

AN AREA EFFICIENT, SCALABLE, HIGH SPEED SERIAL FULL MESH SWITCH FABRIC DESIGN FOR FPGA

**Hamish Fallside, Gautam N Kavipurapu
Xilinx Inc.**

ABSTRACT

The integration into FPGA of high-speed serial IO blocks enables the creation of flexible switching solutions that scale to meet customer requirements in a manner not cost-effective in standard products. This paper describes a Mesh Fabric switch architecture available for the Xilinx Virtex-II Pro family of devices. The design provides a configurable mesh interconnect for line cards that can be directly connected to a back plane at up to 10 Gbps per serial link. Parameters for the design include fabric cell sizes of between 40 and 256 bytes; 1-16 priority levels, with per-priority flow control; and from 4-256 mesh ports. The ability to partition the design across multiple devices of differing size provides flexible FPGA resources for additional logic. The advantages of mesh fabrics over other solutions are discussed, and the overall architecture, features, and performance of the solution are described.

INTRODUCTION

The growth of broadband-internet connections has placed increasing demands on switching equipment and technology [1]. Legacy switching solutions are often unable to meet the new cost, performance, and flexibility requirements demanded by diversification of packet switching into new application areas. A standard architecture of rack-mounted cards interconnected across a back-plane is emerging across multiple segments.

Increasingly the switching takes place across high-speed serial links that increase performance, and reduce overall complexity by minimizing interconnect between system cards. This also lowers cost and improves scalability, allowing greater reuse of protocols and the hardware and software that processes them from system to system. For packet switching, there are typically two topologies for a back-plane: STAR or MESH, and we will briefly contrast them below.

A star provides point to point connections between line cards and a central switching card. This requires an interface device per line card, and one or more devices on the switch card. Usually there are two switch cards for redundancy. The switch devices are

expensive and provide a limited number of port counts. Increasing the overall port count can require installing multiple switch cards, making incremental scalability expensive and complex as shown in [4]. In one common implementation of a star, a crossbar requires multiple schedulers [2] all communicating with each other. All of these factors can make a star architecture an expensive and inflexible solution.

In a mesh topology, all nodes are connected to all other nodes. The switching function is distributed, and the device count is equal to the number of line cards. Redundancy can be implemented using the mesh itself to overcome any individual link failures, as there are multiple pathways between each node. The mesh design we are presenting implements an output-queued switch, as analyzed in [3]. Scheduling is simpler for output queued switches.

In short, a mesh topology can provide a more flexible alternative to a star topology. It is useful where there is a high bandwidth requirement between all or a subset of line cards, such as might be found in server, DSP, or media gateway applications.

Typical Usage Scenario

Figure 1 illustrates how our mesh design is used. The implementation is for a Virtex-II Pro device, and utilizes the 3.125 Gbps RocketIO serial transceivers for the mesh interface. Each line card contains one or more mesh switch FPGA. These contain all the switching functions necessary for both the ingress and the egress directions, and provide additional resource for application-specific functionality.

An Ingress Traffic Manager takes frames for the protocol being terminated by the line card and maps them into the cells that traverse the mesh. The reverse mapping occurs on the egress side of the mesh. This mapping can be implemented in the same FPGA as the mesh switch, or in a separate device.

