HyCo: A Low-Latency Hybrid Control Plane for Optical Interconnection Networks

Felipe Göhring de Magalhães*, Mahdi Nikdast†, Fabiano Hessel‡,
Odile Liboiron-Ladouceur§ and Gabriela Nicolescu*
*Ecole Polytechnique de Montreal, QC, Canada - †Colorado State University, Fort Collins, CO, USA
‡Pontificia Universidade Catolica do Rio Grande do Sul, Brazil - §McGill University, QC, Canada
Contact: felipe.gohring-de-magalhaes@polymtl.ca

***Abstract*—Next-generation multiprocessor systems point to the integration of a large number of cores (e.g., processing and memory) where electrical networks-on-chip (eNoCs) can improve the communication performance. As the number of integrated cores increases, metallic interconnect in eNoCs becomes a bottleneck, leading to communication performance degradation and increased power consumption. Optical interconnection networks (OINs) have emerged to outperform the communication infrastructure in multiprocessor systems. Nevertheless, OINs' full capability is curbed by high latency electrical controllers required to orchestrate and (re)configure the underlying photonic components, realizing a path between sending and receiving cores. Control techniques impose a high latency to perform the network routing, limiting the full utilization of OINs. In this paper, we design a novel low-latency Hybrid Controller (HyCo) that employs acceleration techniques to reduce its execution time. HyCo is developed based on integrating centralized and distributed control techniques as well as by using pre-calculated network routes and a Bloom filter, all of which result in a considerable reduction in HyCo's latency. Simulation and prototyping results for networks up to 64×64 indicate a latency smaller than 50 ns, in the worst-case scenario.**

***Index Terms*—Optical interconnect, optical network control, scheduler, Bloom filter.**

I. Introduction

Since the introduction of chip multiprocessors (CMPs), one of the design main concerns lies in how the communication among different cores is performed. Electrical networks-on-chip (eNoCs) overcome the limitations faced by traditional bus-based interconnect. As a result, systems based on eNoCs tend to provide better communication performance [1]. However, as the number of integrated cores on a single chip continues to increase to even a larger number, metallic interconnect in eNoCs becomes a bottleneck because of the high power consumption, limited bandwidth, high latency, and poor scalability [2]. Moreover, eNoCs rely on point-to-point communication which leads to higher contention for communication among distant cores, and consequently, performance degradation by imposing higher power consumption.

To further improve the communication infrastructure in CMPs, optical interconnection networks (OINs) and 3D die stacking are considered to be the two most promising alternative paradigms [3], [4]. OINs present high bandwidth and throughput efficiency, with a better J/bit performance [5], [6]. Another advantage of employing OINs lies in their physical implementation. While communication in eNoCs is usually local (i.e., point-to-point—P2P), hence reducing their design complexity but imposing a poor scalability for shared resources, OINs offer message broadcasting, making them suitable for the new multiprocessor paradigm [7].

Despite all the potential benefits of OINs when deployed in CMPs, the performance of an OIN is constrained by its control plane [8]. When routing a message from one node to another in a CMP employing an OIN, which is a circuit-switched network, several underlying photonic components (e.g., optical switching elements) should be configured to realize an optical path between sending and receiving nodes. Such configurations are performed in an electronic controller, which, if not fast, imposes high latency in OINs, limiting the application of OINs in CMPs. Previous work demonstrated architectures with either long setup times or high design complexities, thus challenging their practical deployment [9], [10]. While some low-latency controllers have been proposed [11], [12], these solutions are still not scalable ($<$ 64 nodes) or their latency ($>$ 100 ns) has to be further improved in order to realize low-latency OINs. Moreover, most proposed controllers are designed and customized for specific networks and topologies, limiting their application to the networks for which they are designed. Although tailored solutions may result in a better controller performance, the trend for the next generation of communication systems requires independent network layers (i.e., application, control, and physical) [13], [14]. Recent work pointed to the fact that path-division (PDM) and time-division multiplexing (TDM) techniques should be employed together to fully exploit OINs [15].

The novel contribution of this paper is on developing a control solution that exploits the scalability of PDM techniques along with the global view of TDM-based centralized controllers. As a result, our proposed controller, the Hybrid Controller (HyCo), takes advantage of a fast, centralized decision algorithm combined with distributed cores for network configuration. HyCo employs a Bloom filter [26] for self-optimization, storing different configurations from which the controller learns the unavailable routes for posterior acceleration of requests. The main gain in using the distributed approach along with a centralized processing unit lies in the fact that most of the network control is performed in the central core. Therefore, the distributed units are simpler, expediting the controller execution and reducing the entire network reconfiguration time. We evaluate the chip area and

Table I
A COMPARISON AMONG DIFFERENT CONTROL SOLUTIONS PROPOSED FOR OINS.

