Design and Analysis of Routed Inter-ALU Networks for ILP Scalability and Performance

Vincent Ajay Singh* Karthikeyan Sankaralingam 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 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 fundamentals needs of bypass 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 instruction throughput as a full bypass network but at higher speeds and using fewer wires.

1 Introduction

The most critical loop in pipelined processors enables data dependent instructions to execute in consecutive cycles. In fact, 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. As shown in prior research [16, 3], increasing this path by even a single cycle dramatically reduces instruction throughput rates. Most modern processors, including both superscalar and VLIW architectures, use some form of broadcast to deliver instruction results to all places that a consumer instruction could reside.

However, complexity and delay of bypass paths is increasing with modern processors and technologies [11]. Wider-issue machines with conventional broadcasting techniques incur a wire complexity growth proportional to the square of the number of ALUs [2], thus contributing to both increased wiring area and larger fan-in at the bypass targets. The fan-out from each source ALU 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 the both the capacitive load within the network and the multiplexor complexity at each

Execution Model	Network Architecture	Router Control	Acronym	Examples
Point-to-point	Multi-hop	Dynamic	PMD	Parcerisa [12], Grid Processor [10]
Point-to-point	Multi-hop	Static	PMS	RAW [20]
Point-to-point	Single-hop	Dynamic	PSD	M-Machine [6], Multicluster [5]
Point-to-point	Single-hop	Static	PSS	degenerate case of PMS
Broadcast	Multi-hop	Dynamic	BMD	Alpha 21264 [8]
Broadcast	Multi-hop	Static	BMS	-
Broadcast	Single-hop	Dynamic	BSD	Superscalar [18]
Broadcast	Single-hop	Static	BSS	VLIW [4]

Table 1: A taxonomy of routed bypass networks

sink rises. Finally, increases in wiring resistance increase transmission latencies, particularly to pipeline stages and ALUs that are far from the source ALU. 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.

To reduce these delays and improve the scalability of broadcast bypass networks we propose and evaluate a new class of Routed Inter-ALU Networks (RIANs). In these networks, neighboring ALUs are connected via direct links through lightweight routers, and communication between distant ALUs must make 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. Such a network significantly improves the bypass latency between nearby ALUs but may increase the latency between distant ALUs that must traverse many hops through the RIAN.

In general, bypass inter-ALU networks (IANs) can be classified according to (a) the number of target ALUs to which a result is delivered, (b) the number of targets to which a given ALU is directly connected, and (c) when the routing decision is made. While we present the full taxonomy of IANs in Section 2, the RIAN networks we propose can be classified as a Point-to-point, Multi-hop, Dynamically routed networks, represented by the acronym **PMD**. We evaluate the use of this network in statically scheduled architectures in which the source and destination of each communication are determined at compile time. We first explore their utility in wide clustered and non-clustered VLIW machines in which the targets and links can be identified at compile time, but routing and arbitration is performed at run time.

We then examine this network strategy in an emerging architecture that supports static scheduling but dynamic execution to tolerate run-time determined latencies. The key behind the applicability of point-to-point routing in both architectures is that results inherently need to be sent only from the producer to the consumer, rather than being broadcast to all ALUs. We show that scheduling algorithms are effective in placing producers and consumers near one another, thus restricting the communication distance to a few hops in the common case. Multi-hop point-to-point networks are efficient for these patterns of communication.

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 multi-hop switched networks and describes mechanisms for reducing router overhead in this thin network. Section 4 explores the use of thin networks in VLIW architectures, while Section 5 does the same for the dynamically executed Grid Processor architecture.

Finally, Section 6 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 prime 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 have typically been implemented as broadcast networks where essentially the output of every ALU is routed to the input of every other ALU.

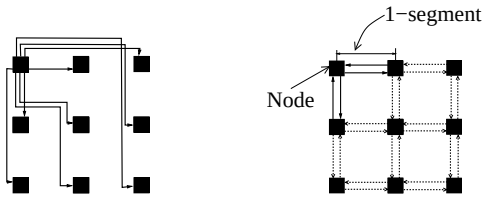
These broadcast bypass networks are really a part of a broader class of Inter-ALU Networks (IANs) which can be classified along three axes: (a) the execution model, (b) the network architecture, and (c) router control. The execution model indicates whether the output of an ALU is to be *broadcast* by default to all ALUs, or whether the target ALUs are specified explicitly prior to execution of the instruction and then routed *point-to-point*. The network architecture indicates whether an operand is routed 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*). Note that 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 the routing does not commence until the ALU operation is complete. The clustering of the Alpha 21264, in which operands are broadcast to both the local and remote cluster can be classified as **BMD**. Traditional VLIW processors with a shared register file namespace broadcast data across the ALUs though 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 long wire lengths and increasing wiring overhead for large connectivity networks. The major challenge in such networks is to reduce the latency for communication to a level equaling or approaching conventional bypass networks. The two components required to achieve this are: (a) the network interface must be integrated into the pipeline so that operands are delivered directly to consuming ALUs and results are injected into the network directly from ALU outputs, and (b) the latency to route through the network must be minimized.

Several architectures have proposed or implemented one of the family of point-to-point IANs to meet these goals. 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 rendering 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, in part because the components that communicate are complete processors, rather than small ALUs, as found within a more conventional processor core [17].

Finally, a budding category of IANs is **PMD** – point-point, multi-hop, dynamically routed networks. Parcerisa, et. al proposed a multi-hop routing network for clustered superscalar architectures with partitioned register files, similar in principle to Multicluster [12]. The microarchitecture keeps track of the location of produc-



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

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

ers/consumers and dynamically inserts instructions to transmit operands from a source to a destination cluster. They evaluate small scale networks of up to 8 clusters using ring, mesh, and torus topologies.

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 producing and consuming instructions are known prior to execution. With this knowledge, bookkeeping hardware to dynamically track instruction dependencies is not required, nor do instruction results need be broadcast to every ALU. We do not restrict ourselves to statically scheduled architectures since dynamic behavior such as variable load/store latencies must be tolerated at runtime. We examine large networks of up to 64 ALUs and explore a range of topologies and connectivities.

3 Circuit modeling of Inter-ALU Networks

In this section we describe our modeling of inter-ALU networks explaining the following aspects — $1)$ the technology models and circuit estimation tools, $2)$ conventional bypass networks and their different delay components, and $3)$ the scalability of these networks. We propose point-to-point networks as an alternative when large numbers of communicating nodes are required, with the communication being mostly amongst adjacent nodes. The router design is crucial for network throughput in point-to-point networks, and we discuss a routing protocol and router design that hides this latency from the network. Finally we compare the performance of multi-hop networks and single-hop networks. In the experiments described in this section, we examine the generic class of switched multi-hop networks (which could be implemented as PMD, PMS, or BMD) and single hop networks (which could be implemented as PSD, PSS, BSD, or BSS). The programming and execution models respectively determine whether the communication is point-to-point/broadcast and whether the router control is static/dynamic. We then address the tradeoffs between the two types of networks.