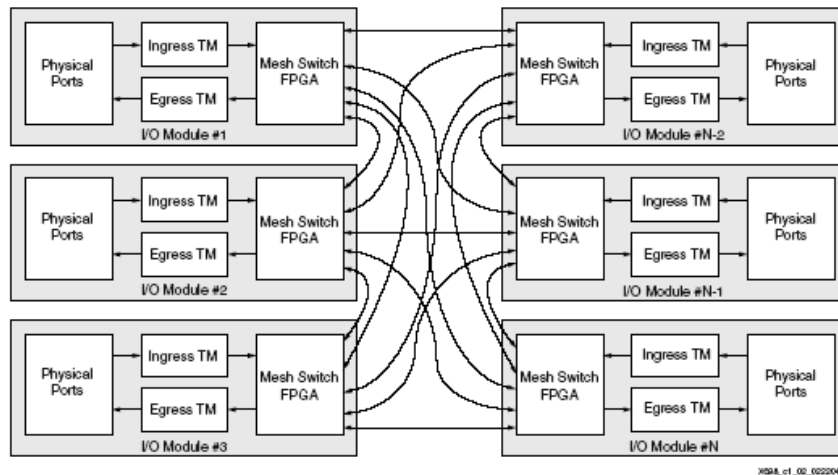


Figure 1. A typical deployment of MESH in a system

Features of the Design

- **Cell payload size of 40 to 256 Bytes**, programmable in two byte increments. The cell payload size is set statically by the user while instantiating the design.
- **Flexible Traffic Scheduling_16 traffic priorities**, Weighted Round Robin(WRR) or strict priority scheduling on egress
- **Scalable Solution (Cascade I/F)** 32-bit interface with frequency up to 200 MHz providing 6.4 Gbps of cascadable bandwidth. The cascade interface provides users of the design the flexibility to incrementally scale the number of ports.

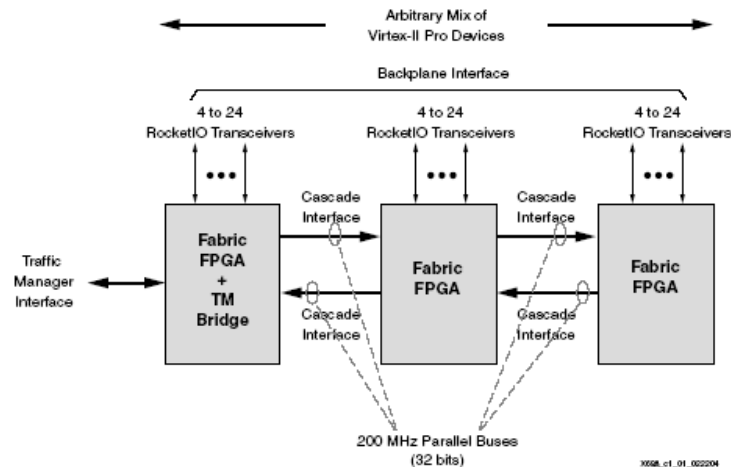


Figure 2. Cascaded Back-plane I/F using multiple fabric FPGA

MESH ARCHITECTURE

The mesh cells include a header containing information about the class of traffic, the destination port, and flow control information. The FPGA top-level functional architecture is fully described in [5], where each sub-module in the design is described in detail. The mesh ports utilize the embedded RocketIO blocks of the FPGA family, and the initialization and management of each mesh interface is performed by software running on the embedded PowerPC mcprocessor.

Ingress and Egress Path Architecture

The ingress and egress path architecture are described in this section

Ingress Path Architecture The ingress path architecture is shown in figure 3. Cells that come into the Ingress portion of the device are interrogated to ascertain their final destination. If the cell destination is another node connected to this device then the cell is routed to the port FIFO, otherwise the cell passes to the next device in the cascade.

The flow control path is shown using dashed arrows. The flow graph in figure 4 summarizes the operation of the ingress interface module.

