. Hanley, T. P. Harty, A. D. Leu, C. M. Löschnauer, D. P. Nadlinger, and D. M. Lucas, "Robust and fast microwave-driven quantum logic for trapped-ion qubits," *Phys. Rev. A*, vol. 110, p. L010601, Jul 2024.

[3] N. Irtija, J. Plusquellic, E. E. Tsiropoulou, J. Goldberg, D. Lobser, and D. Stick, "Design and analysis of digital communication within an soc-based control system for trapped-ion quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–24, 2023.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[5] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, "Realization of real-time fault-tolerant quantum error correction," *Phys. Rev. X*, vol. 11, p. 041058, Dec 2021.

[6] T. M. Graham, L. Phuttitarn, R. Chinnarasu, Y. Song, C. Poole, K. Jooya, J. Scott, A. Scott, P. Eichler, and M. Saffman, "Midcircuit measurements on a single-species neutral alkali atom quantum processor," *Phys. Rev. X*, vol. 13, p. 041051, Dec 2023.

[7] T. Proctor, M. Revelle, E. Nielsen, K. Rudinger, D. Lobser, P. Maunz, R. Blume-Kohout, and K. Young, "Detecting and tracking drift in quantum information processors," *Nature Communications*, vol. 11, no. 1, p. 5396, Oct 2020.

[8] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, 05 2019.

[9] J. Fjeldtvedt and M. Orlandić, "Cubedma – optimizing three-dimensional dma transfers for hyperspectral imaging applications," *Microprocessors and Microsystems*, vol. 65, pp. 23–36, 2019.

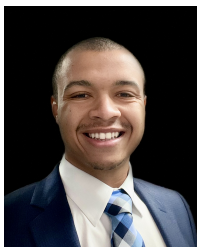
[10] G. Brilli, A. Capotondi, P. Burgio, and A. Marongiu, "Understanding and mitigating memory interference in fpga-based hesocs," in *Automation & Test in Europe Conference & Exhibition*, 2022, pp. 1335–1340.

[11] S. Alismail and D. Koch, "Efficient resource scheduling for runtime reconfigurable systems on fpgas," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, 2023, pp. 123–129.

[12] K. Manev, A. Vaishnav, and D. Koch, "Unexpected diversity: Quantitative memory analysis for Zynq UltraScale+ systems," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 179–187.

[13] M. Argyriou, "Diploma thesis: Main memory performance for realistic data access in fpga systems: An experimental study," April 2021.

