Appears in the 21st International Conference on Computer Design

Routed Inter-ALU Networks for ILP Scalability and Performance

Karthikeyan Sankaralingam Vincent Ajay Singh* Stephen W. Keckler Doug Burger

Computer Architecture and Technology Laboratory
Department of Computer Sciences
*Department of Electrical and Computer Engineering
The University of Texas at Austin
cart@cs.utexas.edu - www.cs.utexas.edu/users/cart

Abstract

Modern processors rely heavily on broadcast networks to bypass instruction results to dependent instructions in the pipeline. However, as clock rates increase, architectures get wider, and pipelines get deeper, broadcasting becomes more complex, slower, and more difficult to implement. This complexity is compounded by shrinking feature size, as the communication speed decreases relative to transistor switching speeds. This paper examines the fundamental needs of bypassing networks and proposes a method for classifying these Inter-ALU Networks based on how operands are routed from producers to consumers. We then propose and evaluate at both the circuit and architectural level a fine grain point-to-point Routed Inter-ALU Network (RIAN) that delivers the same or higher instruction throughput as a full bypass network but at higher speeds while using fewer wires.

1 Introduction

The most critical loop in pipelined processors enables data dependent instructions to execute in consecutive cycles. As shown in prior research [21, 4], increasing this path by even a single cycle dramatically reduces instruction throughput rates. Most modern processors, including both dynamically scheduled superscalar and VLIW architectures, use some form of broadcast to deliver instruction results to all the places that a consumer instruction could reside. The ALU execution delay plus the bypass latency to deliver the ALU output back to its input often sets the cycle time of the machine.

However, the complexity and delay of bypass paths is increasing with modern processors and technologies [14]. The wire delay path from source to destination, fan-out delay at the source, and fan-in delay at the destination are all increasing with increasing issue width and functional unit count. Wire complexity growth is proportional to the square of the number of ALUs, as shown by Ahuja et al. [2]. The fan-out from each source ALU and the fan-in at destinations increases roughly with the product of the pipeline depth and width, as each ALU result must be routed to all possible places it could be used. Larger fan-out and fan-in increases bypass delay as both the capacitive load within the network and the multiplexor complexity at each sink rises.

Based on optimistic wiring overhead models, we estimate that the shortest and longest bypass path delays, for a future, ultra-wide 64-issue processor with a 10F04 clock cycle, are 1 and 8 cycles respectively. In contrast, in many conventional processor designs the worst case bypass delay is small enough to be incorporated into the critical path and is a fraction of the clock cycle.

In this paper we analyze the fundamental requirements of inter-ALU networks (IANs) used in microprocessor bypass networks. We propose and evaluate an emerging class of bypass networks called Routed Inter-ALU Networks (RIANs) which scale with technology and functional unit count, and thus could become a viable alternative to broadcast bypass networks for future designs. In these networks, neighboring ALUs are connected via direct links through lightweight routers, and communication between ALUs is done by making multiple hops through the network. Instead of being broadcast, operands are routed from source to destination based on a destination identifier encoded into each instruction. RIANs reduce the fan-in and fan-out at each ALU, as well as the potentially crippling wiring area overhead by using a network with lower bisection bandwidth than a broadcast network. Such a network significantly reduces the bypass latency between nearby ALUs but may increase the latency between distant ALUs that must traverse many hops through the RIAN.

We present the full taxonomy of IANs in Section 2, classifying them based on (a) the number of target ALUs to which a result is delivered, (b) the number of ALUs to which a given ALU is directly connected, and (c) when the routing decision is made. The RIANs we propose can be classified as Point-to-point, Multi-hop, Dynamically routed networks, represented by the acronym **PMD**.

We discuss the applicability of these networks to dynamically scheduled monolithic and clustered processors where the destination of an operand is not always known. We then evaluate the use of this network in statically scheduled architectures in which the source and destination of each operand can be determined at compile time. We first explore their utility in wide clustered and non-clustered VLIW architectures. Finally, we examine this network strategy in an emerging architecture that supports static placement but dynamic issue to tolerate run-time determined instruction latencies. The key behind the applicability of point-to-point routing in all architectures is that results inherently need to be sent only from the producer to the consumer, rather than being broadcast to all ALUs.

The remainder of this paper is organized as follows. Section 2 describes the design space of inter-ALU networks, and discusses how they relate to prior bypass network architectures. Section 3 examines circuit implementations of high bandwidth single-hop networks used in conventional bypass networks and thin multi-hop networks used in a RIAN. We also describe mechanisms for

Execution Model	Network Architecture	Router Control	Acronym	Examples
Point-to-point	Multi-hop	Dynamic	PMD	Parcerisa et al. [15], Grid Processor [13]
Point-to-point	Multi-hop	Static	PMS	RAW [26]
Point-to-point	Single-hop	Dynamic	PSD	M-Machine [8], Multicluster [7]
Point-to-point	Single-hop	Static	PSS	degenerate case of PMS
Broadcast	Multi-hop	Dynamic	BMD	Alpha 21264 [11]
Broadcast	Multi-hop	Static	BMS	-
Broadcast	Single-hop	Dynamic	BSD	Superscalar [24]
Broadcast	Single-hop	Static	BSS	VLIW [5]

Table 1: A taxonomy of bypass networks.

reducing router overhead in this thin network. Sections 4, 5 and 6 explore the use of thin networks in dynamically scheduled superscalar architectures, VLIW architectures and Grid Processor architectures. Finally, Section 7 provides summary and concluding remarks.