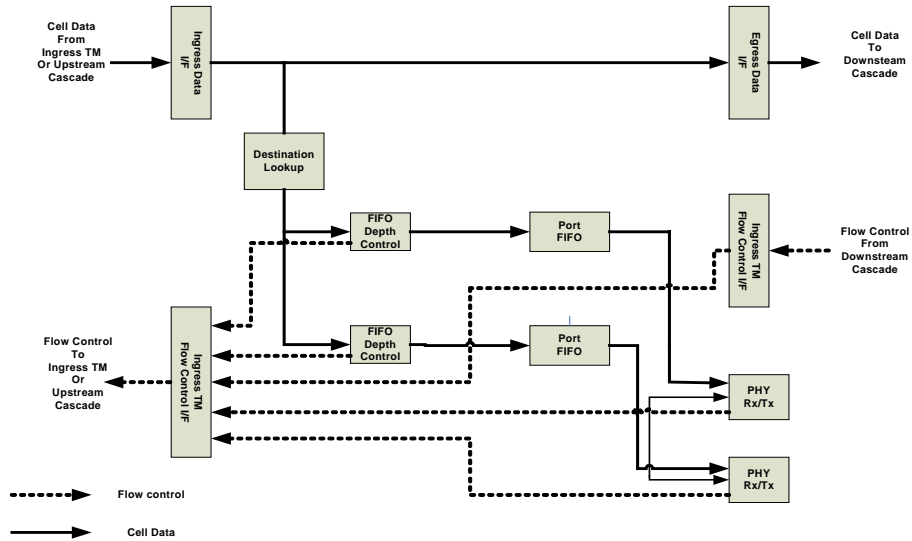


Figure 3. Ingress path architecture

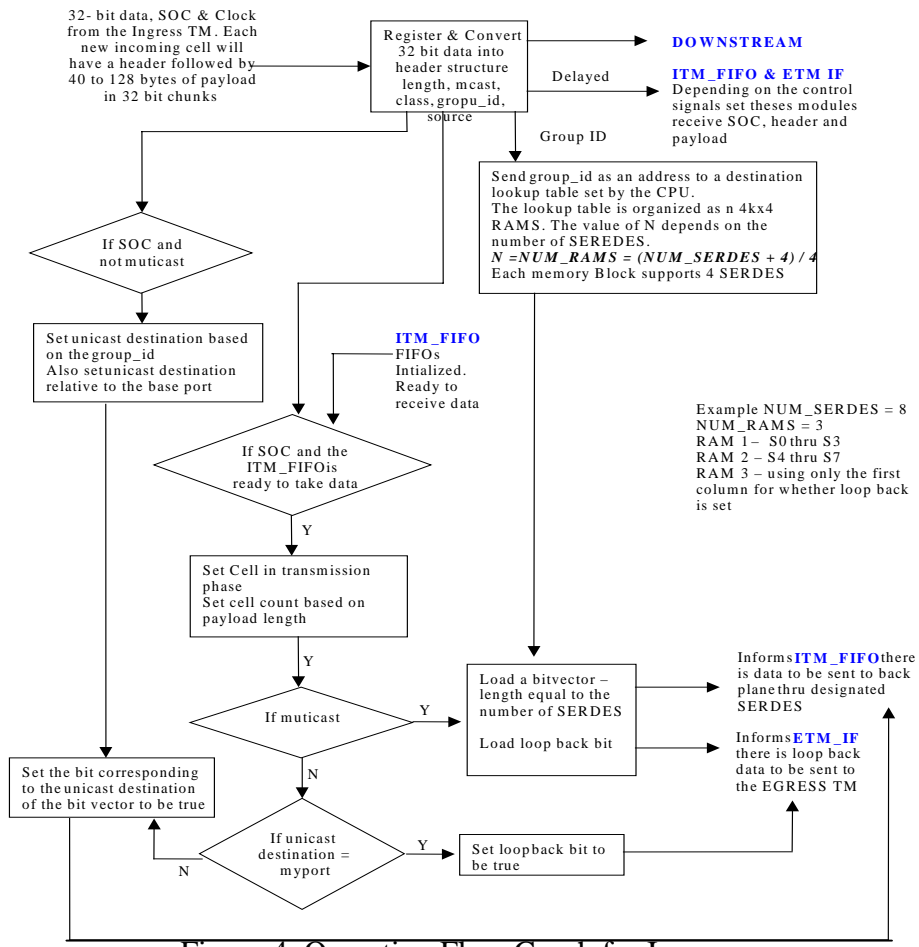


Figure 4. Operation Flow Graph for Ingress

Egress Path Architecture The block diagram of the egress path architecture is shown in figure 5. The back-plane cell data is stored in the cell rx module where CRC and other checks are performed on the data. The data is striped across these BRAM to provide for non-blocking read and write access.

