SWIFT: A 2.1Tb/s 32×32 Self-Arbitrating Manycore Interconnect Fabric

Sudhir Satpathy, Ronald Dreslinski, Tai-Chuan Ou, Dennis Sylvester, Trevor Mudge, David Blaauw
University of Michigan, Ann Arbor

Abstract

A 32×32 64b self-arbitrating switch fabric called SWIFT achieves a bandwidth of 2.1Tb/s with single cycle arbitration and data transfer latency in 65nm technology while operating at 1026MHz at 1.2V. SWIFT co-optimizes arbiter and crossbar logic using a unique fabric architecture that integrates conflict resolution with data routing to optimally use logic and interconnect resources. It spans 0.35mm^2, achieves an efficiency of 7.39Tbps/W, and operates down to 530mV.

Motivation and Proposed Approach

High radix, high bandwidth, and low latency switch fabrics are key enablers for high-end servers, Ethernet routers, multiprocessor NoCs, multimedia accelerators, and manycore computers in general [1,2]. Conventional switch fabrics typically consist of a crossbar to route data and a separate arbiter to configure the crossbar. This poses two hurdles for scalability: 1) Routing to and from the arbiter incurs significant overhead as the number of sources and destinations grows. Also, the rapidly growing complexity of centralized arbitration with increasing crossbar radix can dominate overall delay. 2) Area utilization is poor since the arbiter is logic dominated, whereas the crossbar is routing intensive. The high cost of large radix switches limits fabric scalability [3,4,5] by requiring more stages in the data traversal path. This results in higher latency, reduced energy efficiency due to intermediate data storage, and complex routing protocols to handle inter-stage communication.

To mitigate this, we propose SWIFT(SWizzle Interconnect Fabric Topology), featuring a novel distributed arbitration scheme that reuses the data transfer bit-lines as "priority lines" for conflict resolution and locally stores the connectivity status at crosspoints. This eliminates additional routing and logic overhead to produce a compact design. A 32×32 router with 64b data buses (2048 wires) requires just 0.35mm^2 in 65nm, including single cycle arbitration. This corresponds to the area required to route its 2048 input/output wires at 2× min. spacing (and no additional tracks). SWIFT achieves a bandwidth of 2.1Tb/s at 1.2V with an efficiency of 7.39Tbps/W, and is fully functional down to 530mV with peak efficiency of 36.8Tbps/W.

SWIFT reuses concepts from SRAM design to make single cycle arbitration and data transfer latency possible at high radices. It uses an area efficient thyristor-based sense amplifier enabled latch (SAEL) for fast robust single-ended bit line evaluation. It supports four priorities for fairness during conflict resolution and the ability to multicast. Hence, SWIFT is 1.9× more energy efficient with 53% more bisection bandwidth at 80% lower latency over the best reported single stage fabrics to date, as shown in Table 1. SWIFT's 2.3× smaller area, 2.1× faster speed, and 69% higher energy efficiency (at iso-performance) over traditional switches (Fig. 1) can significantly improve NoC performance/latency when used in place of conventional 5×5 node routers.

In SWIFT the input (source IP) and output (destination IP) buses run perpendicularly (Fig. 1). This creates a matrix of crosspoints and a connection between an input and output bus is established by locally storing a logic 1 at their crosspoint. At most one logic 1 can exist along a column, whereas multiple 1s can be present along a row to multicast data. Every channel can independently operate in one of two modes: *Arbitration* or *data-transmission*. A pre-charge followed by conditional discharge scheme is used for high speed signaling on the channels, eliminating worst-case switching induced crosstalk.

In *arbitration* mode, each input channel is assigned a bit line from the output bus as its *priority line*. Upon requesting an output channel, a particular input channel suppresses lower priority channels competing for the same output bus by discharging their priority lines. It concurrently samples its own priority line with an SAEL which, if high, guarantees that no higher priority inputs requested the channel. In this case, a 1 is stored directly at the crosspoint, indicating a granting of the channel (Fig. 1). Conflict detection and resolution is thus performed in one cycle with the same structure used for data transfer, resulting in a very compact and fast implementation.

For *data-transmission*, bit lines comprising a channel are discharged at a crosspoint if the SAEL stores a logic 1 and input data is high. Incoming data is transition encoded to avoid repeated discharge when transmitting a sequence of logic 1s.

