ASYNCHRONOUS RESEARCH CENTER
Portland State University

Subject: **The Weaver, an 8x8 Crossbar Experiment**
Date: **November 17, 2015**
From: **Ivan Sutherland, Marly Roncken, Navaneeth Jamadagni, Chris Cowan, and Swetha Mettala Gilla**
ARC#: **2015-is07v11**

ABSTRACT

This paper reports measurements from the Weaver, a self-timed 8x8 crossbar experiment built in 40nm CMOS by TSMC. The Weaver's crossbar steers individual data items from any of eight input channels to any of eight output channels. With a nominal 1.0 volt power supply, each channel passes about 6 Giga Data Items per Second (GDI/s). Data items are 72 bits wide giving the crossbar's eight channels a combined throughput of nearly 3.5 terabits per second. Local arbitration throughout the crossbar reduces internal contention so that only input and output channel capacity limit throughput. Each data item passes through the crossbar in less than one nanosecond. The crossbar on the Weaver chip occupies about a tenth of a square millimeter in a triangle 433 microns by 391 microns in size. At full capacity the crossbar itself consumes less than half a Watt; in the absence of traffic, only leakage consumes power.

A triangular array of 56 switches forms the crossbar, one for each possible connection between channels. The Weaver chip places the switches in close proximity and provides recirculating FIFO rings to connect the crossbar outputs back to its inputs for extended high speed testing. A network-on-chip (NoC) application would distribute identical switches geographically to form the on- and off-ramps of a freeway-like data network. A self-timed metastable-aware mutual exclusion element at each merge safely blends arbitrary streams of data elements on a first-come-first-served basis.

INTRODUCTION

Instead of a global clock, a self-timed network paces the flow of data items with local handshakes. Timing signals flowing forward with data items provide source clocking. Timing signals flowing backwards from output to input provide flow control. Together these local timing signals advance data items and limit congestion. Clock gating is automatic because the local timing signals act exactly once only when

necessary to copy a data item. By avoiding a global clock, self-timing allows networks to extend to all parts of even large chips.

The flow of data items in the Weaver is much like the flow of cars in a freeway system. Each data item advances as soon as space for it is available. In the absence of congestion, data items advance at high speed through the network, just as cars travel at high speed in light traffic. In the presence of downstream congestion, the reverse timing signals retard forward flow to prevent collisions. When traffic is heavy, spaces appear to move backwards, just as they do on a congested freeway.

Two of the Weaver's switching circuits serve essential roles for self-timed networks. The *data-controlled branch* serves the role of a freeway exit ramp. A selected bit within the data item itself causes the *data-controlled branch* to steer that data item out of the main flow. The *demand merge* serves the role of a freeway entrance ramp. An arbitration circuit within the *demand merge* fits the new data item safely into the stream of network traffic. The *demand merge* captures data from either of two inputs into 72 two-input latches on a first-come-first-served basis. Together the *data-controlled branch* and the *demand merge* form interchanges between data networks. The Weaver chip tests an 8x8 crossbar of such switches.

This paper reports measurements from the Weaver which was built in 40 nm CMOS technology by TSMC. With a nominal 1.0 volt power supply we observe a throughput per channel of about 6 Giga Data Items per Second (GDI/s). Each data item carries 72 bits, providing a data rate at nominal voltage of about 480 Gb/s per channel, or more than 3.5 tera bits per second (Tb/s) for the full crossbar.

Reducing the power supply voltage reduces both throughput and energy per bit moved. At 70% of nominal power supply voltage, the Weaver runs at a little less than half speed. At that reduced voltage the total throughput is about 1.5 Tb/s for the full crossbar. Throughput tracks changes in power supply voltage automatically without separately adjusting a clock frequency. Throughput scales very closely with (Vdd-0.55). The crossbar is entirely free of the tyranny of an external clock and strikingly robust against power supply variation.

At 70% of nominal power supply voltage, the Weaver consumes about half as much energy to move each bit. This is not surprising because the only loads are capacitive. Energy consumption per data item scales very closely with the square of the supply voltage. Because throughput also scales with voltage, power consumption scales more rapidly with voltage. The Weaver chip has demonstrated error-free operation from 0.6 volts to 1.0 volt; we have yet to explore operation outside that range.

THE WEAVER CHIP

Figure 1 is a schematic of the Weaver test chip. Eight self-timed channels of 48 stages each recirculate data from the output of the 8x8 switch back to its input.

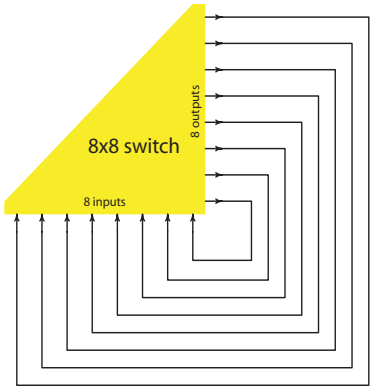

Figure 1 Weaver schematic. 8 rings recirculate data through the switch.

The floor plan of Figure 2 shows these eight recirculating channels flanked by two more that bypass the switch. These ten channels of 48 stages each[1] form a ribbon cable that folds diagonally at the four diagonal edges of the floor plan. The ten closed rings are all approximately the same physical length. The ribbon's middle eight channels (numbers 1:8) pass through the