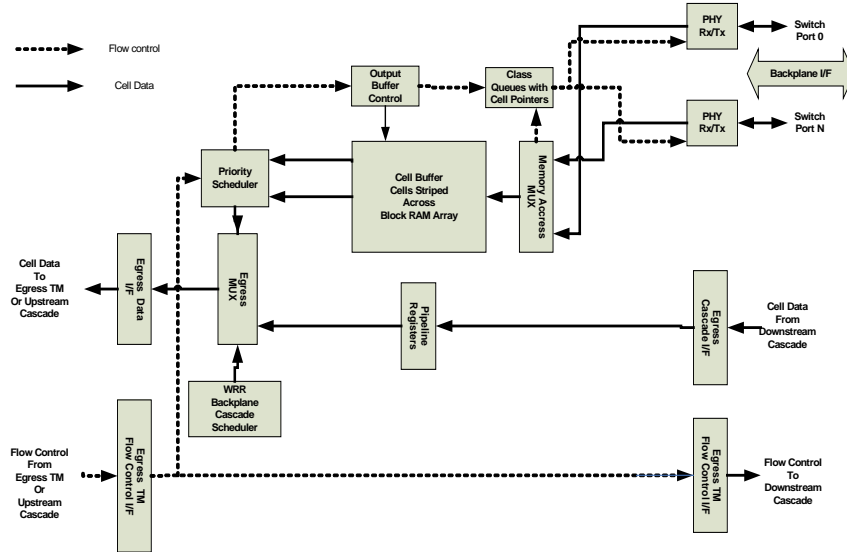


Figure 5. Egress Path Architecture

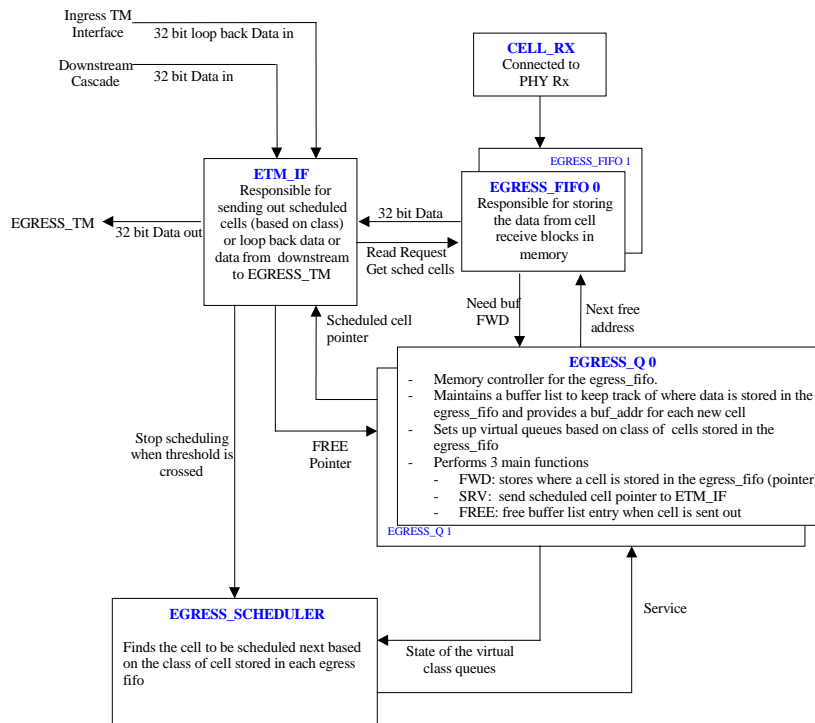


Figure 6. Operation Flow Graph for Egress

The data from downstream cascade comes in through the register pipeline and is muxed together with the back-plane data and is sent out to the upstream cascade interface or the traffic manager egress interface. The flow control path is shown using a dashed line. User set threshold levels for the output queue manager I/F and the cell buffer determine when flow control information is passed to the TM (Traffic Manager) I/F using position encoding in the cell header.

PERFORMANCE

The critical metric for any switching system is the latency. The processing chain in figure 7 shows the system level performance and number of clock cycles it takes for the cell/word to pass through each stage in the switch pipeline.

The clock numbers come from examining a simulation waveform and tracing one cell through the different modules and clock-domains. In the itm_fifo and egress_fifo modules, there are two clock domains. On the input side of the data path, the data goes directly to a block RAM that stores the data in the input clock domain.