3.1 Technology modeling

We estimated circuit latencies using SPICE models derived from the 1999 International Technology Roadmap for Semiconductors [14]. 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 [15], using the 90nm technology point scaled to 100nm. We refer to the wire delay obtained (represented in picoseconds per mm) as t_w.

For our analysis, we assume that the functional units producing and consuming values are laid out in a 2-

Wire delay	117 ps/mm				
Node area	2.54M square microns, 1Gλ^2				
Network size (in nodes)	4	8	16	32	64
Fanin+Fanout delay (ps)	100	150	175	210	240

Table 2: Network delay components at 100nm.

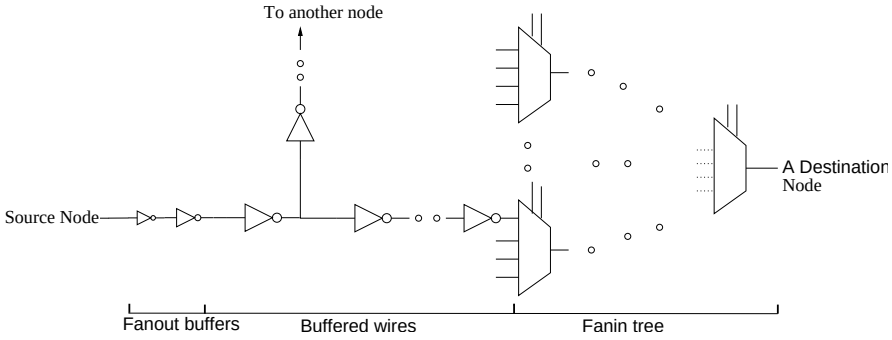


Figure 2: Circuit for a bypass path.

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 [7]. 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 $6G\lambda^2$. Table 2 shows the wire delay and the node area obtained using our circuit tools for 100nm technology.

3.2 Modeling conventional bypass networks

Conventional broadcast networks fall under the BSD class of networks. A general communication path used in such bypass schemes is shown in Figure 2. No network topology decisions or routing protocol decisions are reflected in this abstract model. As shown in the figure, there are three main components that contribute to the bypass delay: the fan-out delay (t_{fo}), the wire delay, and the fan-in delay (t_{fi}). The total delay is given by the following equation.

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

The third term in the equation denotes the wire delay — 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

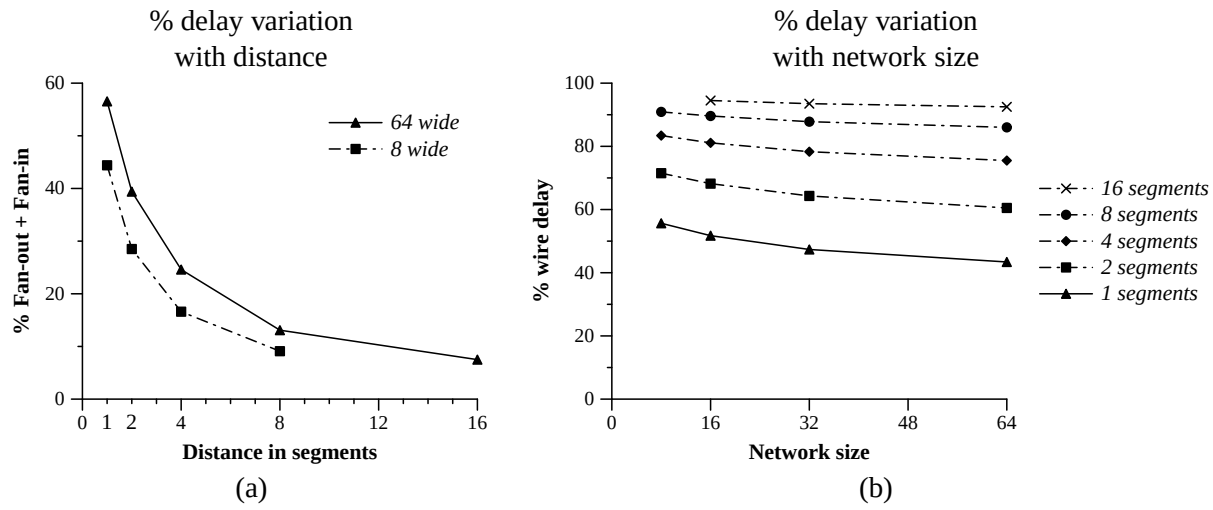


Figure 3: Percentage wire delay contribution to the total communication delay.

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 fanouts (≤ 8) and hence incur no wiring overhead.

A layout corresponding to this circuit for a 3x3 single-hop network is shown in Figure 1a. Only the outbound wires originating from the top left node are shown. The fanin and fanout delays correspond to the delays of the destination multiplexors and the fanout buffers at the source respectively. These delays were obtained using SPICE simulations for different network sizes and the values we obtained are shown in Table 2.

Delay analysis: In a single-hop network, large fan-out and fan-in delays are incurred once for every communication. Figure 3a shows the percentage contribution of the various components to the total communication latency in networks with varying network sizes and distances traversed. For both 64 and 8-wide configurations, the communication latency is evenly shared by the wire delay and the fan-out/fan-in delay for communication over short distances. 57% and 44% of the delay is due to fanin and fanout for communication between adjacent nodes one segment away. Hence, reducing the fanin+fanout contribution can have significant benefits for short distance communications. On the other hand, wire delay dominates for long distance communications.

Figure 3b shows the percentage contribution of wire delay for communicating over different distances in networks of different size. Note that as we increase the network size i.e. the total number of nodes in a network, but keep the communication distance the same, the fanin+fanout delay increases logarithmically, but the wire delay remains constant - hence the percentage contribution of wiredelay drops. As seen in the graph, for a 1-segment path the wire delay contribution drops from 53% to 43% when the network size changes from 8 to 64. On the other extreme for a 16-segment path (the longest path in the network), the wire delay contribution drops marginally from 94% to 92%. Hence, architectures that incur frequent long-distance communications among many nodes should stick to conventional single-hop networks because of the ease of design and better performance.