[14] Y. Xu, G. Huang, J. Balewski, R. Naik, A. Morvan, B. Mitchell, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Qubic: An open-source FPGA-based control and measurement system for superconducting quantum information processors," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–11, 2021.
[15] M.-D. Zhu, L. Yan, X. Qin, W.-Z. Zhang, Y. Lin, and J. Du, "Fpga based hardware platform for trapped-ion-based multi-level quantum systems," *Chinese Physics B*, vol. 32, no. 9, p. 090702, Sep 2023.
[16] N. Fruitwala, G. Huang, Y. Xu, A. Rajagopala, A. Hashim, R. K. Naik, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Distributed architecture for fpga-based superconducting qubit control," 2024.
[17] R. Gebauer, N. Karcher, and O. Sander, "A modular rfsoc-based approach to interface superconducting quantum bits," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, 2021, pp. 1–9.
[18] X. Qin, W. Zhang, L. Wang, Y. Zhao, Y. Tong, X. Rong, and J. Du, "An FPGA-based hardware platform for the control of spin-based quantum systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1127–1139, 2019.
[19] M. Toubeix, E. Guthmuller, A. Evans, A. Faurie, and T. Meunier, "Fasquic: Flexible architecture for scalable spin qubit control," *IEEE Transactions on Quantum Engineering*, vol. 5, pp. 1–16, 2024.
[20] N. Messaoudi, C. Crocker, and M. Almendros, "A hardware-accelerated qubit control system for quantum information processing," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, 2020, pp. 1–5.
[21] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, "On the co-design of quantum software and hardware," in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, ser. NANOCOM '21. New York, NY, USA: Association for Computing Machinery, 2021.
[22] D. S. Lobser, J. W. Van Der Wall, and J. D. Goldberg, "Performant coherent control: bridging the gap between high- and low-level operations on hardware," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2022, pp. 320–330.
[23] Xilinx. (2023) Axi dma logicore ip product guide (pg021).
[24] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, "Overhead analysis of universal concatenated quantum codes," *Phys. Rev. A*, vol. 95, p. 022313, Feb 2017.
[25] A. Cross, A. Javadi-Abhari, T. Alexander, N. de Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, "OpenQASM 3: A broader and deeper quantum assembly language," *ACM Transactions on Quantum Computing*, 2021.
[26] X. Fu, L. Riesebos, M. A. Rol, J. van Straten, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, V. Newsum, K. K. L. Loh, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, "eqasm: An executable quantum instruction set architecture," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 224–237.
[27] R. Gebauer, N. Karcher, J. Hurst, M. Weber, and O. Sander, "Accelerating complex control schemes on a heterogeneous MPSoC platform for quantum computing," 2020.
[28] R. Kasprowicz, N. Karcher, D. Gusenkova, M. Spiecker, L. Grünhaupt, I. Takmakov, P. Winkel, L. Planat, N. Roch, W. Wernsdorfer *et al.*, "State preparation of a fluxonium qubit with feedback from a custom FPGA-based platform," in *AIP Conference Proceedings*, vol. 2241, no. 1. AIP Publishing LLC, 2020, p. 020015.
[29] J. Ireland, S. Protheroe, J. Williams, A. Belcher, R. Dekker, K. Schaapman, R. Iuzzolino, R. Melo, B. Valinoti, M. Bierzychudek *et al.*, "Real-time quantum-accurate voltage waveform synthesis," in *2020 Conference on Precision Electromagnetic Measurements (CPEM)*. IEEE, 2020, pp. 1–2.
[30] N. Karcher, R. Gebauer, R. Bauknecht, R. Illichmann, and O. Sander, "Versatile configuration and control framework for real-time data acquisition systems," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1899–1906, 2021.
[31] G. Kasprowicz, P. Kulik, M. Gaska, T. Przywozki, K. Pozniak, J. Jarosinski, J. W. Britton, T. Harty, C. Balance, W. Zhang, D. Nadlinger, D. Slichter, D. Allcock, S. Bourdeauducq, R. Jördens, and K. Pozniak, "ARTIQ and Sinara: Open software and hardware stacks for quantum physics," in *OSA Quantum 2.0 Conference*. Optica Publishing Group, 2020, p. QTu8B.14.
[32] B. Keitch, V. Negnevitsky, and W. Zhang, "Programmable and scalable radio-frequency pulse sequence generator for multi-qubit quantum information experiments," 2017.
[33] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, "Even more efficient quantum computations of chemistry through tensor hypercontraction," *PRX Quantum*, vol. 2, p. 030305, 2021.
[34] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–9.
[35] J. T. Merrill and K. R. Brown, *Progress in Compensating Pulse Sequences for Quantum Computation*. John Wiley & Sons, Ltd, 2014, pp. 241–294.
[36] E. Mount, C. Kabytayev, S. Crain, R. Harper, S.-Y. Baek, G. Vrijsen, S. T. Flammia, K. R. Brown, P. Maunz, and J. Kim, "Error compensation of single-qubit gates in a surface-electrode ion trap using composite pulses," *Phys. Rev. A*, vol. 92, p. 060301, 2015.
[37] Xilinx, "Zynq UltraScale+ MPSoC base targeted reference design," uG1221 (v2020.1) June 3, 2020.
[38] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.
[39] I. Pogorelov, T. Feldker, C. D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler, and T. Monz, "Compact ion-trap quantum computing demonstrator," *PRX Quantum*, vol. 2, p. 020343, 2021.
[40] V. M. Schäfer, C. J. Ballance, K. Thirumalai, L. J. Stephenson, T. G. Ballance, A. M. Steane, and D. M. Lucas, "Fast quantum logic gates with trapped-ion qubits," *Nature*, vol. 555, no. 7694, pp. 75–78, 2018.
[41] P. Schindler, D. Nigg, T. Monz, J. T. Barreiro, E. Martinez, S. X. Wang, S. Quint, M. F. Brandl, V. Nebendahl, C. F. Roos, M. Chwalla, M. Hennrich, and R. Blatt, "A quantum information processor with trapped ions," *New Journal of Physics*, vol. 15, no. 12, p. 123012, 2013.
[42] J. van Dijk, E. Kawakami, R. Schouten, M. Veldhorst, L. Vandersypen, M. Babaie, E. Charbon, and F. Sebastiano, "Impact of classical control electronics on qubit fidelity," *Phys. Rev. Applied*, vol. 12, p. 044054, 2019.
[43] L. Stefanazzi, K. Treptow, N. Wilcer, C. Stoughton, C. Bradford, S. Uemura, S. Zorzetti, S. Montella, G. Cancelo, S. Sussman, A. Houck, S. Saxena, H. Arnaldi, A. Agrawal, H. Zhang, C. Ding, and D. I. Schuster, "The QICK (Quantum Instrumentation Control Kit): Readout and control for qubits and detectors," *Review of Scientific Instruments*, vol. 93, no. 4, p. 044709, 2022.
[44] A. Stanco, F. B. L. Santagiustina, L. Calderaro, M. Avesani, T. Bertapelle, D. Dequal, G. Vallone, and P. Villoresi, "Versatile and concurrent fpga-based architecture for practical quantum communication systems," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–8, 2022.
[45] A. Bean, "Improving memory access performance for irregular algorithms in heterogeneous cpu/fpga systems," Ph.D. dissertation, Imperial College London, 2016.



