An Analysis of Hypermesh NoCs in FPGAs

M. Binesh Marvasti, IEEE Student Member, and T.H Szymanski, IEEE Member

Abstract—Accurate analytic models for the area, delay and power of the Hypermesh NoC topology, realized with the Altera family of FPGAs, are presented. Hypermeshes are based on the concept of hypergraphs, which consist of a set of nodes and a set of hyperedges, where the hyperedges represent low-latency switches which interconnect multiple nodes with deterministic latencies. Three different switch designs for the hyperedges are proposed and evaluated. Two parallel algorithms are considered; (a) the Bitonic sorting algorithm, and (b) the FFT parallel algorithm. The analytic models are shown to be very accurate, typically within 6%. The 2D Hypermesh is compared to the 2D layouts of the Binary Hypercubes (BHC) and Generalized Hypercubes (GHC) in terms of area, energy per algorithm, and the Energy-Area product. The Energy-Area product is proposed as an useful design metric to evaluate NoCs, which combines both the cost and the performance metrics of an NoC into one. Our analysis indicates that the 2D Hypermeshes generally have considerably lower area, energy, and Energy-Area product compared to the 2D layouts of the Hypercubes.

Index Terms—Network-on-Chip, Hypermesh, analytic model, graph model, power, area, delay, energy, parallel algorithm, FPGA

1 INTRODUCTION

The Network-on-Chip (NoC) is a new design paradigm for Systems-on-Chip (SoC) that uses an interconnection network for communication instead of buses. NoCs offer scalable communications which solve the scalability problems of buses [1][2].

The design of NoCs is a trade off between multiple competing constraints such as minimizing power, minimizing area and minimizing packet delay [3]. Early design-space exploration tools are critical to achieve near-optimal designs, due to the complex trade-offs between area, latency, performance, and power consumption. Early design-space exploration tools can use analytic or simulation-based methodologies. Analytic models have a theoretical basis, they do not require any detailed logic design in VHDL or Verilog, and they are reasonably accurate. Simulation tools such as Altera’s PowerPlay tool or the ORION CAD tool can be highly accurate, but they are complex to develop, and they lack a theoretical basis. Furthermore, simulation based tools require a detailed hardware design to be developed in VHDL or Verilog, before the simulation can begin.

It has been estimated that the majority of the system costs are fixed early in the design process, and hence early design-space exploration tools using analytic models allow designers to focus on feasible designs early, before detailed hardware design and test is undertaken.

Recently, Altera announced a new family of FPGAs based on the Intel 14 nm Tri-Gate CMOS process, providing designers with up to a 10X increase in programmable ASIC gates [11]. Xilinx announced a new 3D FPGA (the Virtex-7 HT) with 6.8 billion transistors, providing designers with ∼ 20 million programmable ASIC gates [12]. These capacities will increase in the future, and promote the development of NoCs designs for parallel processing in FPGAs.

Conventional graph-based networks G(V, E) consist of a set of N vertices V and a set of edges E. (The terms ‘vertex’ and ‘node’ are used interchangeably.) Each vertex contains a router with multiple edges, where each edge e = (u, v) ∈ E joins exactly two vertices (u, v ∈ V). Graph-based networks include the Mesh, Torus, and Hypercubes. In many parallel scientific algorithms, N = r^n processing nodes are logically arranged in an n-dimensional radix-r grid, where n = log_r N. In the 2D Mesh, each node has a router with direct connections to its 4 nearest neighbors in the grid. The diameter of a graph is defined the largest number of edge traversals between any pair of nodes. The diameters of the 2D Mesh and Torus are 2(√N−1) and 2(√N−1)/2 hops respectively. In the Binary Hypercube (BHC), each node has a router with direct connections to log_2 N nodes, and diameter of the BHC network is log_2 N. In the radix-r Generalized Hypercube (GHC), each node has a router with direct connections to (r−1)log_r N nodes, and diameter of the GHC network is log_r N. Fig. 1 shows 2D layouts of the BHC and radix-4 GHC. (The graphs are defined in Appendix A.) The Hypercube topologies have many advantages over meshes such as low diameter, large bandwidth, high fault tolerance, and the ability to realize the types of permutations which frequently occur in parallel scientific algorithms. However, the degree of the routers in the hypercube networks increases quickly, resulting in higher area and power consumption than necessary.

The Hypermesh is based on the concept of hypergraphs, which are generalizations of the conventional graph. A hypergraph G(V, HE) consists of a set of vertices V and a set of hyperedges HE, where each hyperedge he ∈ HE represents a logical relationship amongst several vertices V. An n-dimensional radix-r Hypermesh is a hypergraph with N = r^n vertices arranged in an n-dimensional radix-r grid, where the r vertices that are aligned along each dimension.
belong to a hyperedge. Each vertex has a router of degree \( \log_r N \), and each hyperedge represents a logical switch of degree \( r \) which can interconnect \( r \) vertices with a deterministic latency [19]. A hyperedge with \( r \) vertices can represent a centralized or distributed switch with \( r \) parallel channels, which can perform \( r \) parallel data transfers simultaneously.

Hypermeshes can be constructed using electrical or optical technologies. An electrical Hypermesh can use Space Division Multiplexing (SDM) to achieve the parallel channels, while an optical Hypermesh can use Wavelength Division Multiplexing (WDM) to achieve the parallel channels [19]. We note that the distributed switches used in the hyperedges are much simpler than the routers, since the routers have much more functionality; The routers typically implement Virtual Channel Flow Control; they use Time Division Multiplexing (TDM) to support multiple virtual channels over the same physical link, they have arbiters to resolve contention between multiple TDM-based virtual channels, and they have queues for buffering multiple TDM-based virtual-channels for arbitrary amounts of time [7]. In contrast, the switches in the hyperedges do not have queues, they do not support TDM, and they do not exhibit variable queueing delays (but the switches in the hyperedges may have pipeline latches). The Hypermesh topologies have the same advantages of the Hypercubes topologies, and they achieve lower diameters with lower degree routers. The lower diameter results in lower area and lower power consumption compared to the hypercube networks.

The Cray Titan supercomputer built by the US Dept. of Energy uses a multi-channel 3D Torus interconnection network, which shares the multi-channel property of the 3D Hypermesh. The K supercomputer manufactured by Fujitsu Corp. uses a multi-channel 6D Torus networks, which also shares the multi-channel property of the 3D Hypermesh (see section 2). However, the performance of these graph-based networks can be improved by transitioning to full Hypermesh networks, especially with the emerging Silicon Photonics technology which supports WDM optical transmissions.

To date, there are no analytic models for the area and power of electrical Hypermesh NoCs realized in ASIC or FPGA technologies. We propose analytic models which are very accurate and general, and enable the early design-space exploration of different NoC topologies in FPGAs, before VHDL or Verilog coding and detailed design are undertaken.

