Performance Modeling of a Network Processor Data Path Using Queuing Systems

Manivannan Kolendavelu, Satish Polisetti
Wireline Communications Department
Infineon Technologies Pvt. Ltd.
Bangalore, India
{Kolendavelu.Manivannan|Satish.Polisetti}@infineon.com

Raimar Thudt
Wireline Communications Department
Infineon Technologies AG
Munich, Germany
{Raimar.Thudt}@infineon.com

***Abstract* — Requirement of Performance Modeling during the early SoC design stages becomes increasingly important to take major system level decisions relating to hardware resources, mapping of functionality onto computing modules and selection of scheduling algorithms which affects the design significantly.**

In this work, we make use of Queuing Systems to model the Data Path of Network Processors to decide on the Internal and External memory requirements of an ASIC or Soc. We propose a framework developed using System-C, which can be used for Performance Modeling & Simulations of Network Processing Engines. The real time internet traffic patterns seen across US and Japan are used to stimulate our performance model.

We also demonstrate how the packet length distribution across US and Japan demands different memory requirements in the architecture of the Network Processor.

Leveraging on real time internet traces unlike simulating using Standard Internet MIX approximations, our methodology has an additional advantage of optimizing the SoC architectures based on the region it is being targeted. This results in an optimized architecture which saves the chip area, chip cost, data processing times and memory size requirements without compromising on the throughput requirements.

Keywords: IMIX, IP Traffic, Logical Queues, Memory, Network Processor, QoS, Queuing Systems, System C

I. Introduction and Need for Performance Evaluation

The complexity of Embedded Systems and System on the Chip (SoC) platforms leads to lot of design challenges. Stringent constraints on Time to Market, Platform Costs and Power Dissipation have driven new approaches in terms of Programmable, Heterogeneous and Parallel Architectures to deployment.

SoC Architects need to take conceptual stage decisions pertaining to hardware resource allocation, mapping of functionality on computing modules and scheduling algorithms which determine the final quality and performance. Hence, Performance Evaluation [1] must be supported at the conceptual stage to ease the above decisions which impact the resulting design quality. The platform used for the evaluation should be faster compared to conventional Register Transfer Language (RTL) simulations and provide precise quantitative performance results.

In general, Network Processors perform multiple tasks such as packet parsing classification, and forwarding. They also incorporate Quality of Service (QoS) aware functions like Buffer Management including Weighted Random Early Discard (WRED), and Bandwidth Management including Weight Fair Scheduling and Token Bucket Shaping. Hardware design of these components involves integration of In-House RTL, External IPs, Internal and External chip memories.

We leverage on Queuing Systems concepts to model Network Processors and make productive concept phase decisions. Queuing Systems consists of queues and servers. A queue is similar to a waiting room into which requests are pumped in and wait until the server becomes available for processing.

Entertainment and real-time applications like voice-over-IP, medical telemetry, network gaming and streaming video are quickly becoming prevalent applications over packet-based communication networks. We henceforth used Real time Internet Traffic patterns [13] and utilized the varied profiles as stimuli into our model.

The highlights of our abstract modeling gives the ability to effectively use the hardware resources for each geographic region and come with tunable architectures catering to the Internet traffic requirements of that part of the globe.

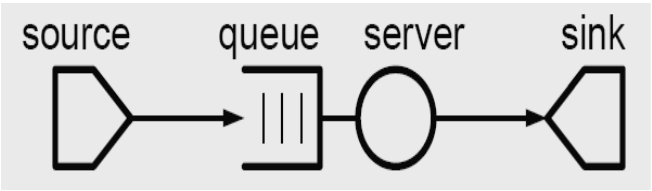

Fig 1a A Queue and Server in Queuing Systems

II. Overview of Queuing Systems

Our target here is to model the performance aspects in addition to the control flow and data flow. Queuing Systems which are used in Operations Research is a right fit with its scaffold built on stochastic analysis of system behavior.

Queuing Systems basically consist of "Queues" and "Servers" as shown in Figure 1a. "Queues" can be visualized as waiting rooms where requests are stored until a 'Server" becomes available for processing the requests.

There is a finite time taken by the Server to process the request which is denoted by Tservice. This service time could be constant for all requests or could be a time distribution or vary with each transaction class. The queues deliver transactions to the servers based on different scheduling disciplines such as Round Robin, Service In Random Order (SIRO), First Come First Served (FCFS), and Processor Sharing (PS) etc. Queues do not modify or alter the transactions, but the transactions spend certain simulation time within the queue in case if other transactions are being served at the time of arrival. Theoretically, Queues' storage capacity can be unlimited, but performance modeling helps us to arrive at the ideal buffer sizes (or internal memories) to prototype the Queue functionality in hardware. Other advantages of adopting Queuing Systems are the ability to determine system throughput, optimized queue length and server loads.