Tiamike Dudley is a Ph.D. student and graduate assistant in the department of Electrical and Computer Engineering at the University of New Mexico. He received his bachelor's in Computer Engineering from the University of New Mexico. His research interests include Quantum Information Systems, Field Programmable Gate Arrays, and embedded system design.

Jim Plusquellic is a Professor in Electrical and Computer Engineering at the University of New Mexico. He received both his M.S. and Ph.D. degrees in Computer Science from the University of Pittsburgh. Professor Plusquellic received an "Outstanding Contribution Award" from IEEE Computer Society in 2012 and 2017 for co-founding and for his contributions to the Symposium on Hardware-Oriented Security and Trust (HOST).

Eirini Eleni Tsiropoulou is currently an Associate Professor at the Department of Electrical and Computer Engineering, University of New Mexico. Her main research interests lie in the area of cyber-physical social systems and wireless heterogeneous networks, with emphasis on network modeling and optimization, resource orchestration in interdependent systems, reinforcement learning, game theory, network economics, and Internet of Things. Four of her papers received the Best Paper Award at IEEE WCNC in 2012, ADHOCNETS in 2015, IEEE/IFIP WMNC 2019, and INFOCOM 2019 by the IEEE ComSoc Technical Committee on Communications Systems Integration and Modeling. She was selected by the IEEE Communication Society - N2Women - as one of the top ten Rising Stars of 2017 in the communications and networking field. She received the NSF CRII Award in 2019 and the Early Career Award by the IEEE Communications Society Internet Technical Committee in 2019.

Joshua Goldberg is a Principal Electrical Engineer in the Photonic Microsystems Technologies Department at Sandia National Laboratories, Albuquerque, NM, USA. He received a B.S. in Electrical Engineering from Texas Tech University and an M.B.A. from the University of Phoenix. Joshua Goldberg has been specializing in software development over the last 22 years. His recent efforts include modernizing control software paradigms for trapped ion quantum systems to further full stack development for quantum information platforms.

Daniel Lobser leads the control systems thrust of QSCOUT. His main research interest is the development of custom classical and quantum control hardware that employs novel paradigms for coherent operations and dynamic noise mitigation in trapped-ion qubit platforms. He received his Ph.D. in physics from the University of Colorado, Boulder in 2015, where he studied ultracold atomic gases.

Daniel Stick is a Distinguished Member of Technical Staff at Sandia National Labs. His research focuses on developing innovative technologies around atomic and quantum systems, including microfabricated surface ion traps for quantum information applications. This work includes the design and fabrication of the traps, as well as experiments with storing, transporting, and performing quantum gates on ions. Dr. Stick received his BS from Caltech and his PhD from the University of Michigan. He was the recipient of a 2012 Presidential Early Career Award for Scientists and Engineers (PECASE) for his research in trapped ion quantum computing.