Reference	Topology-Constrained	Algorithm	Strategy	Latency	Scalable
[16]	*HC*	*PDM*	*Distributed*	*High*	**Yes**
[17]	*C*	*PDM*	*Distributed*	*High*	**Yes**
[18]	*HC*	*WDM*	*Distributed*	*Low*	**Yes**
[12]	*HC*	*WDM*	*Centralized*	*Low*	**No**
[19]	*NC*	*PDM & WDM*	*Distributed*	*High*	**Yes**
[20]	*NC*	*PDM*	*Distributed*	*High*	**Yes**
[21]	*NC*	*PDM & WDM*	*Centralized*	*Low*	**Yes**
[22]	*C*	*WDM*	*Distributed*	*Low*	**Yes**
[23]	*HC*	*TDM*	*Centralized*	*Low*	**No**
[24]	*HC*	*WDM*	*Distributed*	*Low*	**Yes**
[25]	*NC*	*TDM*	*Centralized*	*Low*	**No**
HyCo	***NC***	***PDM & TDM***[1]	***Hybrid***[2]	***Low***	**Yes**

HC: Hard-constrained; C: Constrained; NC: Not constrained. [1]The implemented control is TDM-based, not using strict time slots. [2]The conflict and granting cores are centralized units and the configuration units are distributed.

power consumption of HyCo for ASIC 65 nm technology, showing its low overhead to be integrated in future designs. Furthermore, considering different traffic patterns and network topologies, we show that HyCo can achieve the best-case and worst-case latency of ≈1 ns and less than ≈50 ns, respectively.

The rest of the paper is organized as follows. We review some of the related work on OIN controllers in Section II. Section III presents an overview of optical switching elements and control techniques in OINs, followed by Section IV, where we introduce HyCo. Next, Section V presents simulation and synthesis results for HyCo, comparing it to the state-of-the-art controllers. Last, Section VI concludes this paper.

II. RELATED WORK

The controller of an OIN should ideally work as a standalone unit, thus not being dependent on the architecture (topology) of the network nor using the characteristics of the devices in the network (e.g., switches). Nevertheless, most of the state-of-the-art controllers are topology-dependent (i.e., architecture specific), which are limited to a single network and optimizing at most the performance of the specific network for which they are designed. In [16], a control unit based on the path-division algorithm was presented. A dimension-order routing algorithm is applied to the electrical layer to configure the optical path in the optical network. A similar technique was used in [17] in which the control scheme is also based on PDM. The latency was calculated to be around 3.5 ns in an 8×8 mesh network when peripherals run at 5 GHz. An electro-optic hybrid approach was presented by [20] where optical switching elements are in charge of transmitting data using PDM, while electrical switching elements are in charge of closing the path using packet-switching techniques. Moreover, when the optical-path shows high contention, the electrical path is used to transmit a message.

The control unit in [18] was based on wavelength-division multiplexing (WDM). Each input/output (I/O) is assigned to a specific wavelength to communicate without arbitration. In [19], a routing technique was introduced which is based on wavelength selection integrated with spatial routing (i.e., PDM is used with WDM). An asynchronous and variable-length packet switching technique was presented in [22]. Every intellectual property (IP) is attributed with one exclusive label which corresponds to each output fiber. When a message comes across a new network node, the message gets delayed while its label is computed by the electrical node. Furthermore, a multi-cast scheduling control solution was proposed in [23] focusing on input-queued switching elements based on the weight-based arbiter (WBA) and TDM.

In [25], a technique based on pre-calculated routes stored in fast-access memories (i.e., look-up table (LUT)) was introduced. All communication combinations are evaluated during the design time and the routes are calculated using the shortest-path-first (SPF) algorithm [27]. The main advantage of this technique is the latency reduction due to the use of fast-access memories, and hence no computation is required when establishing an optical path. Nevertheless, this approach lacks scalability and its application is limited to low-radix networks.

The design of a parallel scheduler for Clos-network switching elements was presented in [12]. The work breaks the Clos scheduler circuit into three pipeline stages for accelerating request computation. The three stages are output-port allocation, routing, and configuration update. Although presenting low latency to compute requests, the solution is limited to the Clos network. Also, the scalability of the design is compromised because of the introduced pipeline stages. Finally, the work in [24] presented a control solution for contention handling based on optical-buffering through introducing a three-stage buffering method. This method uses electronically controlled wavelength-routing switching elements in combination with optical-delay lines to temporarily store data [28]–[30]. However, optical buffering is still immature, and consequently this solution is cost inefficient and impractical.