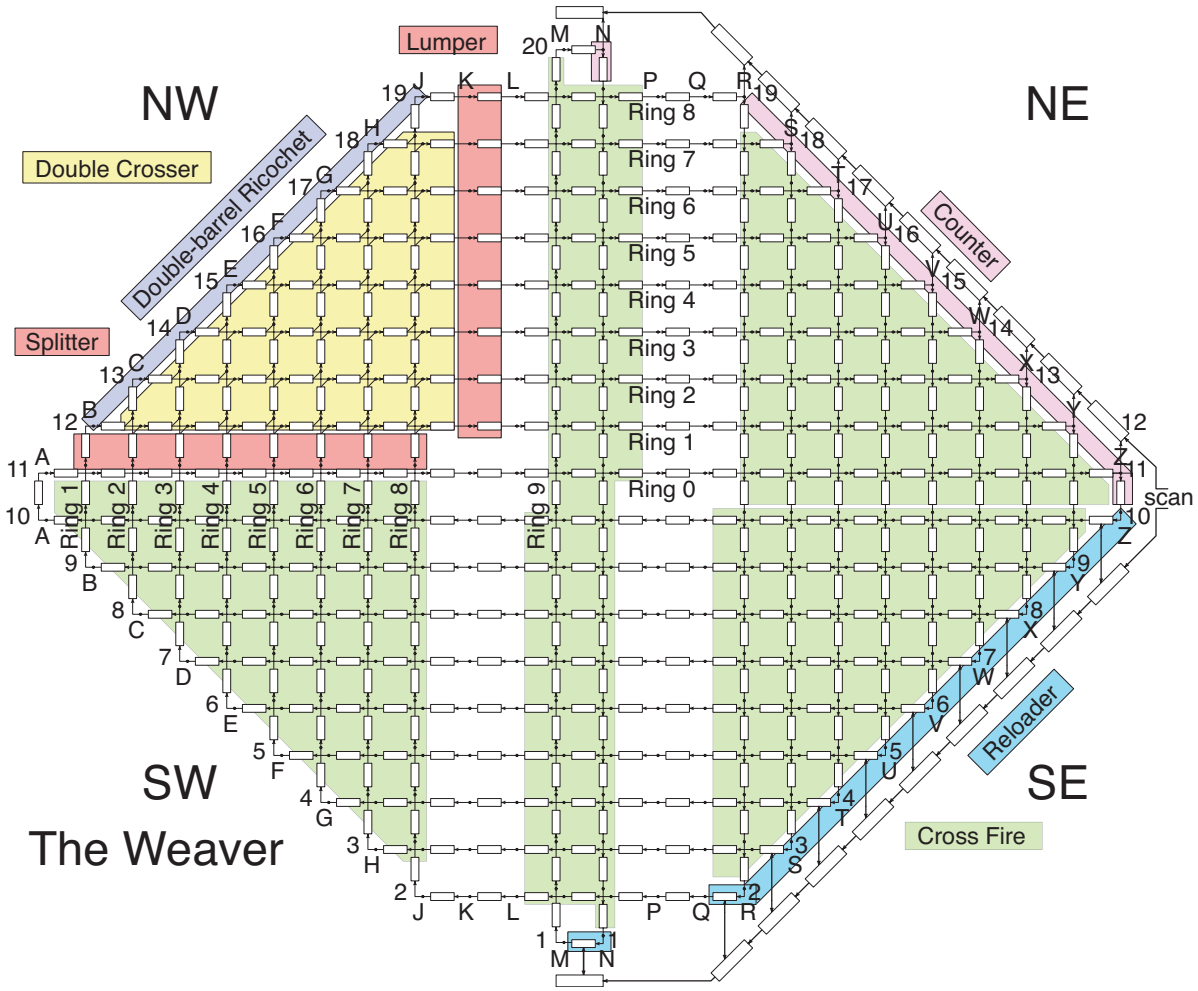

Figure 2: The Weaver floor plan; The Weaver includes ten rings, of which eight recirculate data from the output of the crossbar (yellow) back to its input. Each ring has a reloader stage to load and read data values and a counter to tally data passage.

[1] Channel 9 has only 40 stages. It runs N-S in the middle of the chip.

triangular 8x8 crossbar switch in the North West. The two outer channels of the ribbon (numbers 0 and 9 which are omitted from Figure 1) bypass the crossbar for speed comparison. Each channel includes a *counter stage* to measure throughput and a *reloader stage* to insert and extract data values. The *Weaver* is so named because data items can weave complex paths through its eight middle channels. Many of the part names of the Weaver, such as *ricochet* and *double crosser*, adopt gunslinger lingo from Western movies and appear in italic type in this paper.

Initially we sought a logarithmic crossbar structure to save switches. We failed to find a suitable logarithmic switch structure. Logarithmic structures have irregular geometry and so require repeaters to move data to their switches. A repeater is about the same size as a switch because both switch and repeater have 72 latches and the switch requires only a few more control transistors than the repeater. Instead of a logarithmic structure with N log(N) switches and many repeaters, the Weaver's crossbar has a triangular structure of N*(N-1) switches and few repeaters. The triangular structure minimizes the crossbar's wire length rather than its transistor count. The folded ribbon cable form simplifies layout of the rings that recirculate data at high speed through the crossbar.

Figure 3 illustrates the structure of the Weaver's crossbar. Data items entering from the South on any of eight input channels exit to the East on any of eight output channels. Because the data path folds diagonally like a ribbon cable, each of the eight channels crosses every other channel in the crossbar exactly once. The colored arrows suggest how the switches at those 28 crossings (Figure 3 shows only six) allow data items to change channels. The repeaters that fold the data path near the crossbar are *double barrel ricochet* modules described later in this paper.

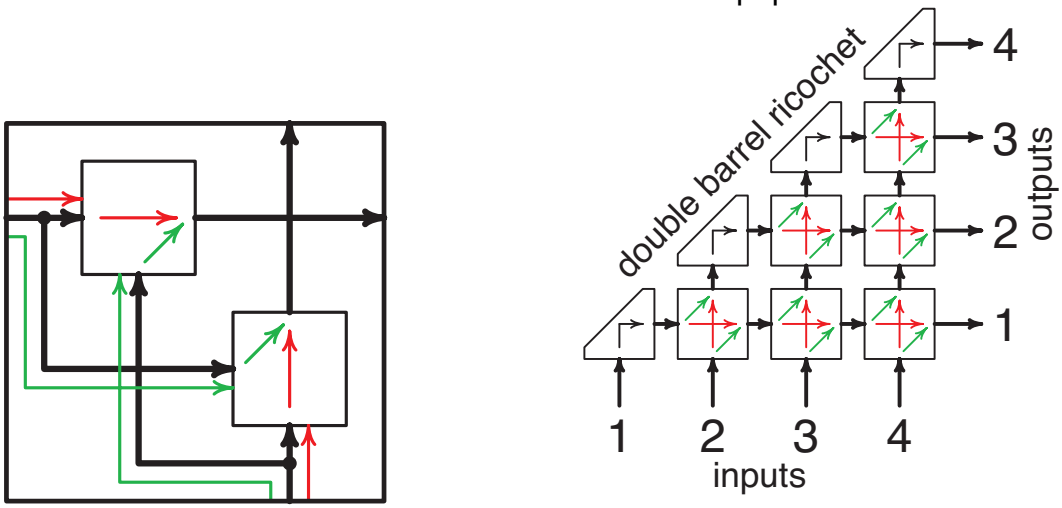

*Figure 3: the Crossbar. Left – each double crosser module has a separate Demand Merge element for each output. Right – the 8x8 crossbar is a triangular structure of 28 = (8*7)/2 double crossers of which only a few appear here.*

A complete 8x8 crossbar must have 8*7 = 56 individual switches. The Weaver arranges these in pairs, called *double crossers*, one pair at each of the 28 crossings.