Recall that in Figure 3a, we plotted the percentage contribution for fanin+fanout as the communication dis-

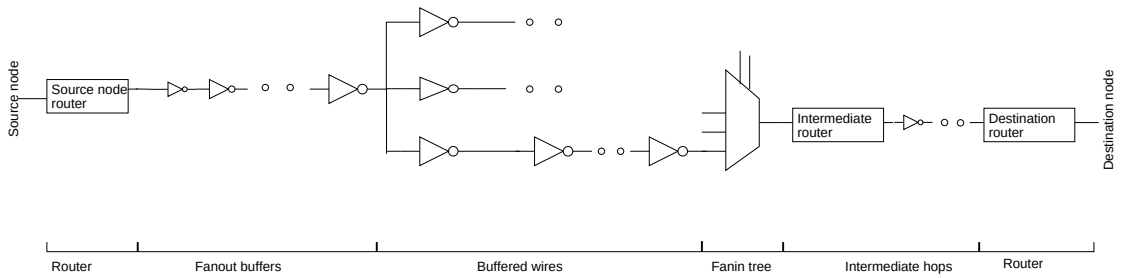


Figure 4: Circuit for a Multi-hop network.

tance varied and showed the results for 2 network sizes 64 and 8. This serves as the motivation for mutli-hop networks which are well suited to architectures that exhibit frequent short distance communication.

3.3 Multi-hop Inter-ALU Network

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 for a multi-hop configuration are shown in Figure 4. They are outgoing router delay(t_{rs}), fan-out delay, wire delay, fan-in delay, and the intermediate router delay(t_{ri}). The wire delay incurred here is identical to the wire delay seen in single-hop networks, and the fan-in and fan-out delays are both dependent on the richness of the interconnect. However, the multi-hop network routes data through multiple nodes, causing a router delay for every node that the data must pass through on the way to the destination node, i.e. the number of hops (*h*). This can be seen in Figure 1b, which shows a possible topology for a multi-hop network. The total delay is given by the following equation.

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

Since multi-hop networks typically have only a small number of connections between neighbors (relying on multi-hop routing for non-local connections), they are well suited to architectures where communication is predominantly between nearby ALUs. It is crucial to optimize the routers since every communication beyond the first hop requires a routing operation.

3.4 Router design

To overcome the challenges posed by technology scaling on large single hop networks, multi-hop networks are an attractive alternative because of their fine grain control and low wiring overhead. The fan-out, wire delays, and the fan-in delays are inherently serial and cannot be removed from the critical path. The router delay at either end of a communication path is incurred because arbitration must be performed to avoid resource hazards. We propose to use a lookahead scheme to hide the arbitration delay. In order to do this, two networks are implemented. One for control and one for payload (the actual data operand). The control 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 path. If no path will be available (due to contention through the node) a buffer slot is reserved for the incoming operand. This information is available a cycle in advance of the payload, since the destination is encoded into the instruction itself and can be processed during the time taken to produce the operand. With this advance knowledge, circuit techniques (such as domino logic) can be used to increase the speed of the operand transmission. Peh *et al.* describe a similar latency-hiding approach for inter-chip networks [13].

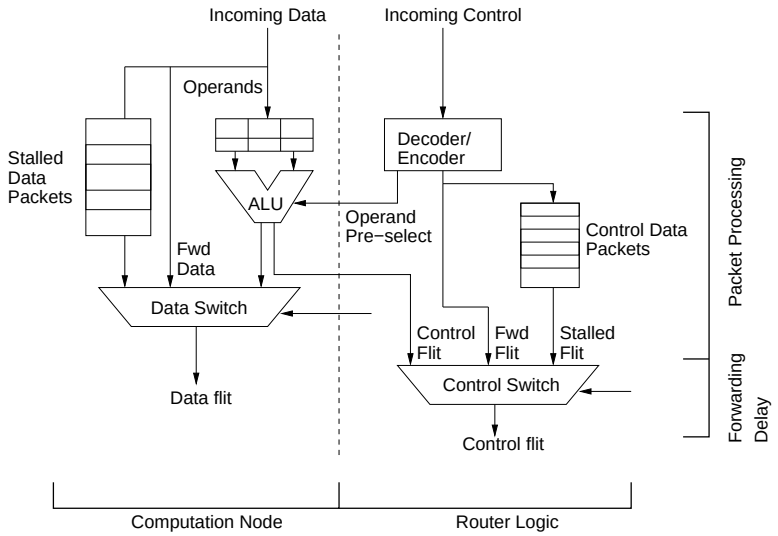


Figure 5: Router schematic

Advance knowledge of incoming operands also allows an efficient flow control mechanism. With knowledge of how many open buffer slots are available and the communication latency to immediate producer nodes, a simple throttling mechanism can be implemented. Assuming that one cycle is required to send an operand from one node to the next, the throttling signal is asserted when a control flit arrives at a buffer having having only two slots free on data arrival. The router requires one slot for the operand arriving on the next cycle (the payload corresponding to the control signal), and one for the operand that could be sent while the throttle signal is traveling back to the producing node. Once the producing node receives the throttle signal, it will cease to transmit data until the stall signal is deasserted. Thus, in the common case, a producing node can send the operand without the need to receive an acknowledgment, since it is guaranteed that storage space will be available at the consumer node if there is contention on the routing path. Only in the uncommon case is any backward information required.

Figure 5 shows the schematic of the router. The key component of the router is the Decoder/Encoder that looks at control packets and steers them either to the control switch for forwarding to neighboring nodes, or sets up the ALU datapath, to be ready to receive a value meant for this node, in the next router cycle. Stalled control and data packets are written to separate buffers and a few cycles before they become full, nodes upstream are throttled. When packets are forwarded, only a small forwarding delay is incurred, while a much larger packet processing delay needs to paid for packets created by the ALU. The router circuit was modeled using our circuit tools to determine these delays. The packet processing delay was 300ps, and the forwarding delay was 100ps in our circuits at 100nm technology. Thus using this router, the multi-hop delay in Equation 2 is transformed as:

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

3.5 Delay analysis and implications

The delays for the two types of networks are represented using the delay equations built with the core circuit components. Equating the two delays, we can define the cross-over point (n_c): the number of hops at which a multi-hop network outperforms a single-hop network as shown in Equation 4. We assume the router delay