III. SystemQ based Simulation platform and Modelling refinements

Simulation based evaluation of queuing networks offers several advantages over analytical solutions. Complex queuing networks must be used for more realistic models where several queuing systems are connected to each other. Often, non-exponential arrival and service times are required for increased accuracy, up to request dependent service times for different servers. These levels of detail are often difficult or even not feasible for analytical treatment.

Moreover, with simulation we can track a single request all through the system enabling end to end delay analysis on the transaction class. The performance evaluation framework SystemQ [1,2] features a rich SystemC class library, as System-C has become a widespread open source modeling language for embedded systems spawning design and verification on different abstract levels from concept to implementation in hardware and software [7]. It follows a structured implementation style using C++'s inheritance features. Particular functionalities, like queuing disciplines (in case of queues), is implemented in separate library elements to increase re-use. Furthermore, elements may be implemented in different abstractions to enable systematic refinement steps.

SystemQ provides support for (re-)configuration, stimulation, and verification of queuing models. The configuration of simulation models is based on parameter files. Thus, recompilation of the model can often be avoided in case of parameter explorations. The framework furthermore includes modular and extensible means for generating realistic traffic loads on different levels of abstraction. A number of elements can be composed to form layered sources, e. g. packet generators. Following the OSI/ISO approach such sources first generate a chunk of data based on a particular distribution (time, length) which then are encapsulated/transformed repeatedly depending on the desired protocol stack. Similarly, layered sinks can be employed, that may be used for workload analysis, such as packet arrival rates and distributions.

We exploit SystemC's 2.0 communication support and its discrete event simulation capabilities for our timed queuing models by providing component and communication libraries as explained in this section. Performance indicators, such as latency or throughput, of the overall system or individual building blocks can be derived and potential bottlenecks can be identified. The main focus of analyzing queuing systems is on statistics such as queue lengths, server utilization, and residence times of transactions in the system. These properties depend on each other and are affected by transaction arrival processes, transaction classes, and queuing disciplines.

With SystemQ's simulation approach we gain additional information when we can trace time dependencies, profile individual resources, and follow the processing of particular transactions through the model. The quality of results depends on the abstraction level.

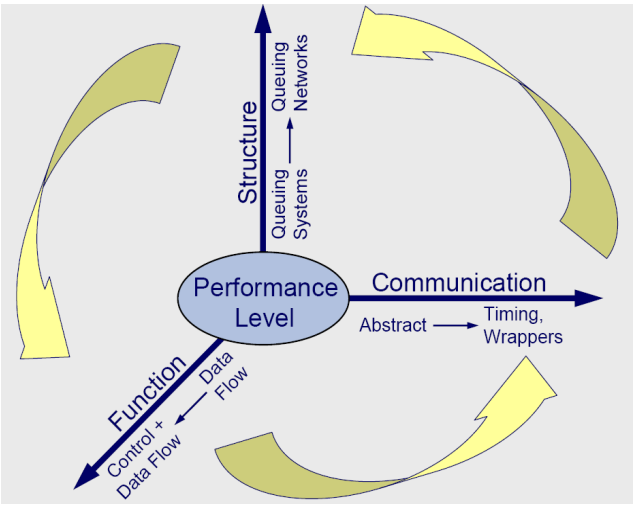

Fig 3a Refinement Strategies

Initially, a rough estimation of the overall system performance is required. Refinement enables better simulation results by adding details either in function, structure, or communication. These refinement strategies are considered to be orthogonal [9]. SystemQ enables the designer to apply these strategies independently within a module as well as in separate modules, e. g. in order to connect a functionally refined [3] queue with a performance-level server. Figure 3a shows an overview of the three refinement strategies that are defined as follows:

• Functional refinement may add (and refine) the function of servers and/or introduce distinguishable transaction classes. A refined server switches transactions to different output ports depending on their class (similar to colored tokens in Petri nets).

• Structural refinement decomposes systems into queuing networks. This form of refinement is often motivated by incorporating details of the architecture in the model.

• Communication refinement differs from the above as it adds precision to communication between queuing systems or changes communication semantically.

IV. Internet Packet Size Distribution Across US and Japan

With Internet connectivity and web-usage becoming nearly ubiquitous, many day-to-day activities have migrated to the Internet. User expectation of Internet use has also evolved from "best-effort" connectivity to a high performance medium meeting the bandwidth requirements/demands for all types of applications. Service providers and consumers alike would prefer to use a single "pipe" for all their communication and entertainment needs, if possible[10].

Quality-of-Service (QoS)-sensitive applications like VoIP, IPTV, Internet gaming, VPN, Peer to Peer and telemedicine are being offered over converged IP networks. Each of these applications demands different IPv4/6 packet lengths. Moreover it is interesting to note the penetration of these applications being entirely different in different parts of the globe [12]. Table1 [4] gives a rough overview of the distribution of different Internet applications across geographical boundaries [4] as on 2007 in Petabytes per month. Hence the densities of the IPv4/6 packets also vary across geographical tracts along with Packet Lengths.