The left panel of Figure 3 illustrates one double crosser. One of its *demand merge* elements serves its North output; the other demand merge element serves its East output. Each double crosser accepts data items from the South or West and delivers them to the East or North. Conflict in the crossbar happens only if two data items try to leave a double crosser concurrently by the same exit. The demand merge modules have arbitration and metastability protection to resolve exit conflicts on a first-come-first-served basis. Although metastability may very occasionally delay the passage of a single data item, the self-timed nature of the Weaver renders such delay harmless. Metastability delays are so rare and so short that we expect them to be undetectable.

STEERING BITS

Steering bits in each data item control how the data item passes through the Weaver. Because each data item carries its own steering bits, each data item weaves its own individual path through the crossbar. To simplify decoding, the Weaver test chip assigns an individual steering bit to each of the 28 *double crossers* in the crossbar; 28 of the 72 data bits in each data item may be steering bits; the remaining 44 data bits are entirely free of assigned meaning in any data item.

Each steering bit applies to a particular *double crosser*. Regardless of how it enters a double crosser, a data item with a *zero* in the steering bit position for that double crosser goes *straight* through it either from West to East or from South to North, staying in the same channel. Regardless of how it enters a double crosser, a data item with a *one* in the steering bit position for that double crosser passes *crooked* through it either from West to North or from South to East, changing to the other channel. At each double crosser each data item either remains in its channel or changes to the other channel. Each channel requires seven steering bits, one for each of the other channels to which it might change. The remaining 72 – 7 = 65 bits can carry arbitrary data, but of course some of those bit positions are used as steering bits in other data items.

This choice of rules for steering simplifies testing by forcing every data item to follow a closed path. Data items with zero in all steering bits circulate only in their initial ring. Data items with a one in the steering bit position for an intersection of two rings circulate alternately around those intersecting rings. Data items with multiple ones in steering bit positions weave through several rings in succession. A different application might use other steering rules such as decoding an exit address.

SCAN CHAIN

Control of Weaver experiments is entirely through a JTAG low-speed scan chain. The scan chain serves three purposes. First, it can read or clear the values in each of the ten 54-bit throughput counters, one for each ring. These counters appear at the North East edge of Figure 2, and Figure 4 offers more detail. Second, the scan chain can load a data value into or read a data value from a data item in a *reload* stage. The South East edge of Figure 2 holds the reload stages. Third, the scan chain can stop the

flow of data, sense the FULL or EMPTY state of every communication link, initialize the FULL or EMPTY state of every link, and restart self-timed flow. The Weaver implements Naturalized Communication and Testing as described in [ronk 2015].

In addition to the low-speed JTAG scan chain, two dedicated medium-speed output pins deliver reduced frequency real-time square wave signals from the counters. These outputs switch at 2^{-20} or approximately one millionth of the throughput rate of a ring. Two strings of multiplexors like those of Figure 4 allow the scan chain to select two rings for frequency output. These outputs permit real-time observation and comparison of throughput.

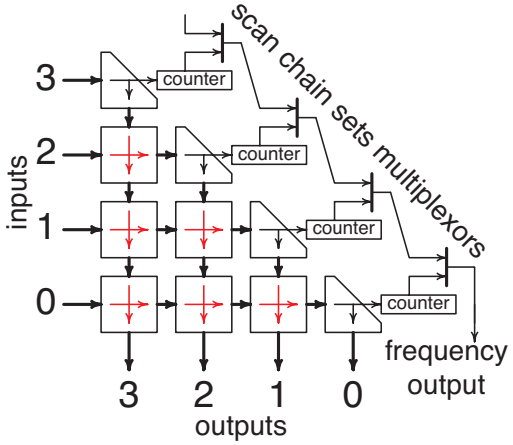

Figure 4: Each ring has a counter (only four appear here). The 19th bit of each counter provides a frequency output. The scan chain selects which output is delivered off chip.

CIRCUITS

Weaver uses the 6/4 GasP circuit family [Ivan 2001]. Figure 5 shows the basic 6/4 GasP control circuit; a *state wire* (SW) accompanies a bundle of data wires in each link to indicate the presence or absence of data in the link. The Weaver uses the *HI means FULL* state wire convention: the data wires in a link carry meaning only when the state wire in that link is HI; if the state wire is LO, the link is EMPTY. The ability to represent an absence of data is central to self-timed systems.

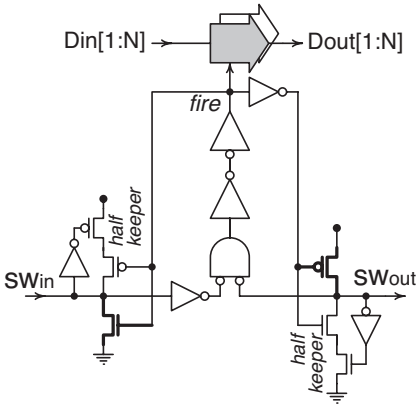

Figure 5: The GasP control circuit family uses one wire to represent FULL or EMPTY.