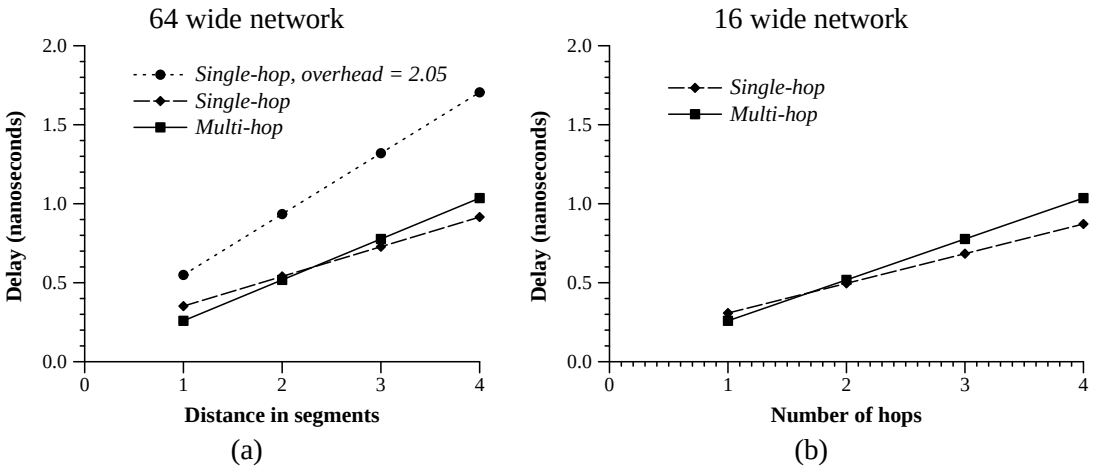


Figure 6: Wire delays in nanoseconds for full bypass and point to point networks. a) Network size of 64 ALU nodes. b) Network size of 16 ALU nodes. Overheads estimated using wiring overhead model.

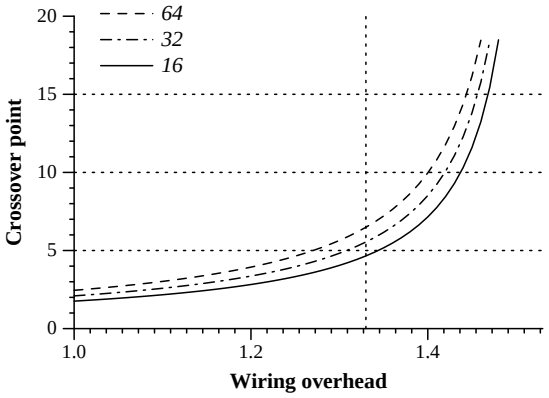


Figure 7: Crossover point.

can be fully hidden, and hence does not figure in the equation. The crossover occurs when the fan-in/fan-out overhead (accumulated over n hops) in a multi-hop network exceeds the single fan-in/fan-out delay of a full broadcast single-hop network. The subscripts S and M are used to denote single-hop and multi-hop networks respectively.

$$n_c = \frac{(t_{fo_S} + t_{fi_S})}{(t_w * (1 - \alpha) + (t_{fi_M} + t_{fi_M}))} \tag{4}$$

Figure 6 shows the variation of the wire delays with number of hops for multi-hop and single-hop networks. For a 64 node array, assuming no extra wiring overhead (i.e. $\alpha = 1$), a multi-hop network has less latency when communicating over one to two segment distances. For a 16 node network, n_c shifts to a little less than 2. Using our circuit models we calculated the delays for up to 16 segments; for a 14-segment trip, the multi-hop network is only 30% slower than a full-bypass network, 25% *faster* for a 1-segment trip, and 5% faster for a 2-segment trip. It is crucial to keep the more common short paths as fast as possible, while, in general, latencies along less common paths are less important. Thus a multi-hop network will be favored if most of the communication can be orchestrated among neighboring nodes. When the overhead is taken into account, a 64-wide single-hop network never outperforms a multi-hop network as the dotted line has a higher slope, and larger y intercept, than the solid line.

Figure 7 plots Equation 4 for three different single-hop network sizes using the fan-out, fan-in and wire delays obtained from our circuit models. The sensitivity to wiring overhead is significant, as indicated by the asymptotic nature of the curve. This graph is particularly important since our wiring area estimates are conservative (we do not account for repeater placement area). As can be seen from the graph, if the wiring overhead of broadcast networks resulted in $\alpha = 1.25$ for example, the crossover point is reasonable large — 5. If the insertion of repeaters results in the wiring overhead exceeding 1.4, then multi-hop networks will always outperform single-hop broadcast networks. Hence interconnect physical design issues are crucial and can have a significant impact on the interconnect architecture.

4 VLIW Architectures

In VLIW architectures, where the routing and arbitration is done at run time, the delay of the ALU bypass network is a critical loop. As illustrated in the previous section, a choice has to be made about the network architecture depending on the number of hops traveled in the common case. We examine the design space of bypass networks in the context of unclustered VLIW architectures. In VLIW machines, the location of producer and consumer instructions is known at compile time, and a multi-hop network can be used to route packets from source nodes to destination nodes. Provided these are close to each other, a machine with a multi-hop network can sustain higher instruction throughput than a single-hop broadcast network, since the execution of data dependent operations in consecutive cycles is critical. In this section, we first describe our machine model and configurations studied. We then describe our benchmark suite and the compilation tools used. To examine the impact on future designs we examined very wide issue machines of width 16. We examined machine widths of 4 and 8 to determine the applicability of multi-hop networks in current designs.

4.1 Machine model

We model a VLIW machine where instructions are statically assigned to named functional units (nodes). The compiler also generates the schedule for the execution order of the long instruction words. The individual instructions in each instruction word are allowed to execute in any order and are independent. We examine

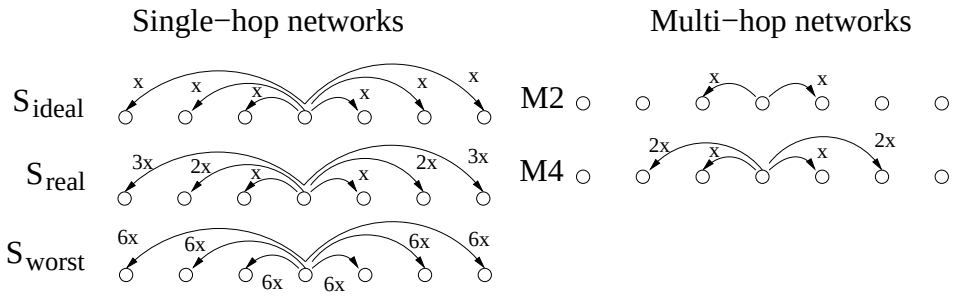


Figure 8: VLIW Interconnects.

multi-hop networks and single networks for bypassing values. When a single-hop network is used for bypass, the values go directly from the source node to the destination node through dedicated paths. When a multi-hop network is used, the values are dynamically routed through the network from the source to the destination. 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}. 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 8. We simulated multi-hop networks with infinite bandwidth (infinite wires and ports between connected nodes) to study the impact of contention. To bound the sensitivity to the wiring overhead, we simulated single-hop networks with $\alpha = 1$ and $\alpha = 2$. We simulated 4-wide, 8-wide and 16-wide machine configurations. Furthermore, fanin/fanout contribution is accounted for in all networks when determining the total delays.