Table I compares different control schemes discussed above with HyCo, highlighting five points: Topology-Constrained, Algorithm, Strategy, Latency, and Scalability. According to the table, most of prior efforts present a network controller that is customized for a specific topology or type of architecture, aiming to optimize at most the performance of one specific architecture. Consequently, the topology-constrained controllers

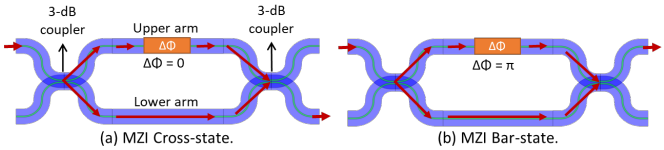

Figure 1. A 2×2 MZI-based switching element in Cross and Bar states.

have a limited contribution as their employment is hard-limited by the very one topology for which they are built. Also, several proposed controllers consider complex PDM techniques, thus adding extra control latency that can completely shatter the benefits of using OINs in CMPs. Compared to the aforementioned work, HyCo stands out by using a hybrid approach, indicating the lowest latency while maintaining scalability. HyCo is also applicable to a large variety of network topologies, including those based on tunable switching elements and the ones that only need arbitration (i.e., using passive components). Moreover, as HyCo is not topology-dependent, designers can easily modify it (i.e., replicating blocks for each frequency used), so that it can comport WDM routing, further expanding HyCo's capabilities.

III. Basic Concepts and Background

OINs are well known for their capacity to transfer data at high transmission rates. Different from their electrical counterparts, OINs use optical carriers over optical vias (e.g., optical waveguides) to transmit data. In general, an optical interconnect consists of different basic components, including waveguides, modulators, laser sources (off-chip or on-chip), switching elements, optical filters, and photodetectors [31]. The waveguides are the equivalent of wires in electrical interconnect, and are used for routing and propagating optical signals throughout the network. Modulators are considered to modulate the electronic (digital) data onto optical signals. The optical switching elements are employed to route the transmitted signals over the optical network, while optical filters can select a specific wavelength (depending on their design parameters) among other wavelengths used in the network. Finally, photodetectors convert optical signals from the network to electronic data. In the context of this work, the concepts of lasers, modulators, and photodetectors are not relevant as the focus is on the network routing and the network-access control. Readers can refer to [31] to learn more about optical interconnect in CMPs.

A. Optical Switching

In OINs, two devices are often used to build switching elements, microresonators (MRs) and Mach–Zehnder Interferometers (MZIs) [32]. For example, a passive MR is a wavelength-selective optical filter applied to select a desired wavelength from a WDM input [33]. Nevertheless, while MRs are compact with low power consumption, MZIs are more robust and less prone to process variations [34], thus are used as a validation technology in this work.

Fig. 1 illustrates a conventional 2×2 MZI-based integrated switching element and its operating states: Cross and Bar.

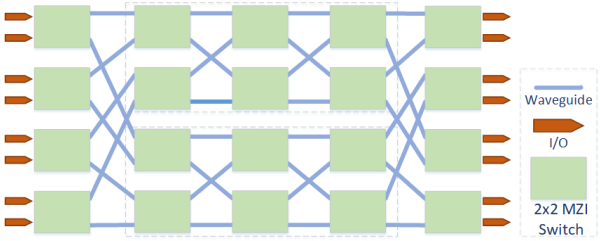

Figure 2. An example of an 8×8 Beneš switch network.

It is composed of two input and two output ports and two interferometric arms of equal length—with an integrated phase shifter on on arm—connected between two 3-dB couplers. The light is split into the two arms (50:50) at the input of the MZI and using a 3-dB coupler. The switching between the ports can be achieved by introducing a phase shift on one arm, and hence constructive or destructive interference at the output coupler. This can be done using electro-optic (e.g., by applying a reverse bias voltage to a P-N junction integrated on MZI arms), or using thermo-optic tuning (e.g., microheaters) within the MZI structure. As a result, the effective refractive index of one of the arms will change, hence the phase difference between the optical signals in two arms of the MZI. Fig. 2 shows an example of a switch network built based on MZIs. In the figure, each square represents a 2×2 MZI, and 20 MZIs are cascaded to form a topology known as an 8×8 Beneš architecture [35]. Note that this is shown as an example only, and HyCO can be applied to any OIN, as we will discuss later.

B. Control Techniques in OINs

The controller in an OIN arbitrates the network access as well as defines routes. Therefore, the controller has to consider busy or broken communication lines as well as contention. The controller should configure each network switching element (based on MZIs in this work) to enable the optical signals to travel in a desired direction. Different techniques can be employed for the network access and routing, including time-division, path-division, and wavelength-division multiplexing.

In the **time-division multiplexing (TDM)** technique, the time is arranged in several recurrent windows (slots) and then a time window is attributed for each input IP [36]. The time slots might have the same duration depending on the priority attributed to the IPs. Moreover, the amount of data transmitted over the media through each slot may either be the same for all the IPs or vary according to their priorities.