When its predecessor link is FULL and its successor link is EMPTY, the AND function in the 6/4 GasP circuit produces a brief ***fire*** pulse. The fire pulse does three things. First, it makes the data latches (Figure 6) momentarily transparent, copying data from the predecessor link to the successor link. Second, the fire pulse turns on the N-type transistor, drawn bold in Figure 5, to *drain* the predecessor link by driving its state wire LO, meaning EMPTY. Third, through an inverter, the fire pulse turns on the P-type transistor, shown bold in Figure 5, to *fill* the successor link by driving the successor state wire HI, meaning FULL. The two series transistors opposing the bold ones are keepers that retain the state wire's

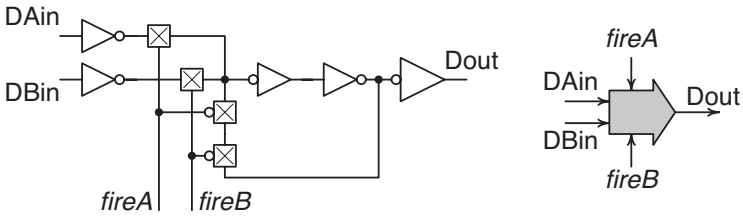

Figure 6: A two-input latch for the Weaver and its icon. The single-input latch omits two of the pass gates.

voltage between fire pulses and "get out of the way" during each fire pulse. By changing the state of the state wires, each fire pulse kills itself after five gate delays. Fire pulses fill the role of any clock signal that might otherwise have been needed. Because fire pulses happen only when needed, "clock gating" is automatic. Fire pulses in adjacent stages must alternate, because one can happen only when the link between them is FULL and the other can happen only when the link is EMPTY. Because of this alternation, latches rather than flip-flops can safely hold data values.

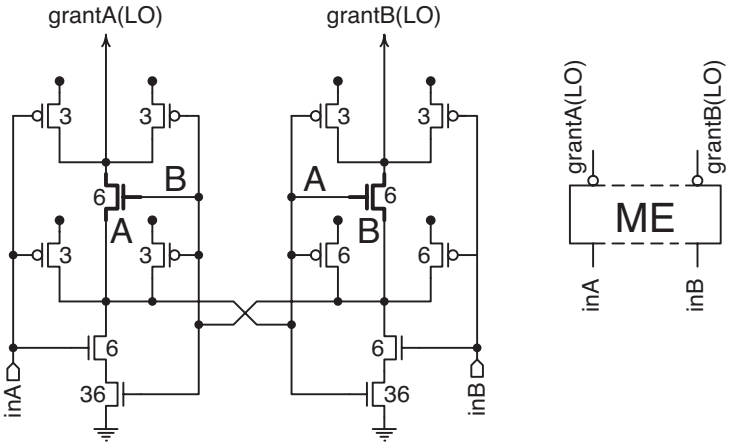

Figure 7: Seitz Mutual Exclusion element [Seitz 1980] and its icon.

The 6/4 GasP circuit of Figure 5 avoids state wire crossover current because of a subtle difference in its *turn on* and *turn off* delays. The action of two parallel transistors detects the condition that ends the fire pulse and turns OFF the state wire drive transistors. The action of two series transistors detects the AND condition required to start the fire pulse and turn ON the state wire drive transistors. Because the action of parallel transistors is faster than that of series transistors each stage turns OFF and stops driving its state wires before adjacent stages can turn ON and drive those same state wires. The difference in delay avoids concurrent conduction and crossover current. The difference in delay is enhanced by threshold shifts in the amplifiers that follow the AND gate.

Demand merge circuits handle contention in the crossbar switch. The heart of each demand merge is the mutual exclusion circuit of Figure 7 patterned after [Seitz 1980] in chapter 7 of the 1980 Mead-Conway VLSI book. This mutual exclusion circuit waits for metastability to end before delivering any output. By allowing only one side at a time of a dual latch driver to fire, as shown in Figure 8, the demand merge puts contending data items in sequence, on a first-come-first served basis. We chose the transistor sizes in the demand merge to minimize the delay of the uncontested case. Contention and metastability in the demand merge are so rare that designing instead for rapid exit from metastability is foolish.

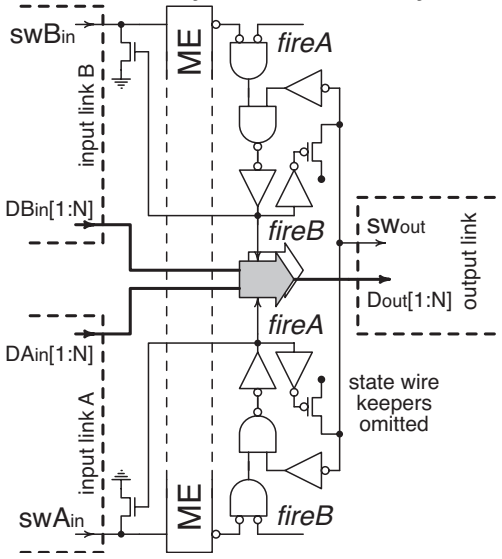

Figure 8: The Demand Merge circuit takes input on a first-come-first-served basis. ME is a mutual exclusion circuit.

DOUBLE BARREL LINKS

Double barrel links are novel and key to the Weaver design.[2] The twin state wires in a double barrel link are mutually exclusive; only one at a time will ever be HI. Either state wire being HI indicates that the link is FULL, but which state wire is HI carries one bit of information in a form useful for steering. A *splitter* stage converts a steering bit from bundled data form to double barrel form. Figure 9 shows a 6/4 GasP *splitter* with N bundled data input bits labeled D_{in}[1:N], N latches to capture them, and N bundled data output bits labeled D_{out}[1:N]. One of the N bundled data input bits, called D_{in}[s], also selects one of two series P-type state wire drivers to drive HI either the "straight" state wire, swS_{out}, or the "crooked" state wire, swC_{out}. The circuit copies all N input bits, D_{in}[1:N], as output bits, D_{out}[1:N], including D_{in}[s], the bit that conditioned the double barrel state wires. Thus D_{in}[s] remains available in bundled data form to repeat its selection task on a subsequent pass.

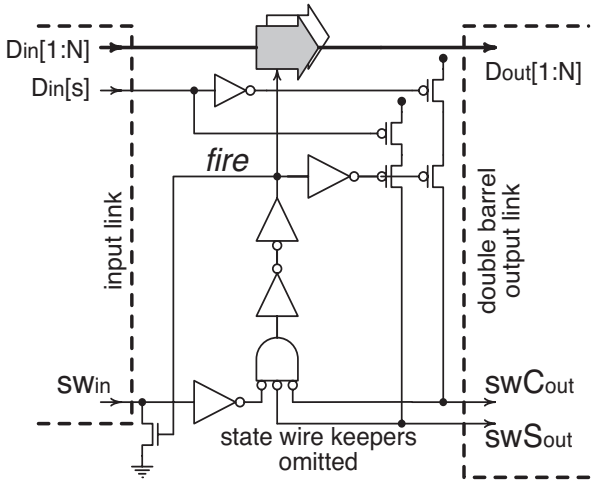

Figure 9: The splitter converts Din[s] into double barrel form.