Our contributions include the following: (1) We present analytic models for the area, delay and power for three switch designs realized in FPGA technology, which can be used for the hyperedges and the routers. (2) We develop accurate analytic models of the area, delay and power of 2D Hypermesh NoCs in FPGA technologies. (3) The Hypermesh NoCs are compared to the Binary Hypercube (BHC) and Generalized Hypercube (GHC) NoCs in terms of area, energy per algorithm, and energy-area product under two parallel algorithms, (i) the Bitonic sorting algorithm, and (ii) the parallel Cooley-Tukey FFT algorithm. (4) Finally, the energy-area product is proposed to evaluate different topologies, which reflects both the cost and performance metrics of NoCs. This paper is the first to: (i) present analytic models for the area, delay and power of Hypermesh NoCs realized in FPGA technology, and (ii) compare hypercube and Hypermesh NoCs when realized in an FPGA technology.

The remainder of this paper is organized as follows. Section 2 surveys prior works. Section 3 develops analytic models for basic components of a router and hyperedge. Section 4 develops analytic models for routers. Section 5 explains Hypermesh topology. Sections 6 and 7 present analytic models for Hypercubes and Hypermesh NoCs. Section 8 and 9 present the results and conclude the work.

## 2 Previous Works

In [5], an analytic methodology was proposed for the area and power of a IQ-switch using a Broadcast-and-Select architecture, using 180 nm CMOS VLSI. Ref. [8] presents a general performance model for latency and throughput of different NoCs using queuing theory. An analytic model for energy consumption of different NoCs realized in ASICs is proposed in [13]. A low-latency packet-switched NoC router for FPGA technology was proposed in [14]. In [15] a general NoC router for FPGAs was proposed. In [6], analytic energy and area models of NoC topologies for tiled chip multiprocessors using predicted 65nm VLSI technology were presented. Ref. [10] proposed analytic power and area models for four crossbar switch designs realized in Altera FPGAs, and an area and power analysis of 2D Torus and GHC NOCs using the different crossbar switches.

Ref. [18] proposed the spanning-bus hypercube network, where all the nodes aligned along a dimension are interconnected with a single bus. The concept of a hypergraph-based Hypermeshes was presented in [17] [19]. In a Hypermesh, all the nodes aligned along a dimension are interconnected with hyperedge, which represents a multi-channel switch with very low latency. Low latency multi-channel distributed switches are easily implemented using optical technologies which support WDM. The implementation of 2D Hypermeshes using centralized electronic crossbar switches was also presented in [19]. It has been
shown that the Hypermesh topology provides better performance relative to conventional graph-based networks, such as the Mesh, Torus and Hypercubes under different traffic patterns [19] [24]. Hypermeshes can also use distributed electronic crossbar switches with bit pipelining to reduce the latency in the hyperedges [19] [20] [21]. Ref [23] explored the impact of communication locality in meshes and hypemeshes. Ref. [22] proposed a performance model for the mean average latency in wormhole-switched Hypermeshes in the presence of hot-spot traffic.

The above papers either have proposed cost and performance models for graph-based networks such as the Mesh, Torus, or Hypercube, or have been addressed performance models of the Hypermesh. None of the prior papers presents accurate analytic models for the area, delay, and power of the Hypermesh NoC realized in FPGA technologies, or evaluated different switch designs for the hyperedges to improve the performance of the Hypermeshes. Preliminary experimental results of area and delay of the Hypermesh NoCs realized in FPGAs was presented in [16].

2.1 The Cray-Titan 3D Multi-channel Torus

The Gemini network used in the Cray Titan supercomputer is a 3D multi-channel Torus, where each node represents 2 Opteron processors, and each node has 10 Torus-links [26]. The Gemini Torus supports SDM with multiple ‘Torus-links’ in each dimension to overcome the bandwidth limitations. There are 4 Torus-links in the X and Y dimensions, and 2 Torus-links in the Z dimension. Each Torus-link is 12 lanes wide. Hence, the Gemini Torus network supports multiple SDM channels in each dimension of a grid, as does the Hypermesh network. However, the Gemini network is a conventional graph-based network, where every edge connects exactly two nodes. A traffic flow must traverse many routers due to the large diameter of the Torus network, and will incur variable queuing delays (due to Virtual Channel Flow Control) at every router traversed. The Hypermesh networks explored in this paper add low-latency switches with deterministic latencies to the dimensions, which significantly decreases the diameter of the network, and allows a traffic flow to traverse only one router and one low-latency switch per dimension. The performance of the Gemini Torus network could be improved, by adding such switches to the dimensions, and these switches can be modelled as hypergraph hyperedges. There is negligible cost to add low-latency switches to the dimensions when WDM optical technologies are used, since the hyperedge switches are ‘virtual’ and can be implemented using tunable laser diodes, where multiple WDM channels can share the same fiber [19].

3 ANALYTIC MODELS OF BASIC COMPONENTS

Routers using Input-Queueing (IQ) are commonly used in NoCs. Fig. 2 illustrates an IQ router which consists of Input buffers, a crossbar switch, a routing table, and an arbiter. Arriving data is stored at the input buffers, which are divided into Virtual Channels (VC) to prevent deadlock and increase throughput. The arbiter schedules the transmission of data. The crossbar switches are typically constructed with multiplexers, demultiplexers, broadcast buses, and memories. In this section, we propose accurate analytic models for the basic components of an NoC router, and we evaluate the best router designs.

In an FPGA, logic functions are configured through a series of programmable resources. These programmable resources are typically called Logic Elements (LEs), Adaptive Logic Modules (ALMs). Array of LEs or ALMs are placed in Logic Array Blocks (LABs). Each of these programmable resources typically includes one or more Look-Up Tables (LUTs) and DFFs. In this paper, we will use the Altera Cyclone IV family of FPGAs, where the basic programmable resource is one LE. All of our analytic models will use the LE as the basic unit of programmable resource, and will express the area of complex designs in terms of the number of LEs used.

3.1 Power Models

In ASICs, the power of typical standard cell is given by (1), where \( f_{clk} \) is the clock rate, \( f_{duty} \) is the probability of a signal transition at the output port, \( K_{int} \) is the intrinsic capacitance of the cell switched during an output transition, and \( K_{load} \) is the capacitive load being driven (including the wire and input capacitances).

\[
P(\text{stdcell}) = \frac{1}{2} f_{clk} f_{duty} (K_{int} + K_{load})(V_{dd})^2 \tag{1}\]

All the power models developed in this paper use (1), where the parameters are adjusted to reflect the FPGA technology.

3.1.1 Mux and Demux

A large N-to-1 multiplexer tree can be constructed with \((N-1)/(m-1)\) smaller m-to-1 multiplexer cells arranged in a tree topology [5]. In an FPGA, the area of an N-to-1 Mux tree with a datapath width of W bits is given by (2.1). The value of m is determined by the FPGA synthesizer (HDL compiler)
and the FPGA family. A mux-tree in which all inputs switch at the duty cycle uses the maximum power given in (2.3). A mux-tree in which only one input switch uses the minimum power given in (2.4).

\[
A_{Mux}(N, 1, W) = \left\lfloor \frac{(N-1)}{m-1} S_{Mux}(m) \right\rfloor W \quad (2.1)
\]

\[
D_{Mux}(N, 1, W) = \log_2 N D_{Mux}(m) \quad (2.2)
\]

\[
P_{Mux}(N, 1, W) = \left\lfloor \frac{(N-1)}{m-1} S_{Mux}(m) \right\rfloor K_{Mux(m)} f_{clk} f_{duty} W \quad (2.3)
\]

\[
P_{Mux}(N, 1, W) = \log_2 N K_{Mux(m)} f_{clk} f_{duty} W \quad (2.4)
\]

The parameter \( S_{Mux}(m) \) is defined as the *Synthesis Efficiency* of a basic m-to-1 multiplexer cell, i.e., it equals the average number of LEs used for the synthesis of a basic m-to-1 logic cell, given the selected FPGA family [9]. The parameter \( K_{Mux(m)} \) is defined as the internal capacitance of a basic m-to-1 multiplexer cell. (Our experiments indicate that the capacitance of the wires between logic cells in a mux-tree is negligible.)