Fig. 2 shows the crosspoint circuit. The 64b input bus is multiplexed to send data or channel addresses. A set of channels is requested/released by raising *Req/Rel* high and loading a multi-hot vector in the input bus (with the position of 1s corresponding to the channel addresses that are requested). In a directory-based shared memory system, this multi-hot vector corresponds to caches having a shared copy of the data and is directly obtained from the home node directory, without requiring any further decoding. The *channel free* signal along each channel indicates its availability for arbitration and toggles its status after a successful request/release operation. Use of *Rel* follows the last data transfer and requires one cycle. However, for bursts with fixed packet lengths, *Col_Rel* and *Row_Rel* allow the release operations to overlap with the last cycle of data transfer. *Row_Rel* frees up all channels held by a source while *Col_Rel* frees up a channel connected to a destination.

As shown in Fig. 3, a SWIFT-enabled 32-core system improves response latency by 1.84× over a mesh-based system for typical workload traffic owing to superior multi-cast and conflict resolution ability. Performance numbers for this comparison were obtained by placing and routing 32 cores and 32 L2-caches with one of the three studied fabrics (bus, mesh, and SWIFT). The interconnect delay was then extracted and the routing overhead studied. For the SWIFT based many core system, the availability of ports at either side of the fabric as datapath-compatible buses (instead of bit interleaved buses) facilitates routing and resulted in the use of less than 6% of total available tracks for connecting cores and caches.

Fig. 4 shows the SAEL and channel monitor circuits. The SAEL uses a thyristor-based sense amp for better robustness, 5× smaller area, and simplified timing compared to a conventional sense amp. In previous work [6], the pre-charge devices in a thyristor were upsized for leakage compensation, degrading performance for functionality at low V_{dd}. In contrast, SWIFT uses a bypass path with a skewed inverter for low V_{dd} operation to allow smaller pre-charge devices, improving speed at higher V_{dd}. The area penalty is small, since SAEL accounts for only 3% of the 254μm^2 crosspoint. The ability to select between the two sensing paths also provides a knob to compensate for process variation at lower V_{dd}. Only falling transitions are critical, as seen in the timing diagram of Fig. 3, allowing device skewing to reduce delay. For fairness in channel allocation, four sets of priorities are locally generated at each crosspoint with one selected every cycle. Instead of storing four 32b priority vectors for each of the 1024 crosspoints, eight logic gates, uniquely selected for each crosspoint, indicate which priority lines are discharged for each priority. This approach incurs no area penalty over a single priority implementation.

Prototype Implementation

A test prototype was fabricated in 65nm CMOS with a 32×32 64b bus SWIFT that emulates a 32-core/32-L2 cache system using traffic generators and signature analyzers to replace the cores and caches. The traffic generators can be digitally tuned to produce traffic and collision patterns mimicking a real system under different workloads. Fig. 5 shows measured bisection bandwidths of up to 2.1Tb/s at 1.2V and 0.16Tb/s at 600mV at room temperature. With all channels active and 20% switching activity on primary inputs, SWIFT consumes 219mW at 958MHz at 1.1V (112fJ/bit). Fig. 6 shows measured SWIFT power consumption (with breakdown) under different switching activities with varying numbers of active channels. At low switching activity, power is dominated by control signals (66% at 1.1V), whereas output lines are more significant (40% at 1.1V) at high activity. Fig. 7 shows bandwidth degradation (44.6% max) and power overhead (9.7% max) with varying collision rates. As shown in Fig. 3, for typical multiprocessor workloads collision rates are less than 10% resulting in <1.5% SWIFT bandwidth degradation and <0.5% power overhead. Die micrograph and SWIFT layout are shown in Fig. 8.

References

[1] S.Bell *et al*, ISSC 2008, pp. 88-89
[2] S.Tremblay *et al*, ISSCC 2008, pp. 82-83
[3] P.Salihundam *et al,* SoVC, 2010, pp. 79-80
[4] M.Anders *et al,* ISSCC, 2010, pp. 110-111
[5] S.Vangal *et al,* SoVC, 2007, pp. 42-43
[6] S.Satpathy *et al,* SoVC, 2010, pp. 81-82