The delays used in the simulations are derived from our circuit models and equations (1) and (3). For example, the nodes in the M2 model and the closer nodes (1x distance away) in the M4 model would incur a total delay of 320 ps (100 ps fan-in/out plus 220 ps to traverse one node.). The farther nodes in the M4 model would incur a delay of 540 ps, as the wire distance is twice as much. The conventional bypass networks were simulated with α equal to 1 and 2 in order to better understand the effect of wiring overhead. Additionally, all the simulations were also run with infinite bandwidth (no contention for links) in order to see what percentage of the latency was due to contention.

Our simulations assumed a processor executing at a 10FO4 clock at 100nm, making each router forwarding delay 0.27 cycles (100 ps) and the time to simply traverse a node length wire (t_w) 0.6 cycles (220 ps). It should be noted that a real machine could not support arbitrary delays, as the circuits are synchronized to the ALU clock. Accordingly, we assumed the routers would be clocked at 4X the ALU clock (quad-pumped) and rounded the delays to the nearest quarter cycle.

4.2 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 [9] benchmarks — gzip, mcf, parser, ammp, art, equake, dct, adpcm, and mpeg2encode. We also examined one in-house benchmark that performs radar signal-processing where the computation is predominantly a 677-point complex FIR filter. The Trimaran tool set, which targets the HPL Play-doh ISA [19] is used to compile these benchmarks. We use a custom built scheduler that

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 3: 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, # of hops for them is 1.

is aware of all the delay paths in the architecture and optimizes the local critical path. We use a custom event-driven simulator to model the micro-architecture. The performance 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 there is no contention while transmitting the control packets. Hence we always pay only the constant router forwarding delay (100ps) at every hop for the multi-hop network, and never incur the full packet processing delay of 300ps. All benchmarks were forwarded five hundred million instructions, and then simulated for two hundred million instructions.

4.3 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 3. 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

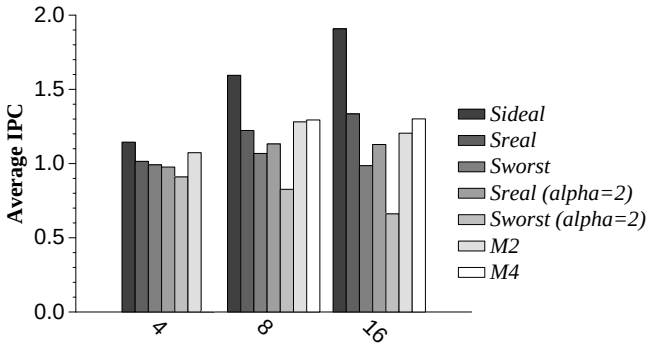


Figure 9: IPC averaged across the high IPC benchmarks dct, mpeg2encode and radar on VLIW machines.

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 as expected. 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.

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 3, 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 9 shows the IPCs averaged across only the 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 9, which shows the IPCs averaged across *dct*, *mpeg2encode* and *radar*. In these benchmarks, the performance with an idealized interconnect 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. 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.

Figure 10 shows the performance of each of the individual benchmarks for 4, 8 and 16-wide machines. The low IPC benchmarks show only little variation in performance as the machine width and interconnect network are varied. Figure 11 shows the performance of the benchmarks, when the wiring overhead is accounted for, by assuming α equal to 2. In these configurations the multi-hop networks always outperform the single-hop configurations.

5 Grid Processor Architectures

Grid Processor Architectures (GPAs) use static placement but dynamically issue instructions. The goal in this architecture is to extract high ILP, execute at a fast clock rate, and scale with technology. We use an array of

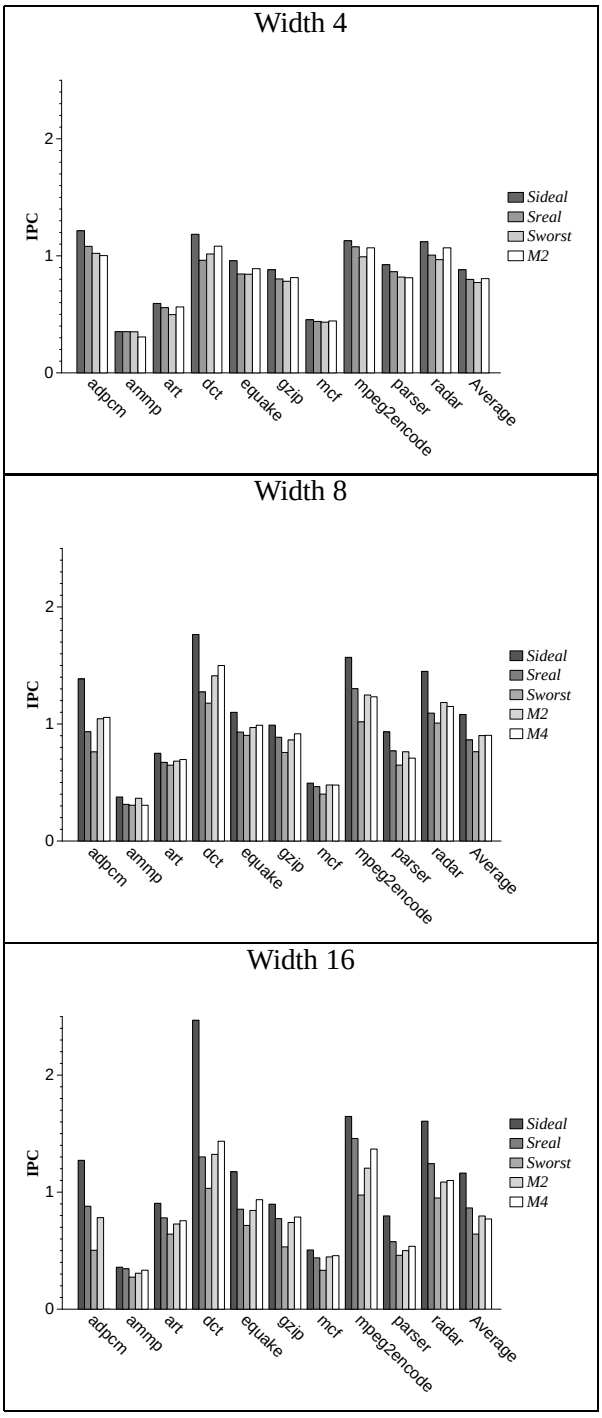


Figure 10: IPC on VLIW machines: No wiring overhead for single-hop and multi-hop networks.