The delay of a large multiplexer tree \( Mux(N, 1, W) \) is given by (2.2), where \( D_{Mux}(m) \) is defined as the delay of a smaller m-to-1 Mux cell. The power consumption of a large multiplexer tree, with a datapath width of \( W \) bits, is given by (2.3) or (2.4).

Using the same methodology, it is possible to construct a large 1-to-N Demux tree using a binary tree of smaller 1-to-m Demux cells. Hence, the area, delay, and power of a large 1-to-N Demux tree are given by (3.1), (3.2), and (3.3) respectively. \( S_{Demux(m)} \) is defined as the Synthesis Efficiency of an 1-to-m Demux cell [9], and \( K_{Demux(m)} \) is defined as the internal capacitance of a basic 1-to-m demultiplexer cell.)

### 3.1.2 Broadcast buses and links

The programmable interconnections in FPGAs use wire segments with different lengths. Long distance wires are constructed by aggregating multiple short wire segments [27], and they use repeaters to increase the signal strength and lower delays. The delay of long wire in an FPGA is linearly related to the length of wire.

**Definition:** In this paper, all wire lengths are expressed in multiples of the ‘LE-length’ (unless otherwise noted), where one ‘LE-Length’ is defined as the square-root of the area of the basic programmable resource, the ‘Logic Element’ (or LE).

According to our experiments, the delay of a short wire between logic gates within one logic block is typically negligible, while the delay of a short bus between distinct logic blocks is \( O(\log L) \), and the delay of a long bus between distinct logic blocks is \( O(L) \).

We use an accurate 2 piece-wise linear approximation for the bus delay given by (4.1) and (4.2) (in nanoseconds).

The power dissipated in a bus with \( W \) parallel wires is given by (4.2) where \( L \) is the length of bus. In this model, the \((1/2)V_d^2\) term has been absorbed into the constant \( K_{wire} \). \( K_{wire} \) represents the capacitance switched by the wire for one LE-length (i.e., for \( L = 1 \)) during a signal transition, which is determined experimentally.

In a bus composed of \( W \) parallel wires which drives \( Z \) other gates (i.e., ‘standard-loads’), the power is given (4.3), where \( K_{std} \) represents the mean input capacitance switched per signal transition for a logic gate (i.e., the ‘standard-load’), and where \( L \) is the average length of each wire.

\[
D_{Bus}(L) = 0.85 + 0.0198L, \quad 4 \leq L \leq 20 \quad (4.1)
\]

\[
D_{Bus}(L) = 1.01 + 0.012L, \quad 20 \leq L \quad (4.2)
\]

\[
P_{Bus}(L, W, Z) = L K_{wire} f_{clk} f_{duty} W + Z K_{std} f_{clk} f_{duty} W \quad (4.3)
\]

#### 3.1.3 Input Buffer

Input buffers can be implemented with 2 methods, (i) using Embedded Memory (EM) Blocks, or (ii) using DFF registers. In the Altera Cyclone IV FPGAs, hundreds of EM blocks named M9K are available, each with 9 Kbits of SRAM. Each EM can be programmed as a W-bit wide memory, for \( 1 \leq W \leq 32 \) bits.

When a FIFO queue is realized using an EM block, two pointers are required (for reading and writing), and the length of these pointers will depend upon the capacity of the input buffer (given by \( D_t \)). The number of the LEs used for these pointers is given by (5.1).

In FPGAs, the power used by a FIFO queue realized using an EM block is relatively independent of the FIFO capacity or the data path width [9], when FIFO queue fits completely within one EM block. The power consumed in an input buffer (i.e., a FIFO queue) residing in one EM block, for \( W \leq 32 \) bits, is expressed by (5.2). Referring to (5.2), \( K_{IBEM} \) represents the internal capacitance switched by each bit of the EM block during a signal transition, and \( f_{duty} \) is the duty cycle.

In the second method, the input buffers are constructed using the DFFs available in the LEs of the FPGA. The number of LEs used will depend on the buffer depth and width. An output buffer is typically used for latching the result of a read operation.

The area and power consumption of the input buffers constructed using DFFs are given by (5.3) and (5.4) respectively, where \( K_{IBDFF} \) is the internal capacitance switched by each bit of a single DFF during a signal transition.

\[
A_{IBEM}(D_t) = 2 \log_2 D_t \quad (5.1)
\]

\[
P_{IBEM} = K_{IBEM} f_{clk} f_{duty} \quad (5.2)
\]

\[
A_{IBDFF}(D_t, W) = (D_t + 1)W \quad (5.3)
\]

\[
P_{IBDFF}(D_t, W) = (D_t + 1)K_{IBDFF} f_{clk} f_{duty} W \quad (5.4)
\]

### 4 Crossbar Switches

In this section, analytic models for three crossbar switch designs as shown in Fig. 3 are presented. These crossbar switches are used in the routers and hyperedges.
the pipeline latches do not incur any additional LE cost. The total area of NxN PBS switch is equal to the area of N N-to-1 Muxes, and is given by (7.1). The total power consumed by the PBS switch is given by (7.2), where the power consumed by the pipeline latches is explicitly shown. The delay of PBS switch of size NxN is is given by (7.3).

4.3 S3: The Pipelined Demux-Mux Switch (PDM)

Fig. 3c illustrates a pipelined Demux-Mux (PDM) switch. The PDM crossbar design consists of Demuxes that connect to the input ports, and Muxes that connect to the output ports. In this switch, NW pipeline DFFs are inserted at the input and output ports of the switch, requiring 2NW DFFs. These pipeline latches are shown by black squares in Fig. 3c, and they utilize the DFFs already available in the LEs used in the mux or demux trees, which were previous unutilized. Hence, the pipeline latches do not incur any additional LE cost. The total resource usage of a PDM switch (W bits wide) uses the area models in (2.1) and (3.1), and is given by (8.1).

The total power consumed by the PBS switch uses the power models in (2.4) and (3.3) and is given by (8.2), where L is the average wire length in a bus between a Demux and Mux tree \( \approx \sqrt{A_{S1}(N,W)} \) and \( Z = 1 \). There are 2NW pipeline DFFs in the PDM design. The power of a DFF \( P_{DFF} \) is given by \( P_{DFF} = K_{DFF}(\bar{f}_{clk})f_{duty} \), where \( K_{DFF} \) represents the internal capacitance switched by one DFF per transition. (The factor \( 1/2 \)) has been absorbed into \( K_{DFF} \).