Another possibility when controlling OINs concerns a path availability approach, implemented as a circuit-switched (CS) technique. This is possible using a **path-division multiplexing (PDM)** technique where each transmission uses a unique (in time and/or space) path in the optical network. Each network node is configured accordingly to perform the optical transmission. It is a suitable technique used for dynamic network configurations, where an electrical layer has access to all the network nodes and configures those needed for each communication. The main appeal of circuit switching is its

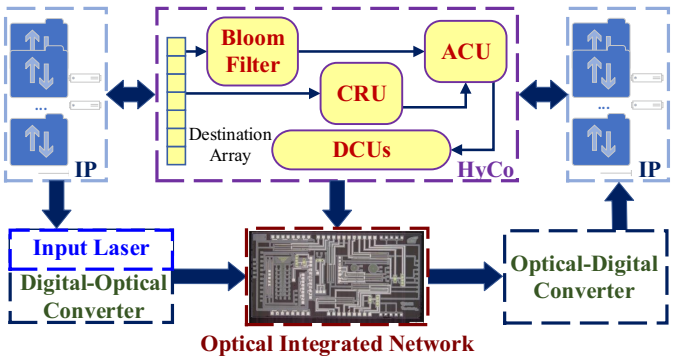

Figure 3. An abstract overview of our proposed controller, HyCo. CRU is conflict-resolution unit, ACU is access-control unit, and DCU is distributed-configuration unit.

application in 3D stacked on-chip systems in which each layer of the chip holds one parcel of the entire architecture [37].

Another technique applied to control units for OINs is **wavelength-division multiplexing (WDM)** [38]. This technique is similar to TDM. However, instead of using different time slots, WDM employs different optical wavelengths without any interference. When using WDM, the available bandwidth of the channel is divided into sub-channels for each of which a given wavelength is attributed [19]. This technique allows wavelengths overlapping to be used for two different scenarios: (i) different IPs requesting access simultaneously, and (ii) for the cases when only one IP is requesting access and all wavelengths are free. In both cases, the controller might attribute more than one wavelength to the same IP, so its transmission might occur in a parallel stream.

In this work, we use both TDM and PDM techniques in HyCo. While the network arbitration is performed using a TDM-based approach, the network routing uses a PDM-based approach. As a result, HyCo is able to further explore the OIN's communication capabilities, as discussed next.

IV. HyCo: A Low-Latency Controller for OINs

Leveraging the scalability of PDM and the full control of centralized cores, HyCo employs both approaches to fully exploit OIN capabilities. Moreover, HyCo takes advantage of pre-calculated routes to expedite the routing decision, and hence further reducing the latency. In particular, HyCo uses a Bloom filter [26] which is a data structure used to accelerate the processing time when checking data sets. It enables self-optimization in HyCo through learning and storing network critical information (e.g., impossible network routes) during runtime. An overview of the different building blocks and organization of HyCo is shown in Fig. 3, and its execution flow is presented in Algorithm 1. We discuss the main building blocks of HyCo and their functions in the rest of this section.

A. Conflict-Resolution Unit

The conflict-resolution unit (CRU) is responsible for detecting conflicts in targeted IPs. A conflict is defined as any situation in which two or more source IPs are targeting the same destination IP at the same time. First, the CRU analyzes

Algorithm 1: HyCo execution algorithm.

```
foreach input i do
    switch State do
        case Idle do
            if requestReceived(i) then
                State ← Request Received;
        case Request Received do
            if targetIsAvailable(getTarget(i)) then
                State ← Test Target;
        case Test Target do
            if targetConflict(getTarget(i) then
                if nextOnRoundRobin(i) then
                    State ← Verify Route;
            else
                State ← Verify Route;
            end
        case Verify Route do
            if checkBloomFilterForRoute(getTarget(i)) then
                State ← Configure Network;
        case Configure Network do
            triggerNetworkConfiguration(getTarget(i));
            State ← Communication;
        case Communication do
            while communicating(i) do
                State ← Communication;
            end
            State ← Idle;
    end
end
```

all the requests, looking for a conflict. If a conflict is found, a Round-Robin (RR) algorithm is applied to determine which IP should have its access granted. For detecting a conflict, a matrix method is used, where for every new request round, all the source-destination pairs are mapped to a matrix $\mathcal{R}$ of requests, and then each column j in $\mathcal{R}$ is checked for any possible conflicts. The matrices are created based on the I/O's port IDs. For example, for a 3×3 optical network, $\mathcal{R}(i,j) = 1$ denotes the input port I_i requests to access the output port O_j in the network, such that:

$$\forall_{ij},\ I_i \ \text{ request } \ O_j \leftrightarrow \mathcal{R}(i,j) = 1. \tag{1}$$

As the matrix can be accessed directly (i.e., the hardware implementation is a register), no extra processing is needed, thus accelerating the conflict detection. Once the request matrix is generated, all the columns of the matrix (each column is associated with an output port) are assessed to find any possible conflicts. Conflict detection can be defined as:

$$\forall ii'j, i \neq i', \neg XOR[R(i,j)] \wedge OR[R(i',j)] \implies \text{conflict}(O_j) = 1. \tag{2}$$

If a conflict is detected, a TDM-based first-in-first-out (FIFO) queue is implemented based on the RR algorithm to decide which port should be granted access. This is to guarantee a minimum access time for each input port, following a TDM approach. Since the RR algorithm uses a FIFO for each input, each request is treated individually.