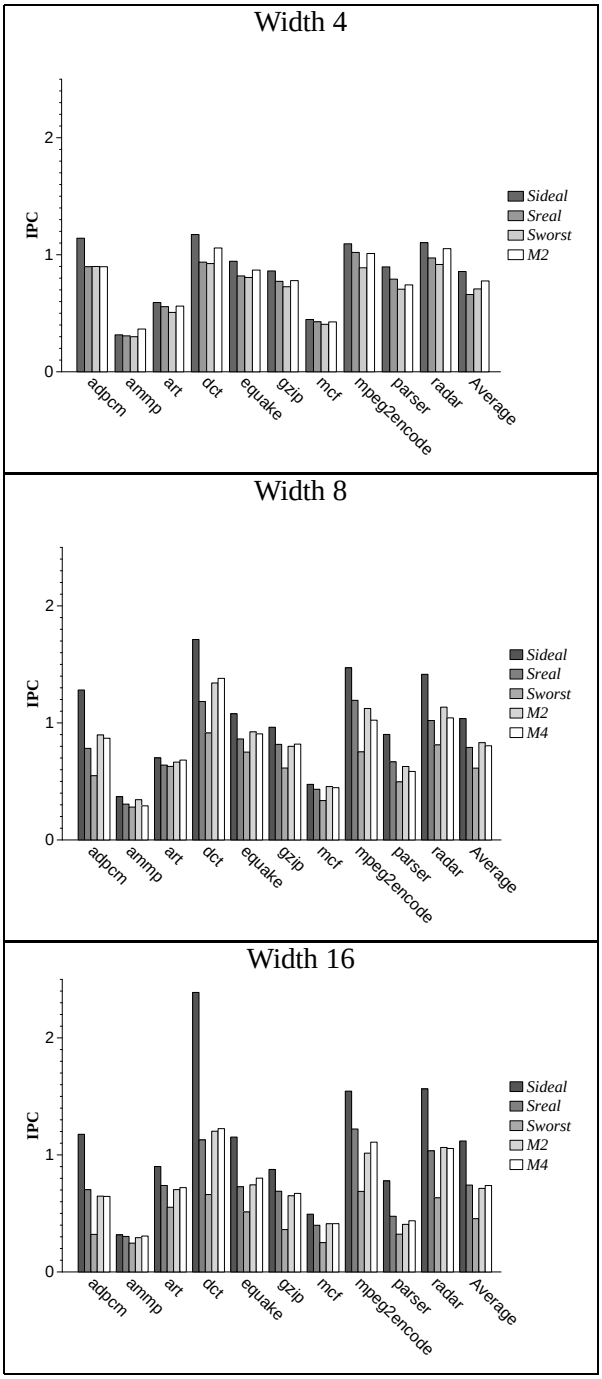


Figure 11: IPC on VLIW machine: Wiring overhead $\alpha = 2$ assumed for single-hop and multi-hop networks.

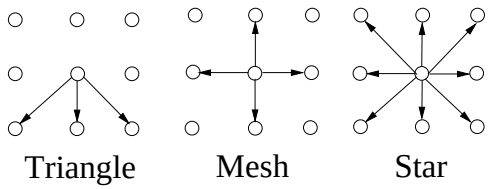


Figure 12: Grid Interconnects.

ALUs with short paths among them, mapping compiler-generated hyperblocks to this array. Multi-hop networks are ideally suited for this class of architectures where the primary goal is to avoid global communication and extract performance from ALU chaining by mapping the critical path on the shortest physical path. Previous work demonstrated the criticality of interconnect latency in GPAs [10]. This section contains a more detailed analysis of the effect 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 fanout/fanin forwarding delay of 0.25 cycles. Again, the *1 segment* delay was 0.60 cycles, and we simulate a 10F04 clock cycle.

5.1 Results and Discussion

We examine the same three parameters of performance as in the VLIW experiments. We examined a multitude of different connection topologies on an 8x8 grid, shown in Figure 12. The networks range from very low fanout, to moderately high fanout. The *triangle* network connects 3 neighbors together, a little similar to the VLIW *M2* interconnect, while the moderately rich *star* network with a fanout of 8, connects the immediate 8 neighbors together. For comparison we looked at the ideal, realistic and worst case single-hop networks similar to the ones in the VLIW machines, scaled to an 8x8 ALU array. We examined wiring overhead factors of 1 and 2 to determine the wiring area effect on the single-hop, high-bandwidth networks. We assumed all configurations to have wires connecting the bottom of the grid to the top. In the express channel configurations (denoted by the suffix E in the tables and graphs), this wire is laid out at a higher level of metal and is hence four times faster. For an 8x8 grid, the express channels have a total delay of 1 cycle.

Routing Latency: Examining Table 4, we can see that the average routing latency in the grid network for the ideal case (S_{ideal}) is the lowest among all the connectivities while the S_{worst} latency is the highest (although the *triangle* configurations are pretty close to worst case). The realistic single-hop network S_{real} with no wiring overhead comes closest to the ideal, followed by the *star* network, the *mesh* network, and the *triangle* network, in that order. It should be noted that the express channels make little difference to the average latency numbers for the multi-hop networks.

When we take into account the wiring overhead for the single-hop networks (α = 2), the average latencies for the S_{ideal}, S_{real}, and S_{worst} cases are now 0.75, 4.2, and 15.6 cycles respectively. The *star* network performs best and is about 30% better than the S_{real} with the wiring overhead incorporated.

Contention: The amount of contention is closely related to the fan-out of the topologies. The star network has the least contention, exhibiting 12% and 19% contention for the topologies with and without express channels respectively, followed by the mesh network (with 27% contention for both with and without express channels) and the triangle network (with 42.9% contention for both with and without express channels). This is to be

Interconnect	Latency (cycles)	Cont Delay %	No. of hops
S_{ideal}	0.25	0	1
S_{real}	2.07	0	1
S_{worst}	7.85	0	1
Mesh	5.11	27	3.2
MeshE	5.06	27	3.1
Star	3.26	12.8	2.5
StarE	3.32	19.8	2.5
Triangle	7.8	42.9	2.8
TriangleE	7.84	42.9	2.8
$S_{ideal}(\alpha = 2)$	0.74	0	1
$S_{real}(\alpha = 2)$	4.2	0	1
$S_{worst}(\alpha = 2)$	15.58	0	1

Table 4: Communication latencies, number of hops, and contention percentages for different interconnects in the Grid Processor. Area overhead factor equal to 1 and 2.

expected because, as the interconnect richness increases, the latency due to contention decreases, making the star network the most efficient here.