The delay of the PDM switch of size NxN is equal to the delay of a large Demux tree plus the delay of a large Mux tree and is given by (8.3).

\[
A_{S1}(N,W) = NA_{Mux}(N,1,W) \quad \text{ (6.1)}
\]

\[
P_{S1}(N,W) = NP_{Bus}(L,W,N) + NP_{Mux}(N,1,W) \quad \text{ (6.2)}
\]

\[
D_{S1}(N,W) = D_{Mux}(N,1,W) + D_{Bus}(\sqrt{A_{S1}(N,1,W)}) \quad \text{ (6.3)}
\]

\[
A_{S2}(N,W) = NA_{Mux}(N,1,W) \quad \text{ (7.1)}
\]

\[
P_{S2}(N,W) = NP_{Mux}(N,1,W) + 2NW P_{DFF} + NP_{Bus}(L,W,N) \quad \text{ (7.2)}
\]

\[
D_{S2}(N,W) = D_{Mux}(N,1,W) + D_{Bus}(\sqrt{A_{S2}(N,W)}) \quad \text{ (7.3)}
\]

5 HYPERMESH TOPOLOGY

An n-dimension radix-k Hypermesh can be defined as a regular hypergraph which has \( N = k^n \) nodes arranged in a n-dimensional grid. Each node is identified with an \( n \) digit vector in radix \( k \) arithmetic. All the nodes aligned along any of the \( n \) dimensions are interconnected with a low-latency multi-channel switch, which is modelled as a hypergraph hyperedge. Each node is connected with a hyperedge to all the other nodes whose positions differ from its own in exactly one dimension. Each node contains a small router of degree \( D = n + 1 \) to allow packets to traverse the \( n \) dimensions, and to interconnect the router to the processor. In a concentrated Hypermesh, let each node contain \( M \) processors which share \( Y \) ports on the router as shown in Fig. 2, so the router degree \( D = n + Y \). Fig. 4a shows the logical diagram of a 2D Hypermesh, where each bold vertical or horizontal line denotes a hyperedge. Each hyperedge

Fig. 3. 3 Switch Designs, (a) BS, (b) PBS, and (c) PDM (Black squares are pipeline DFFs)

4.1 S1: The Broadcast-and-Select Switch (BS)

Fig. 3a illustrates an NxN Broadcast-and-Select (BS) crossbar switch. Each input buffer broadcasts over a dedicated broadcast bus to all output ports. Each output port has one large (N-to-1) Mux-tree, which selects the broadcast from the appropriate input buffer. The total area of this switch is given by (6.1). The total power consumed in this switch uses the models (2.3) and (4.3) and is given by (6.2). (Low-power Mux trees can be used, but additional logic is required to disable the switching of unwanted input signals.) The length of a broadcast bus \( L \) is modelled as \( N \sqrt{A_{Mux}(N,1,W)} \), where \( A_{Mux}(N,1,W) \) is the area of a large Mux tree and can be found by (2.1). (This length is computed from a simple Steiner tree connecting an input port to all Mux trees.) The delay of the BS switch is equal to the delay of a large Mux tree \( (D_{Mux}(N,1,W)) \) plus the delay of the broadcast bus of length \( L \), and is given by (6.3).

4.2 S2: The Pipelined Broadcast-and-Select Switch (PBS)

Fig. 3b illustrates a PBS switch design. In this switch, NW pipeline DFFs are inserted at the input and output ports of the switch requiring 2NW pipeline DFFs, given a width of W bits. These pipeline DFFs are shown by black squares and utilize the DFFs already available in the LEs used in the mux or demux trees, which were previously un-used. Hence,
represents a low-latency multi-channel switch, which can be implemented in an FPGA with a centralized or a distributed electronic crossbar switch.

Fig. 4b shows a 4x4 2D Hypermesh implemented with centralized electrical crossbar switches in each dimension. Fig. 4c shows the Hypermesh using distributed crossbar switches in each dimension. In the distributed implementation a distributed broadcast-and-select switch can be used, where multiple broadcast channels span each dimension and multiplexers are used at each node to extract the data from the correct broadcast channel. (A controller must control the multiplexers [19,20,21]).

The 2D Hypermesh with \( N = r^n \) nodes has \( n = 2 \) dimensions and radix \( r = \sqrt{N} \). For concentration \( M=Y=1 \), it uses small routers of degree 3. In contrast, the 2D Mesh or Torus uses larger routers of degree 5. In a 2D Mesh a packet must traverse up to \( \sqrt{N} - 1 \) routers and edges to traverse a row or column. In a 2D Hypermesh, a transmission along a row or a column only traverses one router and one hyperedge. The traversal of a hyperedge entails traversing a crossbar switch with a low deterministic latency, but the hyperedge switch is much less complex than a router. In a 2D Hypermesh, the graph-theoretic diameter grows much more slowly than the 2D Mesh or 2D Torus as the network size scales up, since fewer router traversals (with their complex routing functions and associated stochastic queueing delays) are required.

### 6 Area and Delay Models for Hypermesh NoCs

In this section, area and delay models of a 2D Hypermesh are presented.

#### 6.1 Area:

Assume each processor core uses \( C \) LEs of the FPGA. In a Concentrated network, let each node have \( M \) processors with a total area of \( MC \) LEs.

Assume 2D Hypermesh has \( N \) nodes, with one router per node, and a concentration of \( M \) processors per node. All of the \( M \) processors in a node share \( Y \) high-speed IO ports on the local router as shown in Fig. 2, where \( Y \leq M \). Therefore, each node has \( Y \) Mux trees of size \( M \)-to-1 to connect the \( M \) processors to the \( Y \) input ports on the router, and it has \( Y \) Demux trees of size 1-to-\( M \) to connect the \( Y \) output ports of the router to the \( M \) processors. In a 2D Hypermesh with radix \( r = \sqrt{N} \) and concentration \( M=Y=1 \), the router degree is fixed at \( D = 3 \).

When a router has a concentration of \( M > 1 \) processors interfacing to \( Y \) router ports, the area of the Mux and Demux trees is given by:

\[
A_{\text{Mux}}(Y, M, W) = Y A_{\text{Mux}}(1, M, W) + Y A_{\text{Mux}}(M, 1, W)
\]

(9)

There are 2 design options to multiplex multiple Virtual Channels (VCs) onto each directed edge, Space-Division-Multiplexing (SDM) or Time-Division-Multiplexing (TDM). The TDM design option is less costly and is assumed here [7]. The use of TDM does not increase the bisection bandwidth of the network, and it does not increase the number of edges (and wires) to be supported. (In a Hypermesh, SDM is used to add multiple channels to each dimension, so the bisection bandwidth is increased relative to the Mesh or Torus.) When \( K \) VCs are time-multiplexed onto each edge, each router input port also has a 1to-\( K \) Demux, a K-to-1 Mux, and \( K \) VC latches (each \( W \) bits wide). Each router output port has also one latch (\( W \) bits wide), to select and hold the flit being propagated [7].