Figure 3 (left) shows the two *demand merge* elements of the *double crosser* located at each intersection of Northbound and Eastbound data paths. The double crosser's two inputs are double barrel links with twin state wires. The dark black line in Figure 3 (left) represents the 72-bit data bundle, and the two narrow colored lines, one on each side of it, represent the twin state wires of each double barrel input link. The dark data lines divide to offer each input data value to both demand merge modules. In contrast, the double barrel state wires go separately, one to each demand merge module. Because the double barrel state wires are mutually exclusive, the arriving data item informs only one of the two demand merge modules of its arrival. Although the other demand merge is offered the data value, it remains ignorant of the validity of that value. The incoming double barrel state wires select which of the demand merge modules can send this data item to its exit. The demand merge modules merely resolve the contention that in a rare case might occur if two data items happen to announce their arrival at the same time to the same demand merge module.

The crossbar works by decoding each bundled data steering bit one stage in advance of need. Although omitted from Figure 3, each *double crosser* not only responds to double-barrel input links, but also produces double-barrel outputs. The Weaver combines the *splitter* circuit of Figure 9 that creates double barrel outputs with the *demand merge* circuit of Figure 8 at each switch position. Each double crosser decodes one steering bit from the input it will forward. Moreover, it decodes the

[2] Chris Cowan, a PSU graduate student, is responsible for this key idea.

particular bit from that input appropriate to the destination of the chosen input. This **advance decoding** enables each double barrel link to announce the exit direction assigned to each data item as the data item and its twin state wires enter a *double crosser*. Advance decoding is key to the Weaver's high throughput. We took great care to decode the proper steering bit exactly one stage in advance of need.

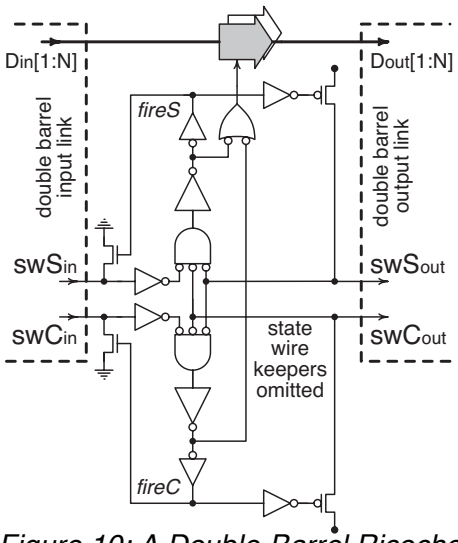

Figure 10: A Double-Barrel Ricochet has straight (swS) and crooked (swC) input and output state wires.

Double barrel links appear only inside the 8x8 crossbar and at its inputs and outputs. Just South of the crossbar, a red area in Figure 2 holds eight *splitter* modules like Figure 9 to create the double barrel links for the first row of *double crossers*. The *double barrel ricochet* modules at the North West boundary of the crossbar repeat and fold double barrel links. Figure 10 shows their 6/4 GasP circuit. They must pass forward the information carried in the twin state wires. Each *double barrel ricochet* acts only when both of its output state wires are LO, meaning EMPTY; if either output state wire were HI, the output link would be FULL.

TEST RESULTS – CANOPY GRAPHS

Figure 11 shows canopy graphs for the ten rings at nominal supply voltage. Rings 0 and 9 flank the other eight but omit switching elements. A canopy graph plots total throughput, as counted in the counter stage, as a function of the number of data items in the ring. An empty ring must have zero throughput just as an empty freeway carries no traffic. Likewise, a completely full ring has no throughput just as a congested freeway stalls traffic. At its left and right ends the canopy graph shows no throughput.

The linear rise in throughput with occupancy at the left of the canopy graph is easy to understand. One data item circulates with a period set by the forward latency around the ring, and so do any small number of data items, just like a few racecars on a circular track. Because data items cannot overtake each other, throughput increases along with the number of moving data items. The right side of the canopy graph shows the impact of congestion. As congestion decreases, more space becomes available for data items to move and there is a corresponding linear increase in throughput.

Somewhere between completely full and completely empty there is an occupancy of maximum throughput. The canopy graph for ring 9 shows its maximum of 6.4 GDI/s at 60% occupancy, namely 24 data items in its 40 stages. The 6/4 GasP circuits in the Weaver move bubbles faster than data items; the forward latency of each 6/4 GasP stage is about 100 psec and the reverse latency is only about 66 psec. The