Number of Hops: From Table 4, we see that the number of hops for the single-hop networks is again one, since there is a dedicated path from every node to every other node. The number of hops for the multi-hop topologies varies by topology, and is lowest in the *star* network (2.5 hops) and highest in the *mesh* network (3.2 hops). The *triangle* network averaged 2.8 hops. All the multi-hop networks exhibited very close average number of hops for the with and without express channel cases. The fact that the *mesh* network has a higher average number of hops but a lower average latency is simply due to the the higher contention of the triangle network.

IPC: The S_{ideal}, S_{real}, and S_{worst} topologies averaged 7.9, 4.9, and 1.5 respectively when wiring overhead is ignored. The *mesh*, *star*, and *triangle* topologies averaged 3.1, 4.2, and 2.5 for the networks with express channels and 3.1, 4.0, and 2.5 for the networks without express channels. These IPCs for all the benchmarks are presented in Figures 13 and 14. They are split between the low and high IPC benchmarks for αs equal to 1 and 2. In the figures we present simulations results when wiring overhead of $\alpha = 2$ is incorporated for the multi-hop networks and the single-hop networks. As expected, the S_{ideal} topology performs the best while the S_{worst} topology performs the worst. Without wiring overhead, S_{real} is the next best topology followed by the *star* network. However, when wiring overhead is included, the *star* networks outperform the S_{real} network and the *mesh* and *triangle* are very close as well.

5.2 Summary

Table 5 shows the normalized average IPC achieved by each of the interconnects. All IPCs are normalized with respect to the S_{ideal} network. We see that the full broadcast single-hop network S_{real} achieves 63% of ideal performance (this network directly corresponds to a broadcast network in dynamically scheduled superscalar processors). The *star* with express channels which is a multi-hop routed network, with an order of magnitude less bandwidth than the broadcast network achieves 56% of ideal performance. When the wiring overhead is taken into account, the S_{real} network achieves only 46% of the ideal network, achieving 10% less than the

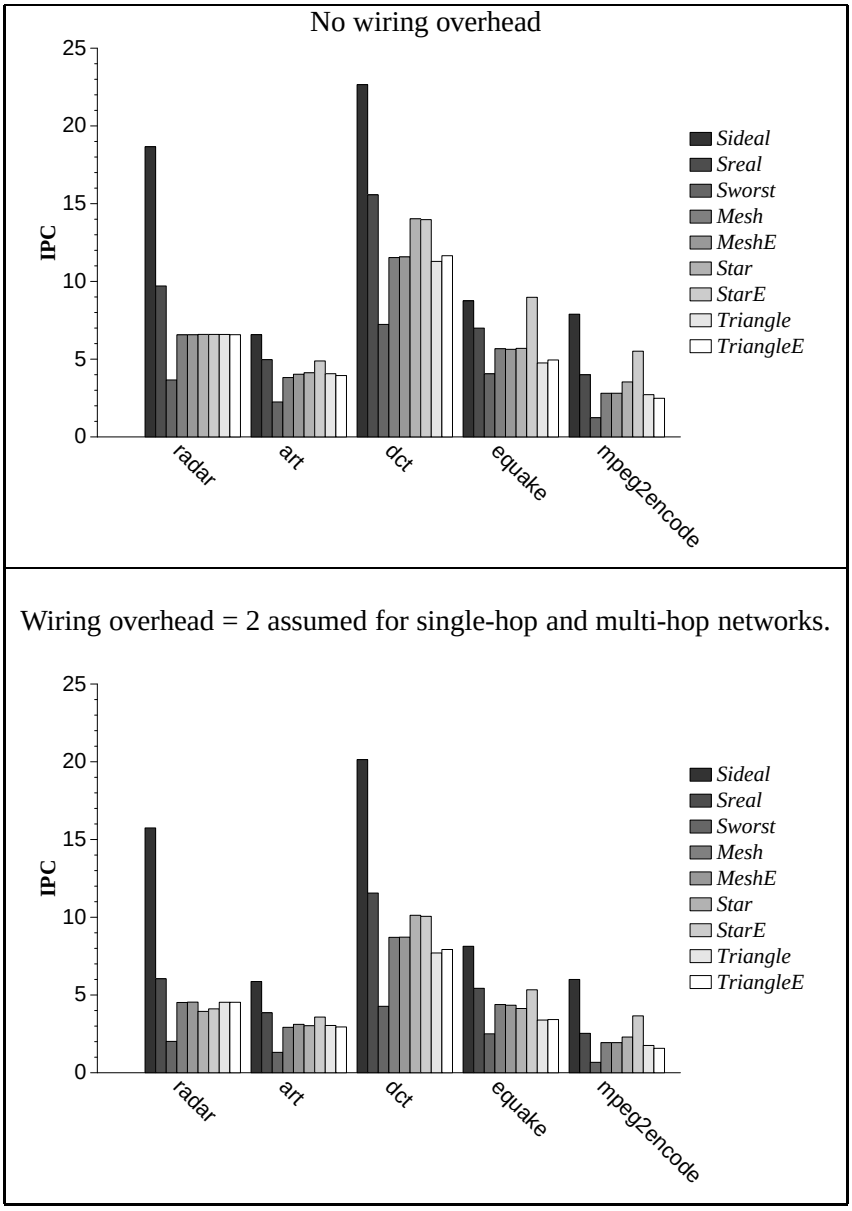


Figure 13: IPC on Grid Processors (8x8 grid). High IPC benchmarks.