B. Bloom Filter

A Bloom filter [26] is employed in HyCo to avoid unnecessary path searching, expediting the network routing. When

routing all requesting inputs, the Bloom filter stores the information of paths that have been tried and were unsuccessful, avoiding re-assessing the availability of such unsuccessful paths in the future requests. A Bloom filter essentially consists of a bit vector of length m, which is chosen based on the number of processed entries. To insert a new entry in the Bloom filter, different hash functions are used. A hash function $h(x)$ is a mathematical expression that maps its input x on a given output, based on an implemented equation. In the context of HyCo, the Bloom filter works by receiving a destination array, applying hash functions, and then testing the filter on the resulting positions. If all tested positions are set to 1, it means that the desired routes have been already tried and were unavailable because of either a conflict or contention. In this case, the RR algorithm is applied and the selected requesting port is stalled. Then, the new destination set is tested again. If any of the tested positions is set to 0, the controller tries to find routes for all the requesting inputs. If any input is not satisfied within a specific time, which can be configured by the user, the input set is fed to the Bloom Filter.

The main gain when using a Bloom filter is the fact that as the system runs, previously received requests can be treated faster as the information regarding whether a route is possible or not is already stored in the filter. This enables HyCo to self-optimize, as its knowledge of the controlled network will improve over time, thus lowering its control latency. Analyzing possible network congestion points during design time would fail to achieve the same results. As the network size grows, the number of network routes increases as well, leading to a larger amount of data to be processed. Consequently, two main problems may rise: unbearable processing time to find all congested routes [39], and the storage needed to save the pre-analyzed routes would be prohibitive as well. This way, the Bloom filter solution employed in HyCo avoids both problems.

C. Access-Control Unit

The access-control unit (ACU) is responsible for controlling the access to the network. It holds the states of the connected IPs (i.e., available or busy). When a request arrives, the ACU evaluates any possible conflicts through checking the status of both the destination IP and the Bloom filter. If no impediment is found, the ACU triggers the network switching-element configuration. By the time it receives the confirmation pointing that the network is configured, an acknowledge signal is sent to the requesting IP. Furthermore, request computation runs in parallel, and each possible request port is considered as one running process. As a result, it is possible for HyCo to receive requests, solve conflicts, and grant access to the network within very few clock cycles.

D. Distributed-Configuration Unit

The optical path setup in an OIN is critical as different aspects are involved in this configuration, such as network contention and proper mapping. Different approaches are used to address this issue, such as PDM techniques [17]. The distributed-configuration unit (DCU) implements a technique similar to PDM. To improve the controller response time, it uses pre-calculated routes stored in fast-access memories, such as lookup tables (LUTs) [25]. Although using LUTs can reduce the network configuration latency, it demands a high memory footprint to store all the routes. This is because each pair of input-output paths leads to a different network configuration. As a result, the LUT memory consumption grows as the number of network ports increases.

Because a practical implementation relying solely on LUTs is hardly achievable for large-scale networks, each DCU block holds a reduced version of a LUT, called LITE-LUT. It only stores a small portion of the entire network information, such as the most requested paths or the paths for a portion of the network. The DCUs, storing the LITE-LUTs, are then distributed. For example, assume a hypothetical network where a read-only memory is connected with four nodes. In this case, the paths these nodes take to read from the memory can be stored in a LITE-LUT, as they would happen more frequently. Any other communication combinations, such as to different IPs connected to the same network, would have to be routed during the system execution. When a route is not stored in the LITE-LUT, regular XY algorithm is employed in the DCU.

V. Simulation and Prototyping Results

Different network topologies, including butterfly, fat-tree, ring, star, and mesh, are used to assess HyCo's efficiency. Also, several traffic patterns, such as complement and all-to-all, are considered to assess HyCo's performance under various request conditions. The complement traffic pattern is used to verify the largest paths in the network. Largest paths refer to the paths in which the transmitted message passes through the largest number of network hops. All-to-all traffic pattern comprises all possible communication combinations in the network, as the IPs present in the system request access to all the nodes, one by one. For such validations, the traffic load is not taken into account as aspects involved in conflict resolution and network configuration are not directly affected by the traffic load. The focus is on determining the response time of HyCo for different request configurations and patterns.

A. HyCo Execution Results

HyCo is powered by a Bloom filter which is used to dynamically reduce the controller latency as it executes. The real impact of the filter on HyCo's latency when used in different networks and using different traffic patterns is shown in Fig. 4. In the figure, $N \times M$ representation is used, with N and M referring to, respectively, the number of I/O nodes. As it can be seen, as HyCO executes (see the x-axis), its latency reduces for different network typologies and scales thanks to the Bloom filter. After HyCo runs for a given time, which changes depending on the topology, the filter reaches its threshold value and the latency becomes steady. On average, the Bloom filter helps reduce the latency by more than 30% in most cases. For the topologies that are strictly non-blocking (e.g., PILOSS and 32×32 strictly non-blocking (SNB) topologies), the Bloom filter has no impact. This is