2 A Taxonomy of Inter-ALU Networks

Bypass networks are intended to provide fast paths between the outputs of ALUs and inputs to prior stages of the pipeline downstream from the register file. Their primary effect on performance is to reduce or eliminate read-after-write hazards and possible pipeline stalls that result from issuing back-to-back producer and consumer instructions. In conventional processors, these bypass paths have typically been implemented as broadcast networks where the output of every ALU is routed to the input of every ALU.

These broadcast bypass networks are really a part of a broader class of Inter-ALU Networks (IANs) which can be classified along three axes: (a) the execution model, (b) the network architecture, and (c) router control. The execution model indicates whether the output of an ALU is to be *broadcast* by default to all ALUs, or whether the target ALUs are specified explicitly and then operands sent *point-to-point*. The network architecture indicates whether an operand is sent directly from the output of one ALU to the input of another (*single-hop*), or whether it may pass through intermediate routers (*multi-hop*). Router control indicates whether all of the routing decisions are made prior to execution of the ALU instruction producing the data (*static*), or whether the routing decisions take place at runtime (*dynamic*). These networks differ dramatically from multiprocessor networks because the payload is a scalar value rather than a multi-word message or cache line.

Table 1 lists the eight possible bypass network configurations and architectures which use them, with a 3-letter acronym for each network criterion: {P,B}, {M,S}, {D,S}. Pipelined and superscalar architectures are classified as **BSD** networks since operands are broadcast to all target ALUs, there are no intermediate routers, and all arbitration is done dynamically. The clustering of the Alpha 21264, in which operands are broadcast to both the local and remote cluster can be classified as **BMD**, since operands take one hop to reach a remote cluster, with arbitration done dynamically. Traditional VLIW processors with a shared register file namespace broadcast data across the ALUs through statically scheduled busses and is thus **BSS**.

As transistors have become faster and wires have become relatively slower, broadcast networks have become less attractive due to increasing wiring overhead for large connectivity networks. Several architectures have proposed or implemented one of the family of point-to-point IANs instead of using broadcast networks. The M-Machine is an example of a **PSD** network since destinations are specified statically and encoded in an instruction while delivery occurs dynamically from the source cluster to the destination cluster. The Multicluster architecture is also **PSD** as it dynamically routes operands on demand between two clusters in a partitioned register file superscalar architecture. The MIT RAW processor includes a bypass routing network which is integrated into the pipeline. The routing overhead is mitigated through a statically scheduled router which eliminates the need for dynamic arbitration for the shared router and wire resources, thus making it a **PMS** network. While this architecture achieved the per-hop latencies of a single cycle, their experience showed that these latencies were too high to achieve sufficient ILP. RAW requires a full cycle for each hop because the components that communicate are complete processors, rather than small ALUs, as found within a more conventional processor core [22].

Finally, a budding category of IANs is **PMD**: point-point, multi-hop, dynamically routed operand networks. Parcerisa et al. proposed a multi-hop routing network for clustered superscalar architectures with partitioned register files [15], similar in principle to Multicluster. The microarchitecture keeps track of the location of producers/consumers and dynamically inserts instructions to transmit operands from a source to a destination cluster.

In this paper, we focus on a different flavor of **PMD** networks in which the instruction dependencies are expressed explicitly in the instruction encoding and the physical locations of the producer and consumer instructions are known prior to execution. With this knowledge, bookkeeping hardware to dynamically track instruction dependencies is not required, nor are instruction results broadcast to every ALU, thus providing support for fast ALU chaining and technology scalability. We examine large networks of up to 64 ALUs and explore a range of topologies and connectivities.

A key feature for using routed point-to-point networks for operand bypass is the identification of producer-consumer pairs prior to execution of the producer, and scheduling of these pairs in nearby ALUs. While we focus on static compile-time schedulers to place instructions to minimize communication distance, runtime scheduling, such as the retire-time cluster assignment scheme proposed by Bhargava and John, is also possible [3].

A different taxonomy for scalar operand networks which consists of a 5-tuple cost model, closely based on network transport models was proposed by Taylor et al. [23]. The 5-tuple consists of send occupancy (# cycles ALU spends in transmitting a

value), send latency (# cycles at sender without consuming ALU cycles), network hop latency (# cycles spent in the network), receive latency and receive occupancy. Each of the class of networks in Table 1 can be characterized by a 5-tuple. Our axes of classification focus on the architecture and circuit characteristics of the operand network and are not specific to delay models. That 5-tuple model, although very expressive in terms of delays, is insufficiently expressive to characterize the diversity of operand interconnect networks completely. Different types of networks have the same tuple: for example **PSD** and **BMD** networks map to $< 0, 0, n, 0, 0 >$, while **BSD** and **BSS** networks as implemented in conventional superscalar and VLIW processors map to $< 0, 0, 0, 0, 0 >$.