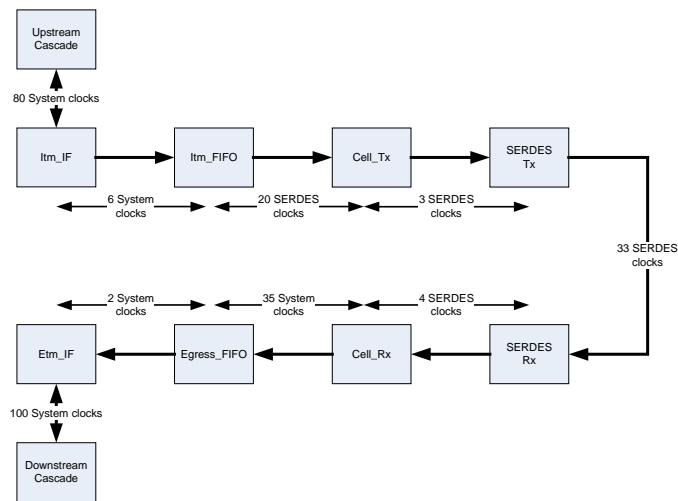


Figure 7. Clock Cycles for Pipeline Traversal

On the output side of the data path, the data is removed from the block RAM in the output clock domain and processed. The table below shows the calculated number of user clocks and SERDES clocks through the different modules. The total time through the Fabric Device in the demo system, which uses a 100 MHz system clock, is on the order of 800 – 1000 ns, with the average at 870 ns, according to the time stamping analysis done by the demo software for cells to the upstream Device only. The demo system uses a 100 MHz user clock, because of limitations in some of the EDK components and FIFOs. However, if the Fabric DEVICE system clock is run at its maximum speed of 200 MHz, the total time through the system can be extrapolated. By taking the clock cycles and frequency used for each clock, the typical time can be calculated for different frequencies.

Each SERDES/MGT clock cycle is 156 MHz, this leads to a latency number of 390 ns for 61 clock cycles. For full rate of 3.125 Gbps the transceivers need to be provided a clock that is 156 MHz in frequency.

Device utilization and efficiency of the use of DEVICE logic was the goal of this project without sacrificing the performance numbers. Fitting the design into lower cost FPGA provides an alternative market strategy using programmable logic rather than tooling an ASIC. Tables 1 summarizes Device utilization metrics for the design. The development environment for the mesh [5]:

- Synthesis = Synplify Pro 7.2.2
- Simulation = ModelSim SE Plus 5.7a (VHDL license)
- Xilinx ISE = 6.1 SP 3
- Xilinx EDK = 6.1 EDK_G.14

Table 3 shows the maximum number of flip-flops (FFs), look-up tables (LUTs), block RAMs (BRAMs), and multi-gigabit transceivers (MGTs) that can be implemented in the mesh fabric reference design based on the number of selected SERDES blocks, the number of ports, and the target Xilinx FPGA [5].

Number of SERDES Blocks	Number of Ports	Xilinx FPGA	Number of Priorities	FFs / %	LUTs / %	BRAMs / %	MGTs / %
20	20	P100	16	10998 / 12%	16143 / 18%	218 / 49%	20 / 100%
20	20	P70	16	10908 / 16%	16143 / 24%	218 / 66%	20 / 100%
16	16	P50	16	8521 / 18%	12287 / 26%	148 / 63%	16 / 100%
12	12	P40	16	7119 / 18%	10209 / 26%	145 / 75%	12 / 100%
8	8	P30	16	4608 / 16%	6548 / 23%	43 / 31%	8 / 100%
8	8	P20	16	4608 / 24%	6546 / 35%	43 / 48%	8 / 100%
8	8	P7	16	4608 / 46%	6544 / 66%	43 / 97%	8 / 100%
4	4	P4	16	3007 / 49%	4096 / 68%	24 / 85%	4 / 100%

Table 1. Device utilization for design parameters

CONCLUSIONS AND FUTURE WORK

Our team is engaged in building a framework to test and evaluate various switching designs such as the Mesh Reference design using traffic generators to feed data traffic to the design. An independent test observer type setup to log the state of the UUT on a cycle by cycle basis. We have also developed a prototype segmentation and re-assembly modules that interface the line side ports to the mesh which is being tested within the test framework.

Our design project shows that FPGA Devices do provide an alternative to ASIC designs by quick turnaround times and flexibility they provide the systems designers in building

switching solutions. Platforms can be prototyped quickly without the cycle times that are expected with a new ASIC design start.

The new platform FPGA's with their embedded memory and hard macro resources such as multipliers, shifters, FIFO and SERDES provide a very flexible platform to prototype SOC type solutions for applications in switching.

References

- [1] "The Roads and Crossroads of Internet History," Gregory R. Gromov, <http://www.netvalley.com/intvalstat.html>

- [2] "Multiple Priorities in a Two-Lane Buffered Crossbar," Nikos Chrysos and Manolis Katevenis, ICS FORTH 2003.

- [3] "An $O(\log^2 N)$ parallel algorithm for output queuing," Amit Prakash, Sadia Sharif, Adnan Aziz, UT Austin, IEEE Infocomm 2002.

- [4] "A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers," Henry C. B. Chan, Hussein M. Alnuweiri, , and Victor C. M. Leung, IEEE Journal on Selected Areas in Communications In, Vol. 17, NO. 6, JUNE 1999.

- [5] Mesh Fabric Reference Design XAPP698, Xilinx Inc. <http://www.xilinx.com/bvdocs/appnotes/xapp698.pdf>