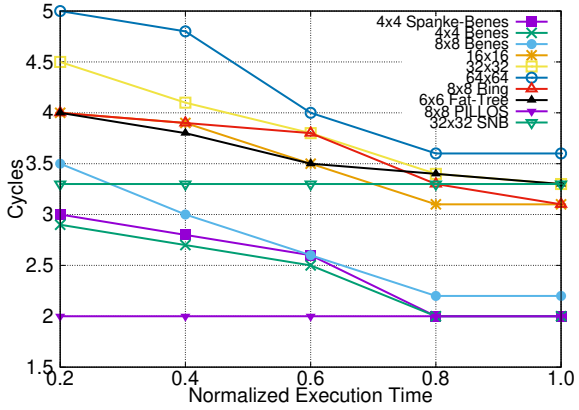

Figure 4. HyCO's latency for different network topologies and scales.

due the fact that there are no impossible routes, therefore the Bloom filter is never employed in the control process. Note that Fig. 4 presents HyCo's latency as a factor of execution time. The latency illustrates the number of clock cycles needed by HyCo to compute each request. The x-axis is presented as a normalized execution time.

B. HyCo Synthesis Results

HyCo is synthesized for the same topologies compared in Fig. 4. The selected networks with various radix, switching element counts, and topological properties are considered to verify the latency of HyCo under different scenarios. Note that HyCo is not limited to the network topologies considered as an example in this section. The only requirement for using HyCO in other topologies is the generation of the LITE-LUT. Both ASIC and FPGA technologies are targeted.

Table II presents the results obtained after the synthesis for the Virtex V 330T FPGA from Xilinx and those for the Stratix IV FPGA from Altera. We give the number of used programmable FPGA blocks (PFB) for each block in HyCo, the minimum block propagation delay (i.e., clock period), and the number of DCUs. For the 64×64 Butterfly network, which includes almost 400 network nodes, HyCo is able to operate with a period of 8.57 ns on the Virtex V FPGA. As presented in Section V-A, it takes five clock cycles in the worst-case execution scenario. In the best-case scenario, HyCo is able to process requests in only two clock cycles. The average latency of HyCo is 3.5 cycles. More specifically, in the worst-case scenario, HyCo can receive, process, configure the network, and grant requests in less than 50 ns.

Finally, synthesis targeting ASIC technology is performed using the proposed flow for the 65 nm STMicro technology. Table III lists the obtained results in four columns. *Area* indicates the occupied chip area for the synthesized block in millimeter square (mm^2). Also, *Delay* is the block signal propagation delay, which shows the minimum clock period for the block. Power consumption is depicted in the *Power* column. Moreover, *#DCUs* shows the number of distributed-configuration units for each topology. Obtained results indicate that, in the worst-case scenario and for the largest network (i.e., 64×64 Butterfly), the maximum delay is 3.3 ns, and the chip area, including the wiring, does not exceed 1.15 mm^2.

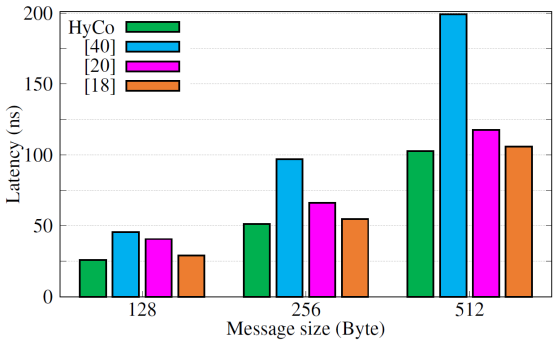

Figure 5. Latency comparison between HyCo and state-of-the-art controllers.

Table II
SYNTHESIS RESULTS FOR XILINX AND ALTERA FPGAS

	Xilinx	Altera	Xilinx	Altera	
Network Topology	**#PFBs**		**Block Delay (ns)**		**#DCUs**
4x4 Spanke-Beneš	*1443*	*2325*	*3.57*	*3.91*	*5*
4x4 Beneš	*1615*	*2573*	*3.39*	*3.7*	*6*
6x6 Fat-Tree	*3037*	*3945*	*3.38*	*4.01*	*54*
8x8 Ring	*2370*	*2931*	*4.15*	*4.76*	*8*
8x8 Beneš	*2556*	*3556*	*4.12*	*4.75*	*20*
8x8 PILOSS	*36225*	*3981*	*4.29*	*5.01*	*64*
16x16 Beneš	*5792*	*6587*	*6.25*	*6.75*	*48*
32x32 Star	*18199*	*19273*	*7.57*	*7.97*	*120*
32x32 SNB	*32948*	*41958*	*7.54*	*8.63*	*1024*
64x64 Butterfly	*67918*	*85837*	*8.57*	*9.77*	*384*