3 Circuit modeling of Inter-ALU Networks

This section describes our circuit modeling of inter-ALU networks and explains the technology models, circuit estimation tools and different delay components in the circuit. In the proposed taxonomy of IANs, the key circuit feature is the *Network Architecture*. Both single-hop and multi-hop network circuits can be used in broadcast or point-to-point execution models, while the router control can be static or dynamic. The circuit design of the IAN is largely independent of the execution model and router control. In this section, we analyze *single-hop* and *multi-hop* network circuits in detail and compare their delays. We show that *multi-hop* networks outperform *single-hop* networks on large networks.

3.1 Technology modeling

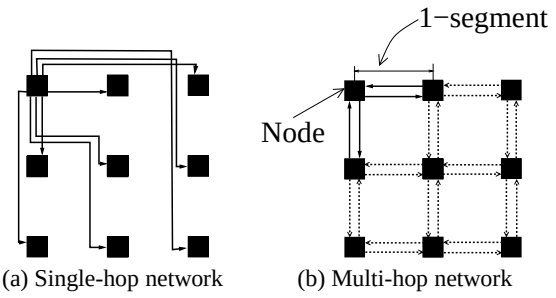
We estimated circuit latencies using SPICE models derived from the 1999 International Technology Roadmap for Semiconductors [17]. We estimated the wire delays assuming optimal buffer placement, with capacitance numbers obtained using Space3D (a three dimensional field solver) [1]. Technology parameters for the wire delay tool were based on the 2001 International Technology Roadmap for Semiconductors [18], using the 90nm technology point scaled to 100nm to match our SPICE libraries.

For our analysis, we assume that the functional units producing and consuming values are laid out in a 2-dimensional rectangular array with a Manhattan routing scheme. We refer to these functional units as *nodes*. All distances are measured in *segments*, with 1 segment being the distance between adjacent nodes. The *network size (N)* is the total number of nodes in the network. Figure 1 shows a single-hop and multi-hop inter-ALU network of size 9.

In our experiments, the node consists of an ALU, an integer multiplier, an FPU, and a 64-entry register file. All the functional units are 64 bits wide. The area and dimensions of these nodes are estimated using an empirical area model [9]. Each node is a square $32K\lambda$ on a side, occupying an area of $1G\lambda^2$, where λ is half the channel length of a minimum sized transistor. The processing core of the Alpha 21264 in comparison occupies an area of approximately $6G\lambda^2$.

3.2 Single-hop Inter-ALU Networks

There are three main components that contribute to the communication delay in single-hop networks: the fan-out gate delay (t_{fo}), the wire delay, and the fan-in gate delay (t_{fi}). The total delay for an n segment path is given by the following equation:



(a) Single-hop network (b) Multi-hop network

Figure 1: Single-hop and multi-hop networks of size 9. Only wires corresponding to the top left node are shown for (a).

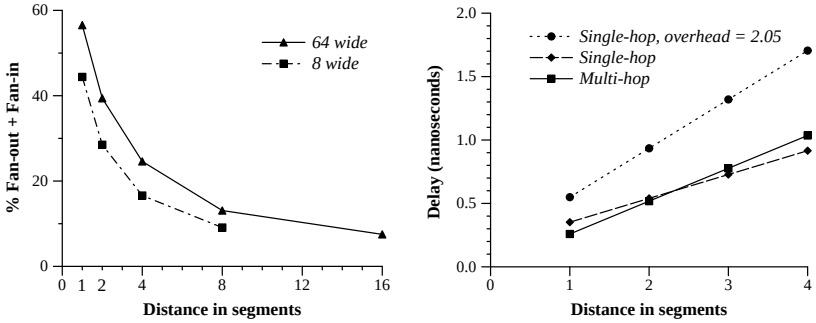
$$t_S = t_{fo} + t_{fi} + n * t_w * l * \alpha \quad (1)$$

The third term in the equation denotes the wire delay, which is the product of the number of segments traversed (n), wire delay per unit length (t_w), length of a segment (l), and the wiring distance overhead α. The wiring distance overhead is a factor used to incorporate the physical VLSI design constraints of wire routing. When the number of tracks required to route the wires fits within the area occupied by the ALUs, $\alpha = 1$, indicating no wiring overhead. However, when the wires require extra area for routing, α indicates the ratio by which the length of these wires increase, because of the excess area they must be routed over. This wiring overhead is strongly dependent on technology, ALU dimensions, data-path width, routing strategies and repeater placement and area. For a 64-bit data-path assuming a wire pitch of 16λ, our simple wiring area models, which do not account for any repeater area overhead, show that only single-hop networks of size greater than 32 incur any wiring overhead. For a network of size 64, $\alpha = 2.05$. All the multi-hop network configurations we examined have very low fan-outs (≤ 8) and hence incur no wiring overhead.

Delay analysis: Using SPICE simulations, the total fan-out and fan-in gate delays for 8-wide, and 64-wide networks was measured as 150ps, and 240ps respectively[1]. Figure 2a shows the percentage that each component contributes to the total communication latency for different distances. For both 8- and 64-wide configurations, the communication latency is evenly shared by the wire delay and the fan-out/fan-in delay for short distances. 57% and 44% of the delay is due to fan-in and fan-out for communication between immediate neighbors. Hence, reducing this fan-out and fan-in delay using a low fan-out network will produce significant reduction in short distance communication latency.

The total wire delay in nanoseconds, for an ideal 64-wide single-hop network assuming no wiring overhead is shown by the dashed line in the graph in Figure 2b. The dotted line shows the realistic delay, incorporating the wiring overhead estimation. A multi-hop network, in addition to having lower fan-in and fan-out

[1]By comparison an aggressive 10FO4 clock period at 100nm technology is 360ps [10].