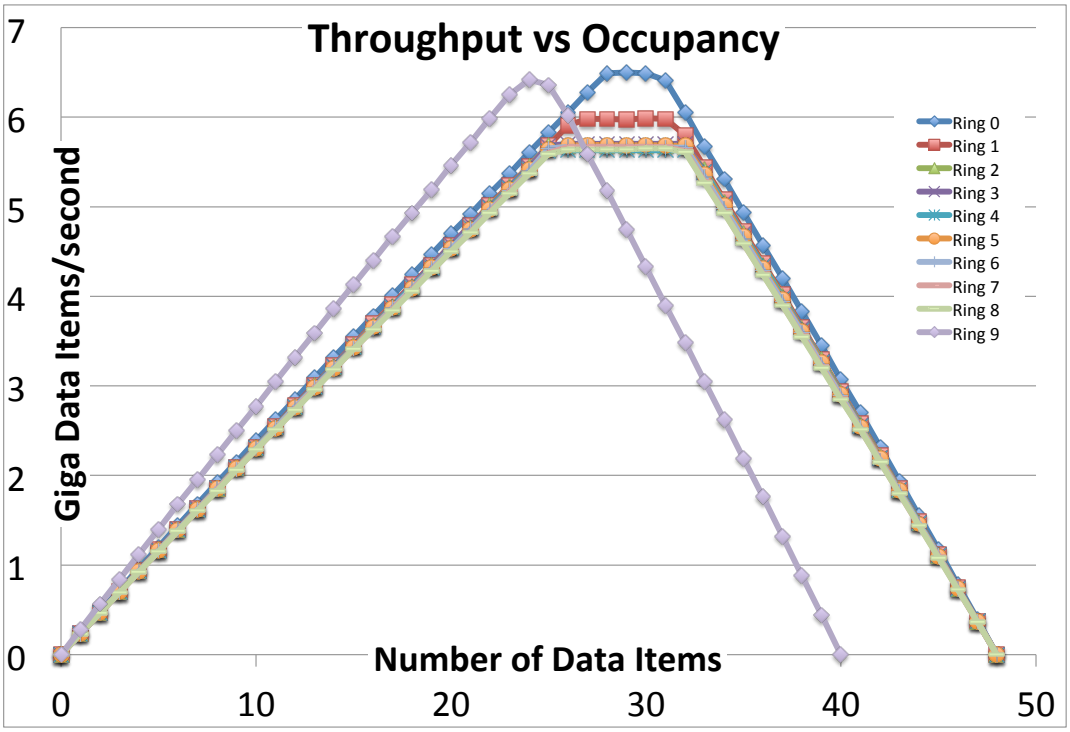

Figure 11: Canopy graphs for all rings at nominal voltage. All rings have 48 stages except for ring 9 with only 40 stages. Maximum throughput is at about 60% occupancy. The lesser maximum throughput of rings 1 to 8 reflects the slower action of the double crosser stages.

choice to move bubbles faster than data items is inspired by the relative ease of moving bubbles: it is easier to declare a stage EMPTY when moving a bubble than to drive the latches and capture an arriving data item. At 60% occupancy the net velocity of bubbles and data items match, resulting in maximum throughput. A fresh bubble and a fresh data item exchange places at each stage when the fire signal copies data into the space vacated by the bubble, vacating the space formerly occupied by the data.

The Weaver's layout accounts completely for the differences in the shapes of its canopy graphs. Within the layout of each Weaver module the GasP control circuit occupies a central row flanked above and below by two groups of 36 latches each. Thus even without any external connection, the predecessor and the successor state wires within each control circuit must extend horizontally almost all the way across the module from East to West just to accommodate its GasP control circuit. Although all modules are approximately square, modules that connect in the East – West direction have shorter state wires. East – West state wires take advantage of the internal East – West length of internal state wires. The state wires for modules that connect in the North – South direction have the shape of an upper case I and are therefore longer and harder to drive. The difference in wire length is even more pronounced in the switch modules. Longer state wires are slower because all state wire drive transistors in the Weaver are the same strength. An improved design could avoid this performance loss by adjusting transistor widths to match better to the lengths of state wires.

To understand the differences between the canopy graphs, consider first the canopy graphs for the switch-free rings 0 and 9. Ring 0 runs predominantly East-West near the equator in the floor plan of Figure 2 and so is faster than predominantly North-South ring 9. Indeed, extrapolating the linear sides of the Ring 0 canopy graph would suggest a maximum throughput of nearly 7 GDI/s. However, two North-South links in Ring 9, one at each end, limit the throughput of ring 9, producing the flat top of its canopy graph. Notice that the maximum throughputs of both Ring 0 and Ring 9 are about the same, each limited by their slower North-South links.

Next consider the canopy graphs for the switched rings 1 through 8. Their flat tops indicate the presence of one or more stages slower than the rest. The slow stages are, of course, the switch stages themselves. The state wires in the switch stages drive more load than in the stages without switching, and are also longer. Again the layout of East-West connections takes advantage of internal East-West state wire, making switches connected East-West faster than switches connected North-South. Because Ring 1 passes across the bottom of the cross bar triangle, it avoids North-South switch stages, and is therefore faster than the other switched rings.

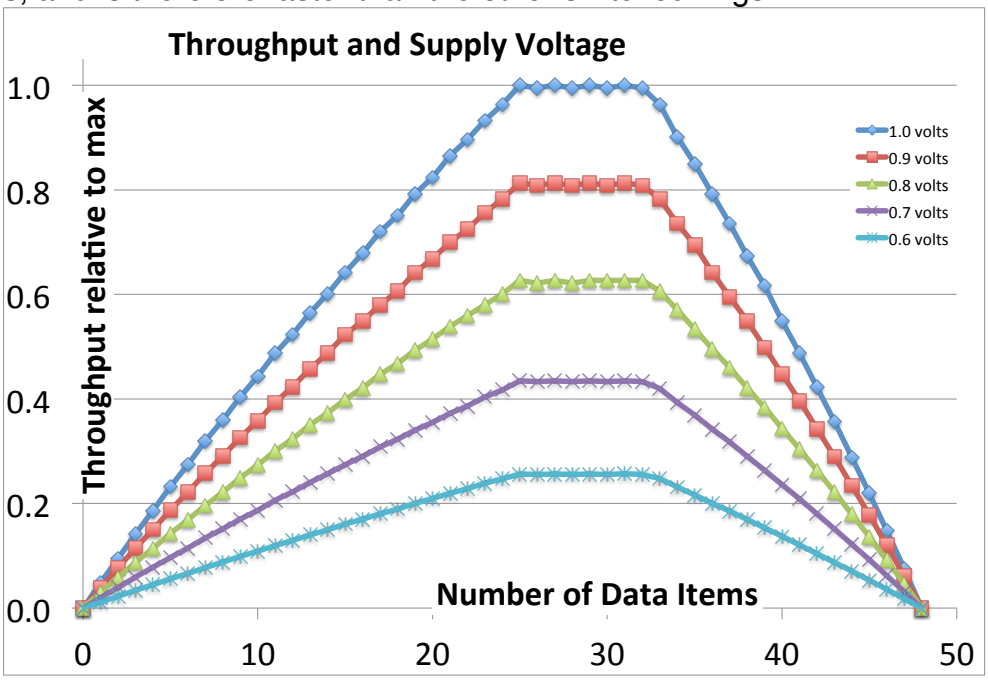

Figure 12: Throughput at various values of supply voltage. Throughput scales as Vdd – 0.55.