Hereafter, assume the concentration \( M=Y=1 \), i.e., each node has one processor. The area used by each router is denoted by \( A_{RTR}(D,W,K) \), where \( D \) is the router degree, \( W \) is the number of bits in each directed edge, and \( K \) is the number of VCs which can be time multiplexed onto an edge. Therefore, the total area of the router is given by:

\[
A_{RTR}(D,W,K) = A_{SW}(D,W) + D \left( K A_{VC} + W \right) + D A_{Dmux}(1,K,W) + D A_{Mux}(K,1,W)
\]

(10)

where \( A_{SW}(D,W) \) is the area of the crossbar switch as described in the section 4. The area for the buffer memory used by a VC \( (A_{VC}) \) is given by (5.1) and (5.3).

In a 2D Hypermesh with \( N \) nodes as shown in Fig. 4b and 4c, there are \( \sqrt{N} \) hyperedges in each dimension. Each hyperedge represents a \( \sqrt{N} \times \sqrt{N} \) crossbar switch, and the 2D Hypermesh has a total of \( 2\sqrt{N} \times \sqrt{N} \) crossbar switches.

The total area of a 2D radix-\( \sqrt{N} \) Hypermesh with \( N \) nodes (with a concentration of \( M \) processors per node sharing \( Y \) router ports, with router degree \( D = 2 + Y \), with datapath width \( W \), with \( K \) VCs time-multiplexed onto an edge, where area of the processor core = \( C \) LEs) can be expressed as:

\[
A_{HM}(K, M, Y, W) = NMC + N A_{RTR}(2+Y,W,K) + N A_{Conc}(Y,M,W) + 2\sqrt{N} A_{SW}(\sqrt{N},W)
\]

(11)

where the first term is the area of the processors given a concentration of \( M \) processors per node, the second term is the area of the routers with degree \( D = 2 + Y \), the third term is the area needed for the Mux and Demux trees to interface the \( M \) processors to the \( Y \) router ports, and the last term is the area of the crossbar switches used in the hyperedges and is found by (6.1), (7.1), and (8.1). (Note that every hyperedge can use slightly smaller switches with \( \sqrt{N} \) ports rather than \( \sqrt{N} \) ports, since a packet arriving on a given port will never depart on the same port. The last term in the above equation can be modified to show this design improvement.)
6.2 Critical Path Delay

The maximum working frequency of each topology is mathematically modelled as the inverse of the critical path delay in the topology. The critical path delay is defined as the maximum delay of combinational logic between a source and destination register, including the setup and hold times for the registers. In each NoC topology the critical path depends on the router delay, the type of input buffers (EM or DFFs), and the wire delay between routers. The delay models for wires and routers were presented earlier, in sections 3 and 4.

Definition: Define the 'Node-Distance' (ND) of an NoC topology with N nodes which is laid out in 2 dimensions in a regular manner, as the shortest distance between 2 adjacent nodes in the X or Y dimensions.

The ND is equal to the length of the shortest distance between 2 adjacent nodes, expressed in terms of the LE-length as defined earlier. The ND of an NoC topology is denoted by \( \Gamma_{\text{Top}} = \sqrt{A_{\text{Top}}}/N \), where \( A_{\text{Top}} \) is the area of the NoC topology. The ND represents the distance between nearest neighbours, assuming a 2D layout with a unity aspect ratio.

In FPGAs, the position of EM blocks are fixed on the die floorplan during the fabrication process. When EM blocks are selected as input buffers, the distance between 2 EM blocks in one row may affect the delay of topology. However, our experiments indicate that this distance results in negligible wire delays.

In 2D Hypermesh, when EM blocks are used for the input buffers, the critical path is defined as the delay of (i) an EM block and (ii) the crossbar switch and (iii) the hyperedge. Assume that the 2D Hypermesh has a 2D layout with a unity aspect ratio, it uses centralized crossbar switches, and the concentration M=Y=1. Therefore, the delay of a 2D Hypermesh is given by:

\[
D_{\text{HM}}(\text{EM})(D, W) = D_{\text{SW}}(3, W) + D_{\text{EM}} + D_{\text{HE}}(\sqrt{N}, W) + 2D_{\text{Bus}}((\sqrt{A_{\text{HM}}} - \sqrt{A_{\text{HE}}})/2)
\]  

(12)

where \( D_{\text{EM}} \) is delay of accessing an EM block, \( D_{\text{SW}}(3, W) \) is the delay of an 3x3 switch used in the

![Image](image_url)

Fig. 4. A 2D Hypemesh with 16 nodes: (a) Logical diagram [19], (b) Hypermesh using centralized crossbar switches in each dimension (black boxes represent centralized switches). (c) Hypermesh using distributed crossbar switches in each dimension (triangles represent routers, small squares represent PEs)

![Image](image_url)

Fig. 5. The length of the worst-case transmission through a hyperedge = \( L_{\text{WC}} = (1/2)(\sqrt{A_{\text{HM}}} - \sqrt{A_{\text{HE}}}) \) router, and \( D_{\text{HE}}(\sqrt{N}, W) \) is the maximum delay of the crossbar switch used in the hyperedges. The last term represents the worst-case delay of a bus for a transmission from a source node to the centralized switch, and back to a destination node. Referring to Fig. 5, the worst-case transmission traverses a distance of \( \sqrt{A_{\text{HM}}} / 2 \) to reach the centralized switch.

With the same assumptions, the delay of a 2D Hypermesh using DFF registers for the router input buffers is given by:

\[
D_{\text{HM}}(\text{DFF})(D, W) = D_{\text{SW}}(3, W) + D_{\text{HE}}(\sqrt{N}, W) + 2D_{\text{Bus}}((\sqrt{A_{\text{HM}}} - \sqrt{A_{\text{HE}}})/2)
\]

(13)

which is the same as (14), except that the delay of accessing the EM block is removed and replaced by the DFF access time, which is negligible. (The delay of the longer busses of length \( (1/2)(\sqrt{A_{\text{HM}}} - \sqrt{A_{\text{HE}}}) \) as shown in Fig. 5 can be reduced by using the pipelined switch designs (PDM and PBS) in the hyperedges.)

7 Analytic Power Models

The power of an Hypermesh NoC operating in steady-state is modelled by 3 components, the power consumed in the routers, the hyperedges and the buffers, as shown in (14).

\[
E[P_{\text{HM}}] = E[P_{\text{routers}}] + E[P_{\text{HEs}}] + E[P_{\text{IBs}}]
\]

(14)

where \( P_{\text{routers}} \) denotes the average power consumed in the routers over a time interval \( T \), where \( P_{\text{HEs}} \) denotes the average power consumed in the the switches and busses in the hyperedges over the time interval \( T \),
and where \( P_{IB} \) denotes the average power consumed in the input buffers over the time interval \( T \). When the NoC operates under a steady-state condition, these powers can be experimentally measured.

Assuming all the components (routers, input buffers and hyperedges) are 100% loaded, the maximum power used by a 2D Hypermesh NoC with \( N \) nodes and with concentration \( M=Y=1 \) is given by:

\[
\text{Max}(P_{HM}(N,W)) = N P_{\text{Sw}} + N D P_{IB} + 2\sqrt{N} P_{HE}(\sqrt{N}, W)
\]  

(15)

where \( D = 3 \) denotes the router degree, and where \( P_{HE}(\sqrt{N}, W) \) is the average power consumed in the switch and busses in a hyperedge.