(a) (b)

α	Crossover point (in segments) N=8	N=64
1.0	2	3
1.2	3	4
1.3	4	6
1.4	7	10
1.5	All distances	

(c)

Figure 2: Percentage wire delay contribution, total communication delay. The table shows the crossover point for single-hop and multi-hop networks. The crossover point indicates the maximum distance communication for which multi-hop networks are better.

delays, can reduce this wire delay also, since it has no wiring overhead, thus making the effective communication distances shorter.

3.3 Multi-hop Inter-ALU Networks

Multi-hop networks are defined as those networks that require routing decisions to be made in between the source and the destination nodes. The five parts of the delay are source router delay (t_{rs}), fan-out gate delay, wire delay, fan-in gate delay, and the intermediate router delay(t_{ri}). Compared to single-hop networks, the wire delay per segment is reduced because typically there is no wiring area overhead. The fan-in and fan-out delays are reduced since these are low-degree networks (typically with fan-in/fan-out of 4 to 8). The multi-hop network routes data through multiple nodes, incurring a router delay for every node that the data must pass through on the way to the destination node. We denote the number of hops as (h). Figure 1b shows a possible topology for a multi-hop network, with routers at every node. The total delay for a n-segment path on such a network is given by Equation 2.

$$t_M = (t_{rs} + t_{fo} + t_{fi} + t_{ri}) * h + n * t_w * l \quad (2)$$

The solid line in Figure 2b shows the variation of communication delay with distance for multi-hop networks. For short distance communication multi-hop networks are better than even the idealized single-hop network on a 64-wide network, whose delay is shown by the dashed line. When the wiring overhead of the single-hop network is included, as shown by the dotted line in the graph, multi-hop networks are better for all distances.

Router design: The source router and intermediate router delays are incurred because arbitration must be performed at each hop of the communication path to avoid resource hazards. Peh and Dally describe a latency-hiding approach for inter-chip networks using a lookahead flit reservation mechanism [16]. Based on their work, we evaluated a lookahead scheme for use in routers in RI-ANs to hide the arbitration delay. Two networks are implemented, one for control and one for the payload (the data operand). The control packet arrives in advance of the payload, and reserves a path (if one will be available) for the payload, thereby taking the routing and decision making logic off the critical data transmission path. The details of this design are beyond the scope of this paper and are explained elsewhere [19]. With the lookahead reservation scheme, the multi-hop delay effectively degenerates to the components shown in Equation 3.

$$t_M = (t_{fo} + t_{fi}) * h + n * t_w * l \quad (3)$$

Based on SPICE simulations we determined the router packet processing delay (t_{rs}, t_{ri}) to be approximately 9FO4 gate delays at 100nm technology. The fan-out/fan-in gate delay ($t_{fo} + t_{fi}$) on 4 fan-out network is 2.7FO4 gate delays. Effectively, the lookahead reservation scheme hides the packet processing delay from the data transmission critical path making the per-hop forwarding latency just the fan-out/fan-in gate delay, which is approximately 1/4th of a cycle, at a 10F04 clock period.

Delay analysis and implications: For short distances multi-hop networks are faster, because of the lower fan-out/fan-in delay and no wiring overhead. However, in a multi-hop network, a forwarding delay is incurred at each hop. For long distance multi-hop communication, this accumulated forwarding delay can overcome the fan-out/fan-in and wire delay savings, eventually making the multi-hop network delay larger than the single-hop network delay at some distance. Equating (1) and (3), we can determine this distance (n_c), measured in the number of segments, at which the single-hop network delay equals the multi-hop network delay. Communication delay is thus lower using a multi-hop network than a single-hop network, for distances less than n_c segments. This cross-over point varies based on the network size, since the fan-in, fan-out, and wiring overhead of single-hop networks is dependent on the size. The table in Figure 2 shows the crossover point for networks of size 8 and 64 for different values of α. For an idealized single-hop network assuming no wiring overhead, short distances of up to 2 and 3 segments in the 8 and 64 size networks respectively are faster using multi-hop networks. The crossover point increases super-linearly with α and when $\alpha \geq 1.5$, multi-hop networks are faster for communication between any two nodes in the network, regardless of the distance. These crossover points are specific to the machine configurations, technology and wiring overhead. Architects must consider these parameters while choosing the operand network design.

Based on this circuit analysis and delays, the following sections discuss the design and performance of single-hop and multi-hop networks in different processor architectures. We compare RI-ANs which are point-to-point, multi-hop, routed networks (**PMD**) to conventional broadcast single-hop networks (**BSS**, **BSD**). We examine dynamically scheduled superscalar architectures, VLIW architectures and Grid Processor architectures.

4 Dynamically Scheduled Architectures

Bypass delays are critical in conventional dynamically scheduled superscalar architectures. Current designs use a broadcast single-hop network among ALUs, since the execution model of such architectures does not provide information in the instruction stream to determine the consumers of the operands of each instruction. Future superscalar processors are likely to have large instruction windows and large numbers of functional units to extract more parallelism. Both the bypass delay and the instruction wakeup-select delay on such designs could become unmanageably large.