TEST RESULTS – THROUGHPUT vs. SUPPLY VOLTAGE

Speed scales with supply voltage. Figure 12 shows the relative throughput of ring 4 at different supply voltages. All throughputs in Figure 12 are scaled to the maximum throughput at nominal supply voltage. The equal spacing of the flat tops at different voltages indicates clearly a linear relationship between throughput and supply voltage. Throughput is very nearly proportional to the excess of supply voltage over threshold voltage. A good fit to the data of Figure 12 is that throughput scales as

Vdd – 0.55. The Weaver operates flawlessly at 0.6 volts; we have yet to explore operation below 0.6 volts or above 1.0 volts.

The low speed real time outputs connected to the counters (see Figure 4) give a vivid demonstration of speed as a function of voltage. It is unnecessary to adjust a clock frequency when changing supply voltage. Turning the knob to adjust supply voltage makes the Weaver automatically speed up or slow down because each part proceeds as fast as the available supply voltage permits. Turning the power supply voltage knob stretches or shrinks the square wave seen on an oscilloscope attached to Weaver's real time counter outputs.

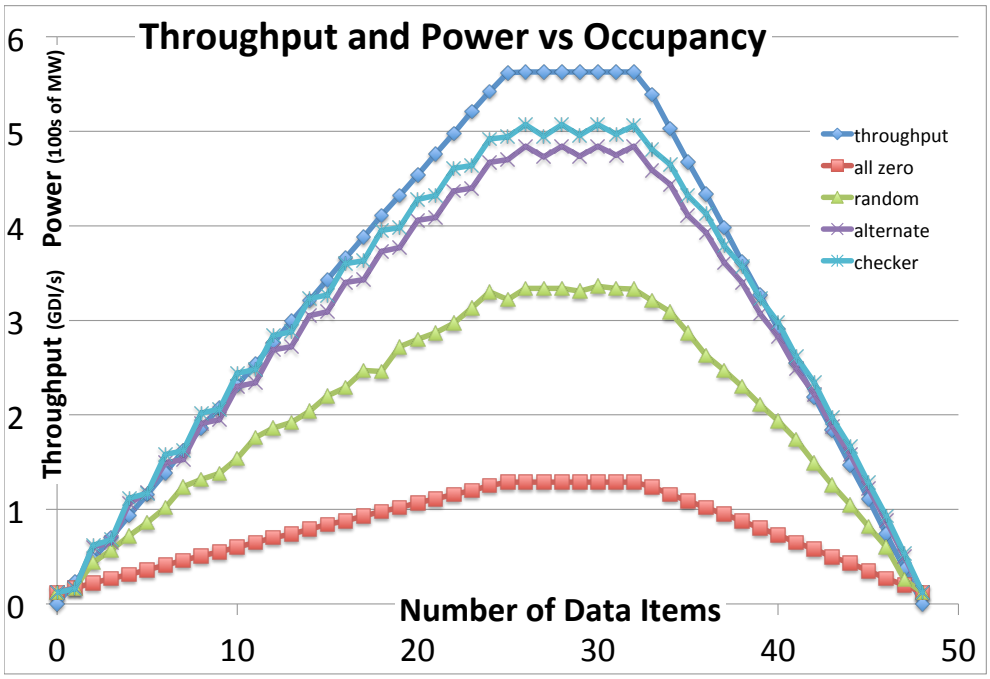

Figure 13: Power depends on how many of the latches actually change as data items travel through the Weaver. With constant data the latch outputs don't change (red bottom curve). Checkerboard data patterns like 101010 followed by 010101 cause adjacent data wires to change oppositely (light blue curve near top). All zeros alternating with all ones takes nearly as much energy (dark purple curve). Odd numbers of alternating data elements must contain a matched pair, resulting in the zig-zag shape of these two top curves. Random data values differing for each run give irregular results (central green curve). The numeric values for Throughput and for Power shown on the left axis happen to match.

TEST RESULTS – POWER

Figure 13 shows the power consumption of all 48 stages of the entire ring number 4 circulating various patterns of data items. Weaver uses the least power if all data items are identical because the data wires never change state. Weaver consumes the most power for a checkerboard data pattern in which adjacent data bits switch in opposing directions as each data item passes. Weaver consumes slightly less power if all-zero and all-one data elements alternate, reducing the impact of side wall

capacitance. The intermediate curve in Figure 13 is for random data and shows random local variation from sample to sample.

The maximum energy consumption of 500 milliwatts shown in Figure 13 is for all 48 stages of ring number 4, of which only 7 seven stages are in the crossbar. All the Weaver's ring and crossbar stages have the same number of latches driving about the same length of wire. They therefore consume about the same energy per action. Thus to estimate the power consumption of the 56 stages inside the crossbar, we could multiply the maximum power consumption of Figure 12 by 56/48 = 7/6 = 1.16. This makes 0.6 watts an upper bound on the power consumption of the 8x8 crossbar at full throughput with a worst-case data pattern. With random data, a more realistic estimate, the crossbar might consume less than 0.4 watts.

With constant data, the minimum power consumption in Figure 13 is about one quarter of the maximum. This minimum reflects the power required to make the latches repeatedly transparent and opaque even though the latch outputs don't change. This number is consistent with the length of the wire connecting the latches to their driver. That latch control wire branches extensively to reach all 72 latches. Reducing the total length of the latch drive wire would save power.

In both the alternating zeros and ones case and the checkerboard case the power consumption ripples between even and odd occupancy. For N data items circulating, there are either N or N-1 changes in value depending on whether N is even or odd, respectively. Adding one more data item to an exiting even set of data items fails to increase the number of data changes, but adding one more data item to an existing odd set of data items introduces another data change with a corresponding increase in power consumption.

The power consumption of the checkerboard and alternating data patterns differ by only about 5%. The data wires in Weaver are all double width and lie on double minimum pitch in layers 3 and 4. The curves of Figure 13 indicate that for this geometry sidewall capacitance contributes relatively little load.

The random data pattern consumes about half as much energy as the checkerboard pattern. Subtracting out the fixed overhead of moving nothing gives:

(random – allZero) / (checkerboard – allZero) = ~ 55%

over a wide range of occupancy. This is consistent with the statistical measure that a random data bit changes about half the time.