Table III
SYNTHESIS RESULTS FOR THE 65 NM STMICRO LIBRARY

Topology	**Area (mm^2)**	**Delay (ns)**	**Power (W)**	**#DCUs**
4x4 Spanke-Beneš	*6.7E-3*	*1.05*	*2.7E-3*	*5*
4x4 Beneš	*7.5E-3*	*1.04*	*2.8E-3*	*6*
6x6 Fat-Tree	*0.05*	*1.7*	*0.015*	*54*
8x8 Ring	*22.7E-3*	*1.5*	*7.92E-3*	*8*
8x8 Beneš	*27.4E-3*	*1.1*	*9.09E-3*	*20*
8x8 PILOSS	*0.04*	*1.2*	*0.01*	*64*
16x16 Beneš	*0.08*	*1.5*	*0.02*	*48*
32x32 Star	*0.35*	*2*	*0.05*	*120*
64x64 Butterfly	*1.15*	*3.3*	*0.23*	*388*

Based on the results for synthesis processes and the latency results presented in Section V-A, it is possible to verify the impact of HyCo in the system execution. Taking as an example the result for the 4×4 Spanke-Beneš, one can estimate HyCo's latency: it takes two clock cycles for HyCo to perform the access control and network configuration. As the presented synthesis values for the ASIC technology states a delay of ≈1.05 ns, it can be concluded that HyCo has the latency of ≈2.10 ns to operate with the 4×4 Spanke-Beneš switching network. For the 64×64 topology, HyCo takes 3.5 clock cycles to perform the access control and network configuration. The FPGA synthesis results point to a delay of ≈9.77 ns (see Table II), which leads to a HyCo latency of ≈35 ns for the 64×64 Butterfly network.

C. Comparison with Other Controllers

For a fair comparison of HyCo with state-of-the-art controllers, the well-known 8×8 Beneš [35] network is used. Fig. 5 indicates that HyCo has the lowest latency under different message sizes. Considering a pre-defined optical switching element latency, the latency for each optical bit to pass through the network is rounded to be 200 ps [40]. The comparison is performed by analyzing the total time it

takes for a message to be arbitrated and traverses the network: $TotalTime = CL + N_{ob} \times TD$. Here, CL is the control latency, N_{ob} is the number of transmitted bits, and TD is the transmission delay. Three different message sizes (128 B, 256 B, and 512 B) are considered. Considering Fig. 5, the solution in [41] utilizes uniquely a centralized control core, which compromises its scalability. Moreover, the approach proposed in [19] relies on PDM techniques, which can lead to high latency as the network scales. Finally, the solution presented in [17] is based on an operation frequency of 5 GHz, and hence the validations are under simulations only.

VI. Conclusion

We present a novel hybrid control solution for optical interconnects. HyCo incorporates the benefits of both centralized and distributed cores to control the network. The main gain in using the distribution approach allied to a centralized core is that most of the processing is performed on the central core, and thus simplifying the distributed units. This helps expedite the controller execution time, hence reducing the time required to (re)configure the network. Indeed, HyCo's distributed-configuration unit (DCU) latency is significantly small (<50 ns). Scalability issues of previous low-latency controllers are addressed using the introduced DCUs. The matrix method accelerates the conflict detection and the Bloom filter enables self-optimizing in HyCo. To the best of our knowledge, HyCo presents the best solution to cope with future optical interconnect needs, presenting low latency while maintaining the flexibility for employment in different network topologies.

Acknowledgements

This research was supported by the National Science Foundation (NSF) under grant CNS-2046226 and by the Fonds de recherche du Québec (FQRNT) under grant 247997.

References

[1] S. Tota *et al,*. A Multiprocessor Based Packet-switch: Performance Analysis of The Communication Infrastructure. In *IEEE Workshop on Signal Processing Systems Design and Implementation*, 2005.

[2] I. O'Connor and F. Gaffiot. *On-Chip Optical Interconnect for Low-Power*. Springer US, 2004.

[3] W. C. Lo *et al,*. TSV and 3D Wafer Bonding Technologies for Advanced Stacking System and Application at ITRI. In *Symposium on VLSI Technology*, 2009.

[4] W. J. Dally. Future Directions for On-Chip Interconnection Networks. *http://www.ece.ucdavis.edu/ ocin06/talks/dally.pdf*.

[5] Book. *Interconnects and Data System Throughput*. 2017.

[6] C. A. Thraskias *et al,*. Survey of Photonic and Plasmonic Interconnect Technologies for Intra-Datacenter and High-Performance Computing Communications. *IEEE Communications Surveys Tutorials*, 2018.

[7] A.R. Mickelson. Silicon Photonics For On-chip Interconnections. In *Custom Integrated Circuits Conference*, 2011.

[8] A. Shacham *et al,*. Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors. *IEEE Transactions on Computers*, 2008.

[9] A. Biberman *et al,*. Broadband Operation of Nanophotonic Router for Silicon Photonic Networks-on-Chip. *IEEE Photonics Technology Letters*, 2010.

[10] Z. Li, M. Mohamed *et al,*. Iris: A Hybrid Nanophotonic Network Design for High-performance and Low-power On-chip Communication. *Journal of Emerging Technologies in Computing Systems*, 2011.

[11] F. Lou *et al,*. Towards a Centralized Controller For Silicon Photonic MZI-based Interconnects. In *IEEE Optical Interconnects Conference*, 2015.