Partitioning the instruction window between clusters of tightly coupled functional units is one mechanism for keeping these two delays reasonably small, while increasing machine issue-width. Such clustered architectures typically use a broadcast single-hop network among the functional units in a cluster. Since the number of ALUs within a single cluster is small, this bypass delay is small compared to the cycle time. Between clusters, a separate network with variable inter-cluster communication delay can be used. In the absence of mechanisms to determine the ALU and cluster to which consumers of operands produced by other instructions have been assigned, every ALU must broadcast its results to every other ALU in the processor. Clustering by itself helps in only scaling the instruction window size and does not solve the broadcast problem.

To avoid having to broadcast operands, Parcerisa et al. proposed a clustered architecture in which an instruction steering stage of the processor pipeline assigns instructions to clusters, and inserts *copy* instructions to orchestrate point-to-point inter-cluster communication [15]. Within a cluster, operands are broadcast using a full bypass network (**BSD**). They studied several network interconnects and cluster configurations and concluded that a low bandwidth torus-like routed network (**PMD**) performed almost as well as an idealized full broadcast high bandwidth inter-cluster network (within 2%).

Based on the bypass network circuit characteristics alone, namely fan-in, fan-out, wire delay, and the wiring overhead, our circuit models suggest inter-cluster operand networks for large clusters will have lower delays when built as a single-hop network instead of a multi-hop network. However, port limitations on storage structures like remote register files in clusters impose limitations on the fan-in allowed at each cluster. In their work, Parcerisa et al. and others point out these constraints and show that RIANs are preferable in inter-cluster networks also.

5 VLIW Architectures

VLIW architectures are naturally amenable to point-to-point routing of operands using multi-hop networks because instructions are statically scheduled and producer-consumer pairs can be determined at compile time and encoded in the instructions. Conventional VLIW processors have used single-hop high bandwidth broadcast networks (**BMD**, **BSS**) to implement operand bypassing. In this section, we compare these networks to **PMD** networks and also evaluate the effectiveness of software schedulers in placing source and consumer nodes close to each other, which is a key feature for multi-hop networks.

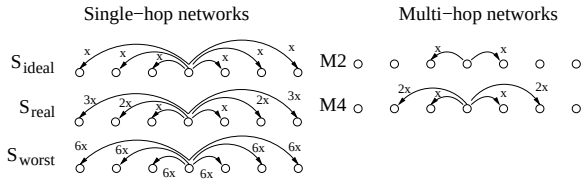


Figure 3: VLIW interconnect networks. The scaling factors on the arcs refer to the relative delays of the wires based on the distance traveled.

5.1 Machine model

We model a conventional VLIW machine where the compiler statically assigns instructions to named functional units (nodes) and decides the execution order of instructions. We used a greedy bottom up algorithm for generating the schedules. We examine a family of single-hop full bypass networks: 1) S_{ideal} - an ideal network where we set all wire delays to the shortest delay path, 2) S_{real} - a realistic best case single-hop network where we scale wire delays between nodes linearly with distance, and 3) S_{worst} - a worst case single-hop network, where we set all wire delays to the longest delay path. Conventional bypass networks resemble S_{worst}, since delay paths between all ALUs is set to be the worst case delay. We compare these single-hop networks with two multi-hop networks, one with only short paths, the *M2* network with wires between adjacent nodes, and one with medium distance paths, the *M4* network with wires to the nearest 4 neighbors. The diagrams of the connectivity are shown in Figure 3. We also simulated multi-hop networks with infinite bandwidth (infinite wires and ports between connected nodes) to study the effect of contention. To bound the sensitivity to the wiring overhead, we simulated single-hop networks with $\alpha = 1$ and $\alpha = 2$, and examined 4-wide, 8-wide and 16-wide machine configurations. Both the multi-hop networks we examined have very low fan-outs and we expect them to have no wiring area overhead. Hence we did not examine configurations with $\alpha = 2$ for these networks.

Equations (1) and (3) are used to obtain the delays used in the simulations. In all our simulations we assume a processor executing at a 10FO4 clock cycle in 100nm technology. At this technology, the router forwarding delay is 0.25 cycles and wire delay to traverse 1 segment (t_w) is 0.6 cycles (220 ps). We use a custom event-driven simulator to model the micro-architecture with the following parameters: functional unit latencies similar to an Alpha 21264; a 3-cycle, 64KB L1-cache; a 12 cycle, 2 MB L2-cache; and a 2-level history based branch predictor. The simulator models an aggressive lookahead resource reservation scheme implemented in our router. We assume that the data packet never catches up with the control packet, and hence only the fan-out/fan-in forwarding delay is incurred at every hop for the multi-hop networks, with the full router processing delay always hidden.