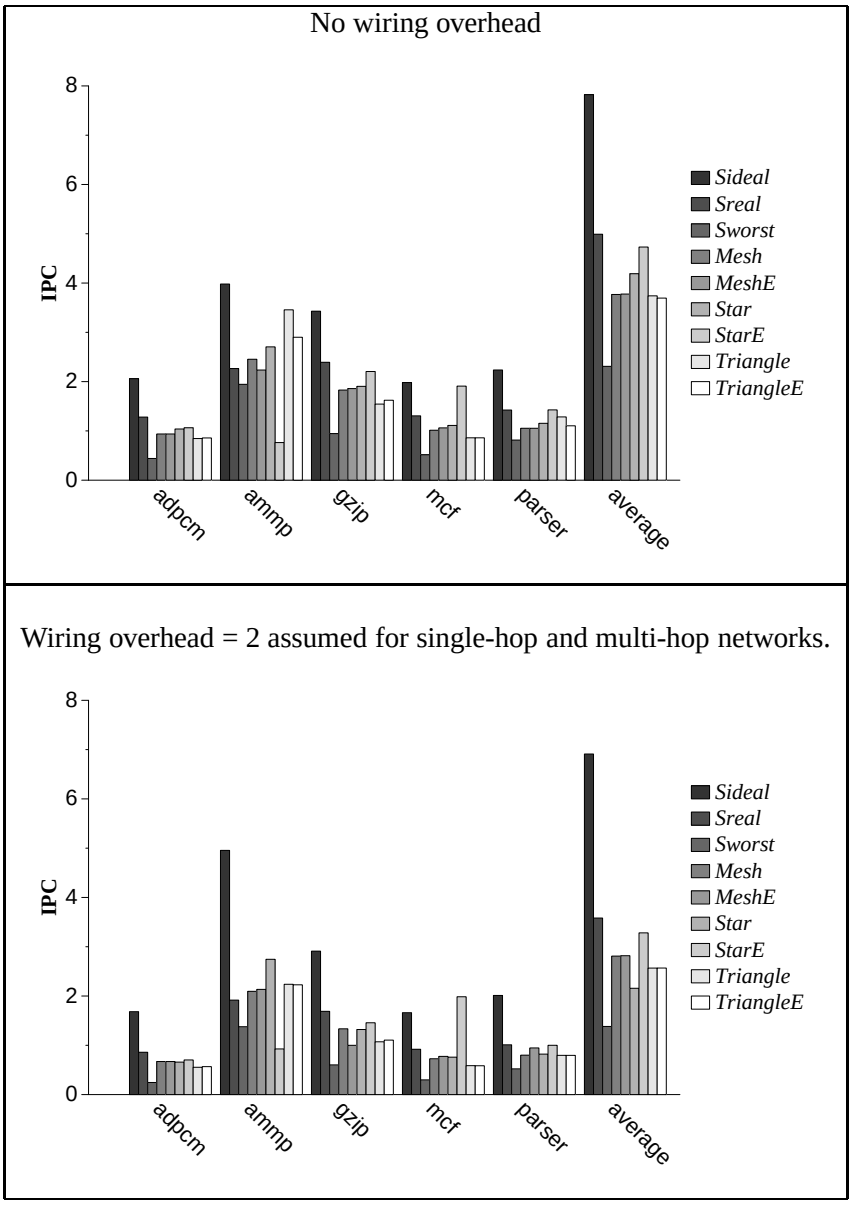


Figure 14: IPC on Grid Processors (8x8 grid). Low IPC benchmarks.

Configuration	Efficiency
S_{ideal}	1
S_{real}	0.63
S_{worst}	0.3
Mesh	0.41
MeshE	0.42
Star	0.52
StarE	0.56
Triangle	0.35
TriangleE	0.34
$S_{ideal}(\alpha = 2)$	0.89
$S_{real}(\alpha = 2)$	0.46
$S_{worst}(\alpha = 2)$	0.19

Table 5: Normalized average IPCs using different interconnects. All IPCs normalized to the ideal S_{ideal} network.

star network. The worst performing multi-hop network was the *triangle* without express channels. While this interconnect performed moderately well in the low IPC benchmarks, when the parallelism is high, the *triangle* interconnect becomes a bottleneck because the multi-hop network is constrained by its low bandwidth and cannot deliver all of the operands that are being produced in time.

These performance trends indicate that multi-hop inter-ALU networks with an optimized router design, are extremely effective, performing almost as well. as full broadcast networks. As the bandwidth and richness of the interconnect is reduced, the low bandwidth becomes a bottleneck and programs with lots of parallelism, are not efficiently executed.

6 Conclusion

Dramatic increases in on-chip real-estate has 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, also called a routed inter-ALU network (RIAN) as a communication architecture scalable to 10s 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 to implement broadcast networks. 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 less than 180ps in a 100nm technology. We applied these routing techniques to a conventional VLIW architecture and a dynamic grid architecture and showed that operand broadcast is not necessary and that existing scheduling algorithms are effective at placing producers and consumers close to one another in such a network.

As a result, our results show equivalent overall performance to a much richer and expensive broadcast network. If we were to impose the area (and therefore communication delay) penalty from the wires required to implement the broadcast network, the RIAN would significantly outperform the broadcast network. A key feature to the processor architectures which enabled our routing strategy is the knowledge of source and destination instruction locations and the optimization of them prior to instruction execution. While we used a static compile-time scheduler to place instructions for minimizing communication distance, similar analysis could be performed at runtime through trace generation or dynamic compilation techniques. For feasibility, however, future work would have to demonstrate that the time required to generate a good schedule does not become a bottleneck.

References

[1] Vikas Agarwal, Stephen W. Keckler, and Doug 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.S. 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*, November 1995.

[3] Mary D. Brown, Jared Stark, and Yale N. Patt. Select-free instruction scheduling logic. In *34th Annual International Symposium on Microarchitecture*, pages 204–213, December 2001.

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

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

[6] Marco Fillo, Stephen W. Keckler, William J. Dally, Nicholas P. Carter, Andrew Chang, Yevgeny Gurevich, and Whay S. Lee. The M-Machine Multicomputer. In *Proceedings of the 28th International Symposium on Microarchitecture*, pages 146–156, Ann Arbor, MI, December 1995.

[7] 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.

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

[9] Chunho Lee, Miodrag Potkonjak, and William H. Mangione-Smith. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *International Symposium on Microarchitecture*, pages 330–335, 1997.

[10] 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*, December 2001.

[11] 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.

[12] Joan-Manuel Parcerisa, Julio Sahuquillo, Antono González, and José Duato. Efficient interconnects for clustered microarchitectures. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, September 2002.

[13] Li-Shiuan Peh and William J. Dally. Flit–reservation flow control. In *Proceedings of the 6th International Symposium on High–Performance Computer Architecture, Toulouse, France*, pages 73–84, January 2000.

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

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

[16] Eric Sprangle and Doug Carmean. Increasing processor performance by implementing deeper pipelines. In *International Symposium on Microarchitecture*, May 2002.

[17] Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Paul Johnson, Walter Lee Jae-Wook Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, March 2002.

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

[19] 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.

[20] Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarsinghe, and Anant Agarwal. Baring it all to software: RAW machines. *IEEE Computer*, 30(9):86–93, September 1997.