For all power analyses in this section, the following assumptions are made:

1) The NoC uses a deterministic ordered-dimension XY routing algorithm, and packets are delivered along minimum hops paths.

2) The loads are evenly distributed over the nodes.

Given a traffic pattern in which the packet destinations are distributed across the network, some links will remain un-utilized and the analytic power model for any NoC network can be approximated by modifying the previous equation, to reflect the reduced load of each component.

In a 2D wormhole-switched Hypermesh NoC with \( N \) nodes, the analytic power model can be approximated by modifying the previous equation (15) as follows:

\[
P_{HM}(N,W) = \lambda_{\text{Sw}} N P_{\text{Sw}} + \lambda_{IB} N D P_{IB} + \lambda_{HE} 2\sqrt{N} P_{HE}(\sqrt{N}, W)
\]  

(16)

where \( \lambda_{\text{Sw}}, \lambda_{IB}, \) and \( \lambda_{HE} \) are the effective arrival rate of flits to (i) a port of a router switch, (ii) an input buffer, and (iii) a hyperedge respectively. \( D = 3 \) is the degree of 3x3 switch used in a router, for concentration \( M=Y=1 \). Referring to (16), the power used by the router switches, input buffers, and hyperedges are reduced relative to the maximum possible power consumption, by a term equaling the effective arrival rate to each component.

**Hypermeshes with Distributed Switches:** We now adapt (15) and (16) for the power of a 2D Hypermesh using distributed switches as shown in Fig. 4c). The average power consumed by a hyperedge using a distributed switch which is 100% loaded is given by:

\[
P_{HE}(\sqrt{N}, W) = \sqrt{N} P_{Mux}(\sqrt{N} - 1, 1, W) + \sqrt{N} P_{Bus}(\sqrt{A_{HM}}, W, \sqrt{N} - 1)
\]  

(17)

The first term is the power used in the multiplexers in each dimension, as shown in Fig. 4c. The second term is the power consumed in the \( \sqrt{N} \) broadcast buses each of length \( \sqrt{A_{HM}} \), which is found using (4.3). Note that in a 2D Hypermesh with \( N \) nodes, there are \( \sqrt{N} \) muxes of size \((\sqrt{N}-1)\)-to-1. The results of (17) is used in (16) to compute the power consumption of a 2D Hypermesh using distributed crossbar switches.
Hop is defined as the transmission of a packet from a source router input buffer, through the router, through the edge or hyperedge, and to the input buffer of the destination router. The expected number of hops $H$ traversed by a packet in a Butterfly permutation indicates how many routers and edges or hyperedges are traversed. Each packet in a permutation traverses $H$ hops on average, and let there be $N_{link}$ links in the network. Therefore, the effective arrival rate of flits to a link per packet is equal to $H/N_{link}$, weighted by the effective arrival rate of flits to the network ($\lambda$).

The traffic patterns used in the Cooley-Tukey FFT Algorithm are shown in Fig. 8. The FFT data-flow graph consists of $n$ Butterfly permutations (denoted $E_j$ for $0 \leq j < n$) followed by a Bit-Reversal (BR) permutation. The expected number of hops traversed by a packet per permutation in the FFT graph is determined next.

In 2D Hypermesh, each packet transmission in a butterfly permutation $E_j$ for $0 \leq j \leq n-1$ requires one hop. The expected number of hops $H$ traversed by a packet per permutation in the FFT algorithm is

$$H = 1$$  \hspace{1cm} (24)

The result of (24) is used to compute the effective arrival rate of flits to each component in (16).

### Table 1: List of Symbols Used Throughout This Paper

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Network Size</td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td>$\log_2 N$</td>
<td></td>
</tr>
<tr>
<td>$K$</td>
<td>$\sqrt{N}$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma_p$</td>
<td>Node-Distance (ND) of Topology $T$</td>
<td>$\sqrt{TEs}$</td>
</tr>
<tr>
<td>$L$</td>
<td>The expected distance traversed by one packet</td>
<td>ND</td>
</tr>
<tr>
<td>$H$</td>
<td>The expected number of hops traversed by one packet</td>
<td>Integer</td>
</tr>
<tr>
<td>$E_j$</td>
<td>Butterfly Permutation, $j = 0, ..., \log_2(N) - 1$</td>
<td></td>
</tr>
</tbody>
</table>

FIG. 7. Traffic patterns in the Bitonic sorting algorithm with 16 nodes

Cooley-Tukey FFT Algorithm: The traffic patterns used in the Cooley-Tukey FFT Algorithm are shown in Fig. 8. The FFT data-flow graph consists of $n$ Butterfly permutations (denoted $E_j$ for $0 \leq j < n$) followed by a Bit-Reversal (BR) permutation. The expected number of hops traversed by a packet per permutation in the FFT graph is determined next.

7.2 Hop Count

Two parallel algorithms are explored, the (1) Bitonic sorting algorithm, and (2) Cooley-Tukey FFT Algorithm, and both use Butterfly permutations. Table 1 lists the symbols used throughout this paper.

Bitonic Sorting Algorithm: The traffic patterns used in the Bitonic sorting algorithm are shown in Fig. 7. The algorithm uses $n(n+1)/2$ conflict-free butterfly permutations $E_j$, $j = 0, ..., n - 1$.

In a 2D Hypermesh, each packet transmission in any butterfly permutation ($E_j$, $j = 0, ..., n - 1$) traverses one hop. Therefore, the expected number of hops traversed by a packet per Butterfly permutation in the Bitonic sorting algorithm is given by:

$$H = 1$$  \hspace{1cm} (24)
given by:

\[ H = \frac{n-1}{n+1} + E[H_{BR}] \]

where \( E[H_{BR}] \) is the expected number of hops traversed by a packet in the BR permutation in the Hypermesh NoC, and is found in table 4 (which is determined experimentally in a Matlab program). The result of (25) is used to compute the effective arrival rate of flits to each component in (16).

8 Results

In this section, we compare the 2D Hypermesh NoC with the 2D layouts of the Generalized Hypercube (GHC) and Binary Hypercube (BHC). These topologies are described in Appendix A.

8.1 VHDL NoC Simulators

Two NoC simulators were designed and implemented in VHDL. Both simulators use wormhole switching. The first simulator implements a traffic pattern corresponding to a Pseudo-Random Walk (PRW) with no blocking and determines the maximum power consumption of an NoC. The second simulator uses credit-based wormhole switching, with ordered-dimension XY routing and round-robin arbitration, and is used to determine the power consumption given a specific traffic pattern.