5.2 Benchmarks

To evaluate the performance of these networks on realistic workloads, we selected a set of benchmarks from the SPEC CINT2000, SPEC CFP2000, and three Mediabench [12] benchmarks — *gzip, mcf, parser, ammp, art, equake, dct, adpcm, and mpeg2encode*. We also examined one in-house benchmark, *radar* that performs

Config.	Latency (cycles) $\alpha = 1$	Latency (cycles) $\alpha = 2$	Contention (%)	# of Hops
4-wide VLIW				
S_{ideal}	0.13	0.43	0	1
S_{real}	0.36	0.81	0	1
S_{worst}	0.79	1.69	0	1
M2	1.06		26.4	1.2
8-wide VLIW				
S_{ideal}	0.17	0.51	0	1
S_{real}	0.78	1.59	0	1
S_{worst}	2.38	4.62	0	1
M2	1.72		23.2	1.5
M4	1.69		20.7	1.2
16-wide VLIW				
S_{ideal}	0.17	0.52	0	1
S_{real}	1.38	2.78	0	1
S_{worst}	5.35	10.54	0	1
M2	2.36		13.9	2.1
M4	2.05		11.7	1.5

Table 2: Interconnect network performance on VLIW architectures. Latencies shown in processor cycles, at a 10FO4 clock cycle. The single-hop networks, S_{ideal}, S_{real}, and S_{worst} have no contention since every pair of ALUs is connected by a dedicated wire. Also, the number of hops for them is 1.

radar signal-processing in which the computation is dominated by a 677-point complex FIR filter. We use the Trimaran tool set, which targets the HPL Play-doh ISA [25] to compile these benchmarks. We fast forwarded all benchmarks by five hundred million instructions, and then simulate two hundred million instructions.

5.3 Results

Routing Latency: Routing latency is the number of cycles between operand production and receipt at the destination. When the source and destination nodes are the same, we assume direct bypass in the execution cycle, and hence the routing latency is zero. This assumption makes the average latency shorter than the fastest transmission path through the network. The routing latency for the different machine configurations is shown in Columns 2 and 3 of Table 2. At width 4, the routed multi-hop network *M2* is worse than the S_{real} and S_{worst} networks since the network size is only 4. At larger machine widths of 8 and 16, the routed multi-hop network *M4* has routing latencies within 120% (1.69 versus 0.78) and 50% (2.05 versus 1.38) of the S_{real} network, and is always better than the S_{worst} network. When we incorporate the wiring overhead of $\alpha = 2$ for the single-hop S_{real} and S_{worst} networks, both the *M2* and *M4* networks are almost as good or better than them at all machine widths.

Contention: We measure the percentage contribution of the delay due to contention by measuring the percentage difference between the routing latency on a real multi-hop network and an idealized multi-hop network with infinite ports and wires between connected nodes. On this idealized network no delays are incurred due to resource hazards in the interconnect network. This percentage of latency due to contention is shown in Column 4. None of the single-hop networks have any contention, since they have a dedicated path between every pair of ALUs. The *M2* and *M4* networks show roughly the same amount of contention, with the higher bandwidth *M4* network always showing slightly less contention. This contention accounts for roughly 20% of the latency for the 8-wide machine and is about 11% for the 16-wide machine, suggesting higher bandwidth multi-hop networks could improve performance further.

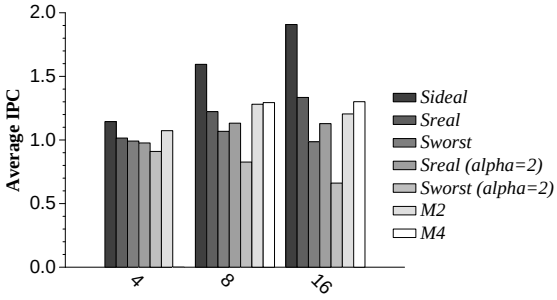


Figure 4: VLIW IPC averaged across the high IPC benchmarks dct, mpeg2encode and radar.

Number of Hops: The number of hops taken to route operands from source to destination indicates the effectiveness of the scheduler in placing producer-consumer pairs close together. For the conventional single-hop networks, number of hops is always one, since there is a dedicated wire from every node to every other node. As shown in Column 5 of Table 2, the average number of hops in the *M2* and *M4* networks is relatively low (≤ 2.1) compared to the machine width, showing that the scheduler is effective in placing producer-consumer pairs close together.

IPC: Figure 4 shows the IPCs averaged across only high performance benchmarks for the 4-wide, 8-wide and 16-wide configurations. Some benchmarks in our suite did not exhibit much improvement in performance when the machine width is increased. These benchmarks are not included in Figure 4, which shows the IPCs averaged across *dct*, *mpeg2encode* and *radar*. In these benchmarks, the performance with an idealized interconnect nearly doubles when the machine width is increased by a factor of 4 as shown by the S_{ideal} bar in the graph. Multi-hop networks are effective at extracting a significant fraction of this idealized performance and are almost as good or better than single-hop networks where wiring overhead is ignored. When we incorporate the wiring overhead of single-hop networks ($\alpha = 2$), the multi-hop networks are better at all machine widths. Performance details for individual benchmarks are presented in [20]. We also examined clustered VLIW processors, where our results showed that a inter-cluster routed multi-hop network connecting every $N/4$th node, with a full bypass intra-cluster network performed best, in an N-wide processor.