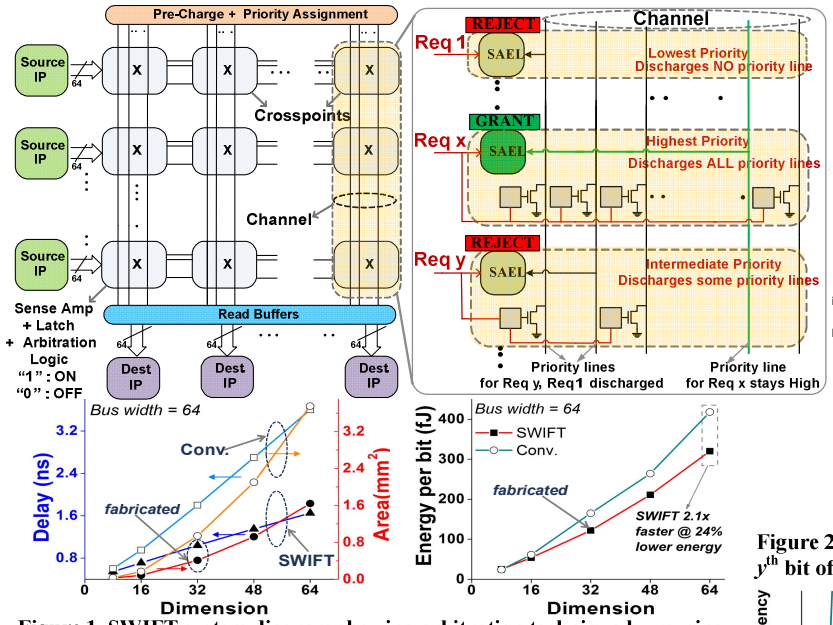

Figure 1. SWIFT system diagram showing arbitration technique by reusing bitlines. SWIFT is 2.3× smaller, 2.1× faster, and 24% lower energy

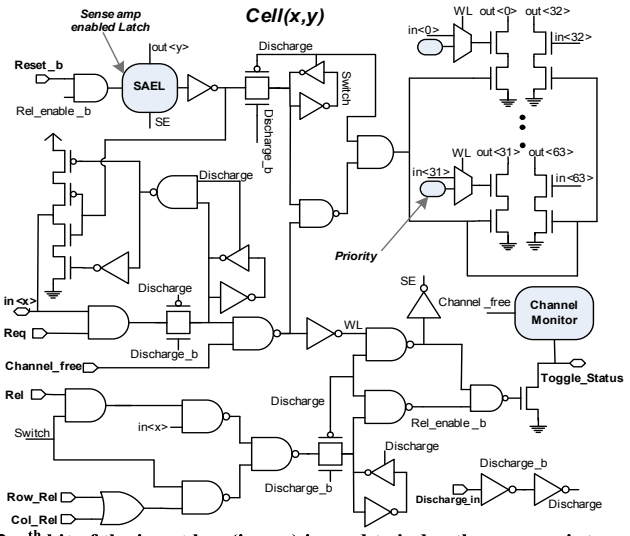

Figure 2. x^{th} bit of the input bus (in<x>) is used to index the cross-point, y^{th} bit of the output bus (out<y>) is sampled by SAEL during arbitration

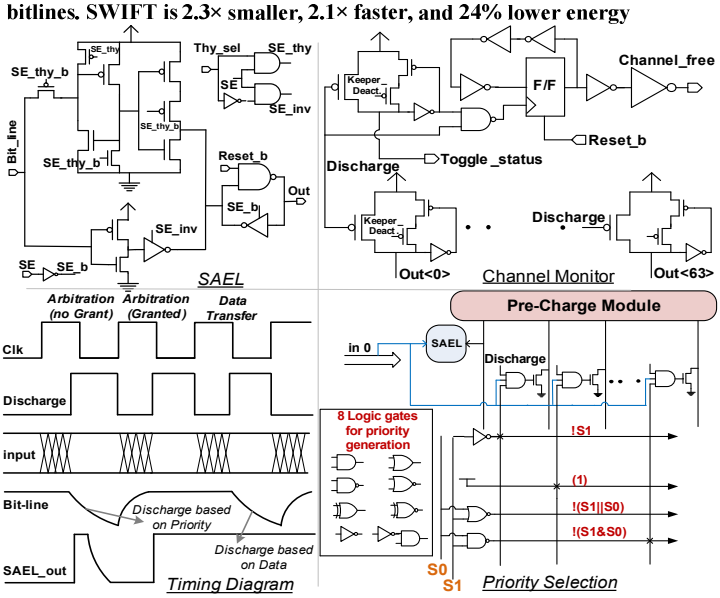

Figure 4. Top, SAEL using thyristor based sense amp. Channel monitor updates *channel_free* when *toggle_status* is pulled low. Bot, Timing diagram showing that falling edge transitions are critical. 4 priorities are available at each crosspoint, one of which is selected for arbitration.