TEST RESULTS – ENERGY PER STEP

Figure 14 shows the relationship between power consumption and supply voltage. These data are for Ring 4 circulating the worst-case checkerboard pattern. The graph plots the product of supply voltage and supply current for different occupancies at five different supply voltages. The graph is normalized to maximum power consumption.

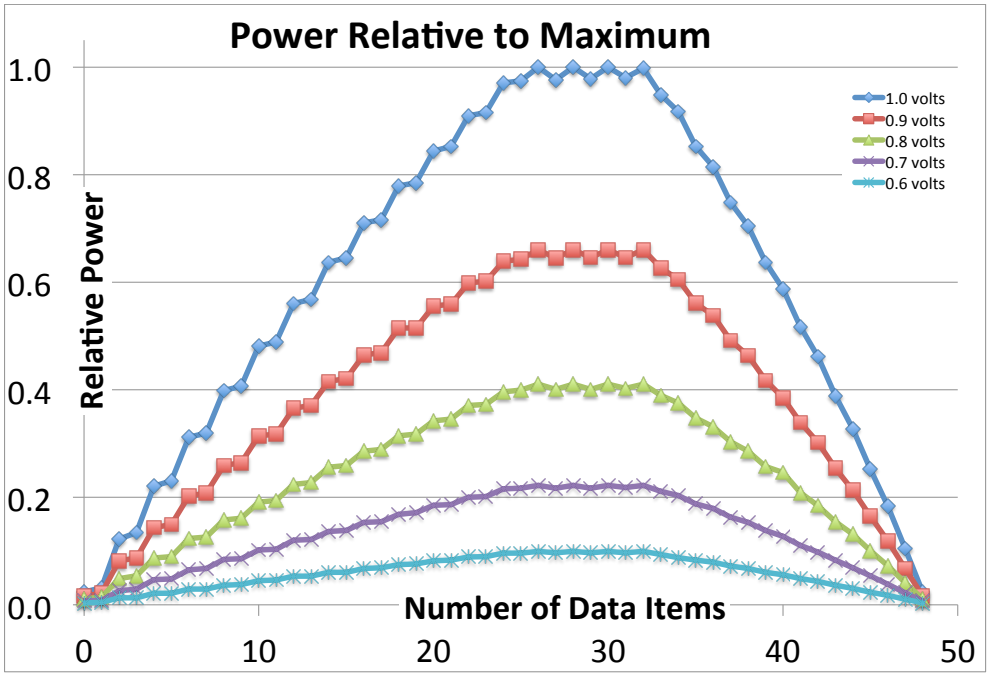

Figure 14: Power at different supply voltages. Power decreases rapidly with supply voltage because the crossbar runs both more slowly and more efficiently.

Examination of these data reveals that power consumption is very nearly proportional to 2*(Vdd – 0.5) * Vdd * Vdd. The first term, 2*(Vdd – 0.5), is the relationship between the throughput rate and Vdd; that term accounts for operating speed. The squared additional dependence on Vdd is to be expected because all loads are capacitive. We can conclude that the energy required to move a single data item forward one stage is proportional to Vdd^2.

How shall we allocate energy between the switches and the recirculating rings? As we argued above, the energy consumed in each stage in the Weaver is about the same. Consider the case of maximum throughput in Figure 12. The throughput of $5.5 * 10^9$ data items per second for all 30 data items costs about half a watt. If we call the energy required to move one data item forward one stage x_1:

$x_1 * 30 * 5.5 * 10^9$ / sec = 0.5 Joules/sec or

$x_1 * 3.3 * 10^{11}$ / sec = 1 Joules /sec or

$x_1 = 3 * 10^{-12}$ Joules = 3 pico Joules.

CONCLUSION

It costs time and energy to move data across a silicon chip. Self-timing provides no reduction in that fundamental cost. However, it does offer a way to get large bandwidth and independence from any one clock. The Weaver demonstrates switching modules that can serve as the on- and off-ramps for a high-speed clock-free on-chip communication network. We anticipate that the Weaver itself will never appear “as is” in

a commercial project. We hope its parts, however, may serve as building blocks for commercial on-chip networks.

The switches reported here can serve not only as a fast crossbar, but also as the on ramps, off ramps, and interchanges of widespread on-chip data networks. Indeed, it makes little sense to bring a myriad of data channels together in one place just to switch data between them. A more sensible design spreads out many switches similar to those tested in the Weaver to place them near to the sources and destinations of the data they switch. The resulting network on chip (NoC) can be entirely self-timed.

Widespread networks are a particularly challenging task for externally clocked systems because of their long wires and extended clock delays. We believe that self-timed circuits like those in the Weaver will greatly simplify the timing requirements of widespread on-chip communication. Self-timed networks may also provide better throughput than possible with external clocking.

ACKNOWLEDGEMENTS

We thank Oracle Corporation for funding fabrication of this chip and for its partial financial support of our work at Portland State University. Jon Lexau of Oracle smoothed the myriad details required to plan, make, mount and test the Weaver. We thank ForrestHunt inc. for its help and support in preparing this report. We thank also the private individuals whose grants to the Portland State University Foundation help make this work possible.

References:

[Seitz 1980] C.L.Seitz, System Timing, in *Introduction to VLSI Systems* (C.A.Mead and L.A.Conway, eds.), ch. 7, Addison-Wesley, 1980.

[Ivan 2001] Ivan Sutherland and Scott Fairbanks. GasP: A Minimal FIFO Control. In Proc. Asynchronous Circuits and Systems (ASYNC). pages 46-53, 2001.

[Lines 2003] A. Lines, “Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs,” in *Proceedings of the 11th Symposium on High Performance Interconnects*, pp. 2–9, Aug. 2003.

[Singh 2007] Singh, Montek and Nowick, Steven M., “MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines,” IEEE Trans on VLSI Systems, 15, 6, June 2007

[Beerel 2010] Peter Beerel, R Ozdat & M. Ferretti, A Designers Guide to Asynchronous VLSI, Cambridge University Press, 2010, Chapter 14.

[Ronk 2015] M.Roncken et al, Naturalized Communication and Testing, in Proc. Asynchronous Circuits and Systems (ASYNC) 2015