6 Grid Processor Architectures

Grid Processor Architectures (GPAs) use static placement but dynamically issue instructions. The three goals of this family of architectures are to extract high ILP, execute at a fast clock rate, and scale with technology. We use an array of ALUs with short

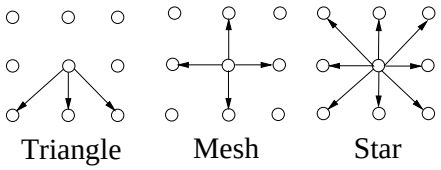


Figure 5: Multi-hop GPA Interconnects. Only the wires out of the center node of the representative 3x3 array is shown.

paths among them, mapping compiler-generated hyperblocks to this array. Multi-hop networks are ideally suited for this class of architectures for which the primary goal is to avoid global communication and extract performance from ALU chaining by mapping the critical path onto the shortest physical path. Previous work described the architecture and demonstrated the criticality of interconnect latency [13]. This section contains a more detailed analysis of the effects of latency and different network configurations on overall performance. Similar to the VLIW machine model, we simulate a perfect lookahead reservation scheme, hiding the router processing delay and incurring only the fan-out/fan-in forwarding delay of 0.25 cycles. Again, the *1 segment* delay was 0.60 cycles, and we simulate a 10F04 clock cycle.

We examine the same three parameters of performance as in the VLIW experiments, and investigated several different connection topologies on an 8x8 grid. This paper discusses only 3 representative multi-hop interconnect networks shown in Figure 5, while details on all the networks we investigated are presented in [20]. The networks range from very low fan-out (3), to moderately high fan-out (8). The *triangle* network connects 3 neighbors together, similar to the VLIW *M2* interconnect, while the moderately rich *star* network with a fan-out of 8, connects the immediate 8 neighbors together. For comparison we looked at the ideal, realistic and worst case single-hop networks. We examined wiring overhead factors of 1 and 2 to determine the wiring area effect on the single-hop, high-bandwidth networks.

Routing Latency: Column 2 of Table 3 shows the average routing latency in the grid network averaged across all the benchmarks. The realistic single-hop network S_{real} has a 2-cycle routing latency and achieves performance closest to the ideal network S_{ideal}. The very low fan-out *mesh* and *triangle* multi-hop point-to-point networks show significantly higher routing latency. The *star* network, with a fan-out of 8, is within 50% of the high bandwidth all-to-all S_{real} network. When we account for the wiring overhead for the single-hop networks (as shown in the last 2 rows in the table), they perform worse than the *star* network.

Contention: The contention in the network is again closely correlated with the fan-out of the topologies. The *star* network has the least contention (13%), while the *triangle* and *mesh* show much higher contention. This trend is to be expected because, as the interconnect richness increases, the latency due to contention decreases, making the *star* network the most efficient.

Number of Hops: As shown in column 4 of Table 3, the number of hops is also the least for the *star* network, with the *triangle* and *mesh* networks requiring slightly more. In all networks the number of hops is approximately 3 on the 64 node array, indicating effective scheduling. The *mesh* network has a higher average number of hops but a lower average latency simply because of the higher contention of the *triangle* network.

Network	Latency (cycles)	Contention (%)	# of Hops	Avgerage IPC
S_{ideal}	0.25	0	1	7.8
S_{real}	2.07	0	1	5.0
S_{worst}	7.85	0	1	2.3
Star	3.26	13	2.47	4.2
Mesh	5.11	27	3.17	3.7
Triangle	7.8	43	2.80	3.7
S_{real} (α = 2)	4.20	0	1	3.6
S_{worst} (α = 2)	15.58	0	1	1.6

Table 3: Grid processor interconnect network performance and instruction throughput.

IPC: Due to its dynamic issue capability, the Grid Processor achieves significantly higher IPC than the VLIW machines described in the previous section. Column 5 in Table 3 shows the IPC averaged across all of the benchmarks, using each of the interconnects. The fully connected single-hop network S_{real} achieves 63% of ideal performance (7.8 versus 5.0), when wiring overhead is ignored. The three multihop networks provide roughly the same performance, between 4.2 and 3.7, and approach the performance of the S_{real} network without wiring overhead. When the wiring area overhead is imposed on the single-hop networks, their performance drops significantly (from 5.0 average to 3.6 for the S_{real} network), making them worse than all the multi-hop networks. These results show that RIANs are effective and perform better than single-hop high bandwidth interconnects on very wide-issue architectures.

7 Conclusions

Dramatic increases in on-chip real-estate have driven architectures to scale the number of execution units in search of higher performance. However, traditional operand transmission networks that rely on broadcasting do not scale well with the technology constraints of faster transistors and slower wires. In addition, wiring overheads for broadcast networks scale poorly. In this paper, we have provided a taxonomy of Inter-ALU Networks (IANs) that includes traditional routing networks as well as emerging classes of point-to-point operand networks. The key components of these networks are their execution model (broadcast or point-to-point), their connectivity (single-hop or multi-hop), and when routing decisions are made (dynamically or statically). We have proposed a dynamically routed, point-to-point, multi-hop network, classified as **PMD** in our taxomony, and also called a routed inter-ALU network (RIAN) as a communication architecture scalable to tens of ALUs.

In our circuit analysis, we showed that these multi-hop networks scale much better than broadcast networks which suffer primarily from wire delays resulting from significantly larger area required for wiring. We designed and measured novel features of a router tailored to a fine grain RIAN including simple topologies and lookahead routing prior to data arrival. With these mechanisms, our measurements show that we can limit per-hop latency to 1 cycle in a 100nm technology with a 10F04 clock cycle. We applied these routing techniques to a conventional VLIW architecture and Grid Processor architecture, showing that operand broadcast is not necessary in these architectures and that existing

scheduling algorithms are effective at placing producers and consumers close to one another in such a network.