Note that in the PRW simulator, the 2D Hypermesh is implemented with distributed crossbar switches as shown in Fig. 4c, since the control signals for the switches in the hyperedges are determined ‘off-line’ and are supplied to the switches during run-time. As a result, the Altera CAD tool chose to implement the switches without arbiters using the distributed switch layout shown in Fig. 4c, to minimize latency. In the wormhole simulator, the 2D Hypermesh was implemented using the centralized crossbar switches by the Altera CAD tool, (since each centralized switch in a hyperedge requires an arbiter to determine the multiplexer control signals). As a result, the Altera CAD tool realized the switches with arbiters using the centralized switch layout shown in Fig. 4b and Fig. 6, to minimize latency.

8.2 Experimental Measurements for NoCs

To validate the proposed analytic models, we compare the analytic results to extensive experimental results. In our experiments, each NoC is clocked at the same clock frequency \( f_{clk} = 50 \text{ MHz} \). The power consumed by the processors is not modelled, to focus on the power used only in the NoC topology.

The designs were created in the VHDL language and synthesized for the Altera Cyclone IV family of FPGAs, which use a 60 nm CMOS process and a 1.2v core. The NoC topologies were realized with 16 nodes. Each PE is a packet generator implemented using a linear feedback shift register (LFSR). The packet includes 32 flits, and each flit has a number of bits equal to the physical datapath width (16 bits). Each input buffer has capacity of 4 flits per physical channel, to minimize the buffer sizes and power used for buffers.

In order to tune our analytic models for the Altera Family of FPGAs, we synthesized some simple multiplexers and demultiplexers and performed measurements to determine the parameters shown in Table 2, for the Altera Cyclone IV FPGA.

Table 3 shows the discrepancy between the experimental and analytic results of area, maximum power, and maximum frequency for the 3 NoCs with 16 nodes. The PBS switch was used in the hyperedges. (All routers are small and use the BS switches.) Comparison between analytic and experimental results shows an excellent match, on average with 6% error.

8.3 Evaluation Methodology of the NoCs

In this sub-section, (1) the NoCs are compared based on their area and energy consumption, and (2) the NoCs are evaluated by their energy-area product. The 2D Hypermeshes use the 3 different switch designs presented in section 4 for the hyperedges, i.e., the BS, PBS, and PDM designs shown in Fig. 3. The 2D Hypermeshes are compared to the 2D layouts of the GHC and BHC. We assume that each PE uses 4K LEs, and a concentration of M=V=1 so that each node has one PE.

To have a fair comparison between the NoC topologies, each NoC is normalized to have an equal bisection bandwidth (BW) of O(N) bits/sec. We set the datapath width of GHC \((W_{GHC})\) to 16 bits, and adjust the
TABLE 4

<table>
<thead>
<tr>
<th>Topology</th>
<th>Results</th>
<th>Network Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHC</td>
<td>Hops $H_{BR}$</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Distance $L_{BR}$</td>
<td>2</td>
</tr>
<tr>
<td>GHC</td>
<td>Hops $H_{BR}$</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Distance $L_{BR}$</td>
<td>2.5</td>
</tr>
<tr>
<td>Hypermesh</td>
<td>Hops $H_{BR}$</td>
<td>1.5</td>
</tr>
</tbody>
</table>

widths of the BHC and Hypermesh to have datapath widths of $W_{GHC} = \frac{2}{\sqrt{N}} \cdot W_{BHC} = \frac{2}{\sqrt{N}} \cdot W_{HM}$. For example, for $N=16$ nodes, $W_{GHC} = 16$, $W_{BHC} = 32$, and $W_{HM} = 32$ bits.

To determine the power consumption of the NoCs under the FFT algorithm, we need to estimate the expected distance and hops traversed by a packet under the BR permutation. Table 4 shows the results for $E[H_{BR}]$ and $E[L_{BR}]$ in the BR permutation, which are determined experimentally in a Matlab program.

8.3.1 Area

Area is one of the most important design metrics of an NoC realized in an FPGA. One of the most important advantages of the Hypermesh topology over the Meshes, Torus and Hypercubes is its area-efficiency, and its high performance. Fig. 9a, 9b, 9c illustrates an area comparison of the 3 NoCs with 256 nodes under equal bisection bandwidth. The resource usages are expressed in terms of Logic Elements (LEs) and EM blocks. In Fig. 9a and 9b we assume that the input buffers are constructed using EM blocks, while in Fig. 9c the input buffers are constructed using DFF registers.

Most of the published studies of NoCs tend to minimize the amount of buffers, since the buffer memory often consumes significant amounts of power and area in the design. The Hypermesh NoCs use fewer buffers than the Meshes, Tori and Hypercubes, since the router degree is small and constant and equal to 3 (when the concentration $M = Y = 1$). For example for $N=256$ nodes and with equal bisection bandwidth, the 2D Hypermesh uses 33% of the EM blocks of the 2D BHC, and 38% of the EM blocks of the GHC.

As seen in Fig. 9, the 2D Hypermesh using the BS switch outperforms the 2D layouts of the BHC and GHC in terms of resource usage (LEs and EM blocks) by a significant margin.

Table 5 summarizes the area comparison between different NoCs with 256 nodes with different types of buffers under equal bisection bandwidth. As seen in table 5, the 2D layouts of the BHC and GHC are compared to the Hypermesh, using 3 different switch designs in the hyperedges. The values reported in this table are the percentages of the area that a 2D Hypermesh uses, when compared to the 2D layouts of the BHC and GHC. Under equal bisection bandwidth, the area usage (in terms of LEs) of the 2D Hypermesh with 256 nodes using the BS switch is 30% of the area of the GHC, and 42% of the area of the BHC.

TABLE 5

<table>
<thead>
<tr>
<th>Topology</th>
<th>HM-BS</th>
<th>HM-PBS</th>
<th>HM-PDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHC</td>
<td>Buffer</td>
<td>LEs</td>
<td>EM</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>35%</td>
<td>30%</td>
</tr>
<tr>
<td>BHC</td>
<td>EM</td>
<td>30%</td>
<td>35%</td>
</tr>
<tr>
<td>GHC</td>
<td>DFF</td>
<td>39%</td>
<td>-</td>
</tr>
<tr>
<td>BHC</td>
<td>DFF</td>
<td>42%</td>
<td>-</td>
</tr>
</tbody>
</table>

8.3.2 Energy Per Algorithm

In this sub-section, the NoCs are evaluated based on their energy consumption when performing the Bitonic sorting and the FFT algorithms. The completion time of an algorithm in an NoC is expressed in nanoseconds, and is given by:

$$T_{Alg} = E X_{Alg} T_{Clock}$$ (26)

where $T_{Clock}$ is the clock period ($1/F_{max}$) of the NoC, and $EX_{Alg}$ is defined as the execution time to perform the given parallel algorithm in clock cycles. The $EX_{Alg}$ (in terms of clock cycles) is given by:

$$EX_{Alg} = T_{P} N_{Perm} + T_{C} N_{Perm}$$ (27)

where $T_{P}$ is the average number of clock cycles needed to realize a permutation, $N_{Perm}$ is the number of permutations in the algorithm, and $T_{C}$ is the average time taken to perform a computation by a PE per permutation. For example, in the FFT algorithm $N_{Perm} = n + 1$, and in the Bitonic sorting algorithm $N_{Perm} = n(n + 1)/2$.