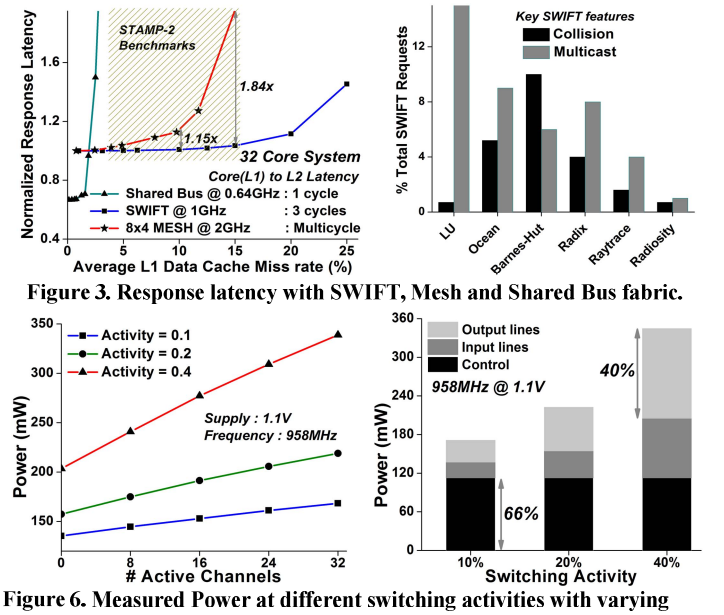

Figure 3. Response latency with SWIFT, Mesh and Shared Bus fabric.

Figure 6. Measured Power at different switching activities with varying number of active channels and Power breakdown within SWIFT

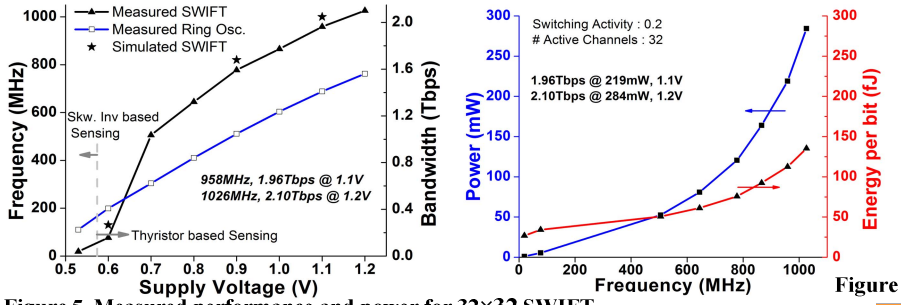

Figure 5. Measured performance and power for 32×32 SWIFT

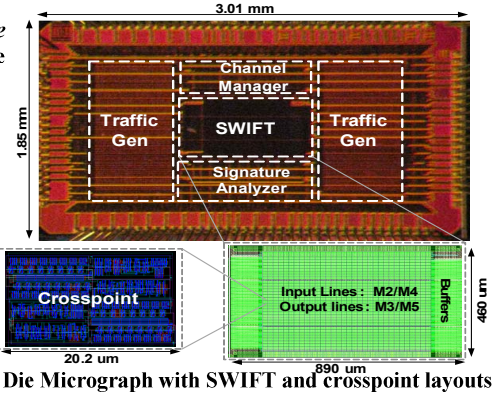

Figure 8. Die Micrograph with SWIFT and crosspoint layouts

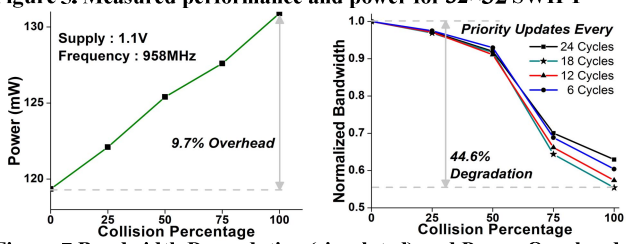

Figure 7.Bandwidth Degradation (simulated) and Power Overhead (measured) with increasing number of collisions for a channel.

Table1. Comparison of SWIFT with prior works. SWIFT is 1.9× more energy efficient with 53% higher bisection bandwidth at 80% lower latency.

Spec	VLSI 2010[3]	VLSI 2010[6]	SWIFT		
Tech.	45nm	65nm	65nm	65nm	65nm
Supply(V)/Temp.(°C)	1.1/50	1.1/27	1.2/27	1.1/27	0.6/27
Size	5×5(16B)	128×128(2B)	32×32(8B)		
Arbitration	5 to 1	None	32 to 1		
Area(mm²)	0.39*	0.75	0.35*		
Max. BW (Tb/s)	1.28	1.07	2.10	1.96	0.16
Frequency (MHz)	2000	530	1026	958	77
Power (mW)	467.5**	227	284	219	5.4
Efficiency(Tb/s/W)	2.74	4.71	7.39	8.87	29.1

*only Crossbar + Arbiter area accounted **Payload buffer power excluded