Our results show these RIANs outperform single-hop broadcast networks in both architectures when conservative wiring overheads are incorporated into the modeling of single-hop networks. These results are similar to Dally's conclusions on the design of k-ary interconnection networks for multicomputers [6]. Low-degree k-ary n-cube networks with express channels performed better than very high degree networks which are difficult to build because of physical wire limitations - the exact constraint faced in IANs in microprocessors. The lessons from the design of multicomputer networks may be applicable to scalar on-chip IANs. One important difference though, is that in multicomputer networks the node-delay (forwarding delay) is more than an order of magnitude higher than the wire delay, whereas in RIANs those two delays are comparable.

Future architectures must address the issue of operand bypass among a large number of ALUs. The RIANs we propose scale to tens of ALUs and are designed to work at fast clock rates. A key feature to the processor architectures which enabled our point-to-point routing strategy is the knowledge of producer and consumer instruction locations. While statically scheduled architectures naturally provide this support for RIANs, dynamically scheduled superscalar processors currently do not. We foresee these architectures also adopting RIANs by using dynamic rescheduling techniques and clustering.

References

[1] V. Agarwal, S. W. Keckler, and D. Burger. Scaling of microarchitectural structures in future process technologies. Technical Report TR2000-02, Department of Computer Sciences, The University of Texas at Austin, Austin, TX, February 2000.

[2] P. Ahuja, D. W. Clark, and A. Rogers. The performance impact of incomplete bypassing in processor pipelines. In *Proceedings of the 28th Annual International Symposium on Microarchitecture*, pages 36–45, November 1995.

[3] R. Bhargava and L. K. John. Improving dynamic cluster assignment for clustered trace cache processors. In *Proceedings of the 30th International Symposium on Computer Archictecture*, pages 264–274, June 2003.

[4] M. D. Brown, J. Stark, and Y. N. Patt. Select-free instruction scheduling logic. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pages 204–213, December 2001.

[5] R. P. Colwell, R. P. Nix, J. J. O'Donnell, D. B. Papworth, and P. K. Rodman. A VLIW architecture for a trace scheduling compiler. *IEEE Transactions on Computers*, 37(8):967–979, August 1988.

[6] W. Dally. Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks. 40(9):1016–1023, September 1991.

[7] K. I. Farkas, P. Chow, N. P. Jouppi, and Z. Vranesic. The multicluster architecture: reducing cycle time through partitioning. In *Proceedings of the 30th International Symposium on Microarchitecture*, pages 149–159, December 1997.

[8] M. Fillo, S. W. Keckler, W. J. Dally, N. P. Carter, A. Chang, Y. Gurevich, and W. S. Lee. The M-Machine Multicomputer. In *Proceedings of the 28th International Symposium on Microarchitecture*, pages 146–156, December 1995.

[9] S. Gupta, S. W. Keckler, and D. C. Burger. Technology independent area and delay estimations for microprocessor building blocks. Technical Report TR-00-05, Department of Computer Sciences, The University of Texas at Austin, Austin, TX, February 2001.

[10] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, April 2001.

[11] R. E. Kessler. The Alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, March 1999.

[12] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of 30th Annual International Symposium on Microarchitecture*, pages 330–335, December 1997.

[13] R. Nagarajan, K. Sankaralingam, D. C. Burger, and S. W. Keckler. A design space evaluation of grid processor architectures. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pages 40–51, December 2001.

[14] S. Palacharla, N. P. Jouppi, and J. Smith. Complexity–effective superscalar processors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 206–218, June 1997.

[15] J. M. Parcerisa, J. Sahuquillo, A. González, and J. Duato. Efficient interconnects for clustered microarchitectures. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pages 291–300, September 2002.

[16] L. S. Peh and W. J. Dally. Flit–reservation flow control. In *Proceedings of the 6th International Symposium on High–Performance Computer Architecture*, pages 73–84, January 2000.

[17] The National Technology Roadmap for Semiconductors. Semiconductor Industry Association, 1999.

[18] The International Technology Roadmap for Semiconductors. Semiconductor Industry Association, 2001.

[19] V. A. Singh, S. W. Keckler, and D. Burger. A routing network for the grid processor architecture. Technical Report TR-03-10, The University of Texas at Austin, April 2003.

[20] V. A. Singh, K. Sankaralingam, S. W. Keckler, and D. Burger. Design and analysis of routed Inter-ALU Networks for ILP scalability and performance. Technical Report TR-03-17, The University of Texas at Austin, July 2003.

[21] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *Proceedings of the 29th International Symposium on Computer Archictecture*, pages 25–34, May 2002.

[22] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, W. L. Jae-Wook Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, March 2002.

[23] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal. Scalar operand networks: On-chip interconnects for ILP in partitioned architectures. In *Proceedings of the 9th International Symposium on High Performance Computer Architecture*, pages 341–353, February 2003.

[24] R. Tomasulo. An efficient algorithm for exploiting multiple arithmetic units. *IBM Journal Research and Development*, 11:25–33, January 1967.

[25] V.Kathail, M.Schlansker, and B.R.Rau. Hpl-pd architecture specification: Version 1.1. Technical Report HPL-93-80(R.1), Hewlett-Packard Laboratories, February 2000.

[26] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarsinghe, and A. Agarwal. Baring it all to software: RAW machines. *IEEE Computer*, 30(9):86–93, September 1997.