Fig. 10 shows the completion time of an algorithm ($T_{Alg}$) in the 2D Hypermeshes, and 2D layouts of the BHC and GHC with 256 nodes, when performing the Bitonic sorting and FFT algorithms (with equal bisection bandwidth). As seen in Fig. 10, the 2D Hypermeshes outperform the 2D layouts of the BHC and GHC by a significant margin. For example, when performing the FFT algorithm the 2D Hypermesh uses 42-98% of the time of the BHC, and 5-8% of the time of the GHC.

The energy consumption of an NoC given an algorithm is expressed by:

$$E_{Alg} = P_{Alg} T_{Alg}$$ (28)

where the $P_{Alg}$ and $T_{Alg}$ are the average power consumption and completion time of the NoC when performing the algorithm respectively.

All three topologies (Hypermesh, BHC and GHC) can implement all the butterfly permutations in the Bitonic and FFT algorithms without conflict and in one hop. In both algorithms, let each network have 256 nodes and let each packet have a size of 10 Kbits. Let the Input buffers be constructed using DFFs and have a capacity of 4 flits. Let the flit injection rate be 0.25 flit per node per clock cycle. Today, the performance of most digital systems is limited by their communication rather than computation latencies. Therefore, in our analysis we assume that the computation time $T_{C}$ is negligible compared to the network latency for a packet and $T_{C} \approx 0$ ns. In our
Fig. 9. Area analysis of the NoCs (a) in terms of LEs (when buffers use EM blocks), (b) in terms of EM blocks (when buffers use EM blocks), and (c) in terms of LEs (when buffers use DFF Registers) (for $N = 256$ nodes, with Equal Bisection Bandwidth, with BS switches).

Fig. 10. Completion time of the NoCs for the (a) Bitonic Sorting and (b) FFT algorithms (with $N = 256$ nodes, and Equal Bisection BW).

Table 6 summarizes the energy ($E$) and Energy-Area product ($E-A$) comparison between different NoCs with 256 nodes when performing the FFT and Bitonic sorting algorithms. As seen in table 6, the 2D BHC and 2D GHC are compared to the 2D Hypermesh using the 3 switch designs in the hyperedges. The values reported in this table are the percentages of the Energy and Energy-Area product product that a 2D Hypermesh uses, when compared to the 2D layouts of the BHC and GHC. Referring to table 6, the energy use of the 2D Hypermesh depends on the switches used in the hyperedges. To minimize energy use, the use of the PDM and PBS switches in the hyperedges are most efficient. As seen in table 6, the 2D Hypermesh outperforms the 2D layouts of the GHC and BHC in terms of Energy and Energy-Area product with a significant margin. The Energy-Area product of the 2D Hypermesh (for $N = 256$) is 9-18% of the 2D GHC, and 29-67% of the 2D BHC.

Table 6: Energy and Energy-Area product comparison between different NoCs

<table>
<thead>
<tr>
<th>NoC</th>
<th>FFT (E%)</th>
<th>FFT (E-A%)</th>
<th>Bitonic (E%)</th>
<th>Bitonic (E-A%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHC</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>GHC</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2D Hypermesh (PDM)</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>2D Hypermesh (PBS)</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

Thompson has shown that for all VLSI circuits that perform the FFT algorithm, the $AT^2$-product meets or exceeds a fundamental lower bound of $\Omega(N^2 \log^2 N)$ [25], where $A$ is the VLSI area of the circuit, and $T$ is the execution time. Following Thompson’s work on VLSI complexity, we outline as future work the determination of whether the Energy-Area product (or a variant) obeys any fundamental lower bound.
Fig. 11. Energy analysis of the NoCs for the (a) Bitonic sorting and (b) FFT algorithms (with \( N = 256 \) nodes, and Equal Bisection BW).

Fig. 12. Energy-Area product of NoCs when performing the (a) Bitonic sorting and (b) FFT algorithms (with \( N = 256 \) nodes, and Equal Bisection BW).

TABLE 6
Comparison of Energy (E) and Energy-Area Product (E-A) of NoCs for 2 algorithms (given \( N = 256 \) nodes, equal bisection BW, and 10Kbit packets)

<table>
<thead>
<tr>
<th>NoC</th>
<th>Alg.</th>
<th>HM-BS</th>
<th>HM-PBS</th>
<th>HM-PDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHC</td>
<td>FFT</td>
<td>41%</td>
<td>18%</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td>E-A</td>
<td>E</td>
<td>E-A</td>
<td>E</td>
</tr>
<tr>
<td>BHC</td>
<td>FFT</td>
<td>120%</td>
<td>55%</td>
<td>21%</td>
</tr>
<tr>
<td>GHC</td>
<td>Bitonic</td>
<td>58%</td>
<td>18%</td>
<td>23%</td>
</tr>
<tr>
<td></td>
<td>E-A</td>
<td>E</td>
<td>E-A</td>
<td>E</td>
</tr>
<tr>
<td>BHC</td>
<td>Bitonic</td>
<td>133%</td>
<td>67%</td>
<td>83%</td>
</tr>
</tbody>
</table>

9 CONCLUSIONS

Analytic models for the area, delay, power, and energy of 2D Hypermesh NoCs realized in the Altera family of FPGAs, have been developed. The analytic models are accurate, have a theoretical basis and allow for the early design-space exploration of hypergraph and Hypermesh NoC topologies, and for early design optimization. Three different switch designs for the hyperedges have been proposed and evaluated. The 2D Hypermeshes using the three switch designs have been compared to the 2D layouts of the BHC and GHC in terms of area, energy, and Energy-Area product. The Energy-Area product is proposed as a useful design metric to evaluate NoCs, which combines both the cost (i.e., power and area) and performance metrics (i.e., latency) of an NoC into one metric. Our analysis indicates that given equal bisection bandwidth, the 2D Hypermesh NoC will generally have considerably lower area, energy and Energy-Area product compared to the graph-based NoCs including 2D layouts of the BHC and GHC. Under equal bisection bandwidth, the area usage (in terms of LEs) of the 2D Hypermesh using the BS switches in the hyperedges is 30% of the area of the GHC and 42% of the area of the BHC. The Energy-Area product of the 2D Hypermesh when performing several common parallel algorithms (i.e., FFT and Bitonic sort) is 9% of the 2D GHC, and 29% of the 2D BHC. It has been shown that the Hypermesh NoC offer considerable benefits over conventional graph-based NoCs such as Meshes, Tori or Hypercubes, when using DWDM optical technologies [19]. Our analytic models establish that the Hypermesh NoC also offers considerable benefits over conventional graph-based NoCs such as Meshes, Tori or Hypercubes, in an FPGA or VLSI environment.

10 ACKNOWLEDGEMENTS

The careful reviews and comments of the editors and reviewers are appreciated.

REFERENCES

Ted H. Szymanski

Mohammadreza Binesh Marvasti received his M.Sc. degree in Computer Engineering from University of Tehran, Tehran, Iran, in 2007 and his PhD degree in Electrical and Computer Engineering from McMaster University, Hamilton, Canada, in 2013. His current research interests include computer architecture, low power digital design, FPGAs, and various aspects of NoCs, including power/area/delay modelling and evaluation.