[12] P. Andreades *et al,*. Low Latency Parallel Schedulers for Photonic Integrated Optical Switch Architectures in Data Centre Networks. In *2017 European Conference on Optical Communication (ECOC)*, 2017.

[13] Huawei Technologies. White Paper - Huawei Observation to NFV. Technical Report 399662, 2014.

[14] Ericsson AB. Network Functions Virtualization and Software Management. Technical Report Uen 284 23-3248, 2014.

[15] H. Gu *et al,*. Time-Division-Multiplexing–Wavelength-Division-Multiplexing-Based Architecture for ONoC. *J. Opt. Commun. Netw.*, 2017.

[16] H. Jia *et al,*. Five-Port Optical Router Based on Microring Switches for Photonic Networks-on-Chip. *IEEE Photonics Technology Letters*, 2013.

[17] Z. Li *et al,*. ESPN: A Case For Energy-star Photonic-on-Chip Network. In *IEEE International Symposium on Low Power Electronics and Design*, 2013.

[18] H.A. Khouzani *et al,*. Fully Contention-free Optical NoC Based On Wavelenght Routing. In *CSI International Symposium on Computer Architecture and Digital Systems*, 2012.

[19] J. Chan *et al,*. Photonic Interconnection Network Architectures Using Wavelength-selective Spatial Routing For Chip-scale Communications. *IEEE/OSA Journal of Optical Communications and Networking*, 2012.

[20] J. Wang *et al,*. A Highly Scalable Butterfly-Based Photonic Network-on-Chip. In *IEEE International Conference on Computer and Information Technology*, 2012.

[21] F. Yan *et al,*. Hifost: a scalable and low-latency hybrid data center network architecture based on flow-controlled fast optical switches. *IEEE/OSA Journal of Optical Communications and Networking*, 2018.

[22] H. Yang *et al,*. Design of Novel Optical Router Controller and Arbiter Capable of Asynchronous, Variable length Packet Switching. In *International Conference on Photonics in Switching*, 2006.

[23] M. Shoaib. Selectively Weighted Multicast Scheduling Designs For Input-Queued Switches. In *IEEE International Symposium on Signal Processing and Information Technology*, 2007.

[24] Y. Liu *et al,*. All-Optical Buffering Using Laser Neural Networks. *IEEE Photonics Technology Letters*, 2003.

[25] F. G. de Magalhães *et al,*. Design and Modelling of a Low-Latency Centralized Controller for Optical Integrated Networks. *IEEE Communications Letters*, 2016.

[26] B.H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications ACM*, 1970.

[27] B.V. Cherkassky, A. V. Goldberg and T. Radzik. *Shortest paths algorithms: theory and experimental evaluation.* Elsevier, 1996.

[28] D.K. Hunter *et al,*. Buffering In Optical Packet Switches. *Journal of Lightwave Technology* , 1998.

[29] M. Renaud *et al,*. Transparent Optical Packet Switching: The European ACTS KEOPS Project Approach. In *IEEE Lasers and Electro-Optics Society*, 1999.

[30] T. Sakamoto *et al,*. Variable Optical Delay Circuit Using Wavelength Converters. *Electronics Letters*, 2001.

[31] S. Pasricha and M. Nikdast. A survey of silicon photonics for energy-efficient manycore computing. *IEEE Design & Test*, 37(4):60–81, 2020.

[32] Z. Guo *et al,*. 16 × 16 Silicon Optical Switch Based on Dual-Ring-Assisted Mach–Zehnder Interferometers. *Journal of Lightwave Technology*, 2018.

[33] M.K.Chin *et al,*. Design And Modeling of Waveguide-coupled Single-mode Microring Resonators. *Journal of Lightwave Technology*, 1998.

[34] M. Nikdast et al. Chip-scale silicon photonic interconnects: A formal study on fabrication non-uniformity. *Journal of Lightwave Technology (JLT)*, 34(16):3682–3695, 2016.

[35] V.E. Beneš. On Rearrangeable Threestage Connecting Networks. Bell Syst. Tech. J., 1962.

[36] W.P. Boothroyd *et al,*. A Time Division Multiplexing System. *Transactions of the American Institute of Electrical Engineers*, 1949.

[37] J. Carson. The Emergence of Stacked 3D Silicon And Its Impact On Microelectronics Systems Integration. In *IEEE International Conference on Innovative Systems in Silicon*, 1996.

[38] C. White. *Data Communications And Computer Networks: A Business User's Approach*. Thomson Course Technology, 2007.

[39] J. Leeuwen. *Handbook of Theoretical Computer Science*.

[40] Y. Xiong *et al,*. Towards a Fast Centralized Controller for Integrated Silicon Photonic Multistage MZI-based Switches. Optical Society of America, 2016.

[41] J. Jian *et al,*. A Fast Hierarchical Arbitration in Optical Network-on-Chip Based on Multi-Level Priority QoS. *IEICE Transactions*, 2016.