Studies [5, 6] indicate that Internet traffic consists of fixed percentages of different frame sizes at a given point and time. IMIX (Internet MIX), as shown in Table2 is widely adopted packet distribution for quick approximations. It has an average packet size of 340.3 bytes and correlation of 0.892. But this approximation is far from the current realistic Internet Traffic Packet Distribution. CAIDA, the Cooperative Association for Internet Data Analysis [11], provides tools promoting the engineering and maintenance of a robust, scalable global Internet infrastructure.

Figure 4a shows the graph with IPv4 packet length distribution across US (Eq-Chic) and Japan (WIDE) as on 2008 [13] which are published by CAIDA. The "Eq-Chic" is a sampling monitor in Chicago, USA. The WIDE is a project by another traffic data repository group, MAWI in Japan [14]. The total number of IPv4 packets sampled is 1.75G and 3.82G across US and Japan respectively.

TABLE1: INTERNET APPLICATION TRAFFIC (IN PETABYTES PER MONTH, PB/MONTH), 2007

	Web,Email	P2P	Gaming	VoIP	Video
NA	209	416	17	6	4
Europe	153	378	30	10	7
APAC	244	773	71	14	12
Japan	42	63	11	6	1

TABLE2: STANDARD IMIX RATIOS

Packet length	Proportion Priority
(bytes)	(parts)
64	7
594	4
1518	1

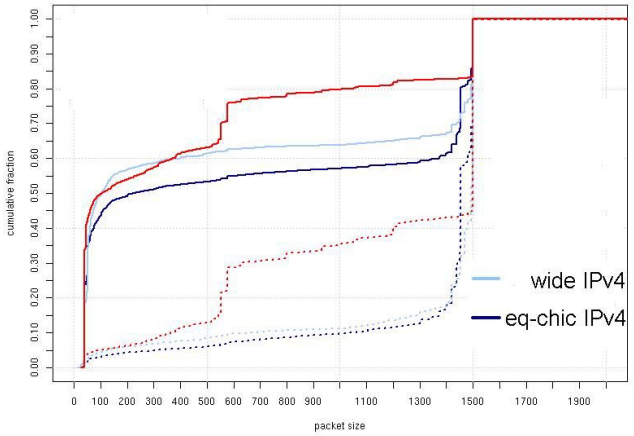

Figure 4a – IPv4 packet length distribution across US (Eq-Chic) and Japan (WIDE), 2008

The thick contiguous lines represent the cumulative packet size distribution by packets and the dotted lines represent the same by bytes. For the thick curve, the y-axis represents the cumulative fraction in terms of number of packets and in the case of dotted curve, it represents the cumulative fraction in terms of number of bytes.

The first thick contiguous line and the dotted line represent the distribution for fix-west[13], which is not considered for our analysis. The variation in the distribution is due to the varied demand of Internet applications in these regions as explained above.

Similarly Figure 4b shows the distribution of IPv6 Internet packets across these regions as on 2008. The total number of IPv6 packets sampled is 76K and 11M across US and Japan respectively [13]. The third and the fourth thick contiguous lines show the distribution of the WIDE and Eq-Chic IPv4 in packets. The dotted lines show the same distributions in terms of bytes.

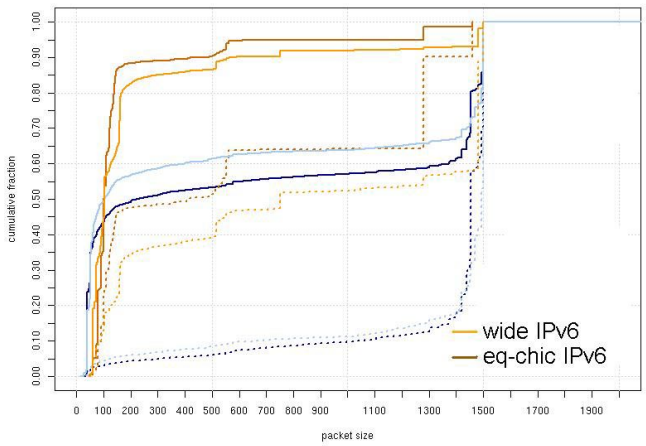

Figure 4b – IPv6 packet length distribution across US (Eq-Chic) and Japan (WIDE), 2008

The ranges of packet lengths were chosen in such a way that significant variations could be caught from the packet distribution patterns.

Both graphs map the Packet Lengths with the Cumulative Fraction. In order to get the precise percentage of Packet Lengths between different ranges, we used the empirical distribution function [15] Fn(x) - based on sample X1 ... Xn, a step function defined by

$$F_n(x) = \frac{number\ of\ elements\ in\ the\ sample \le x}{n}$$

$$or\ F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \le x)$$

where I (A) is the indicator of event A.

The percentage of data points between Xp and Xq would be {Fn (Xq) – Fn (Xp)}*100 and the exact number of packets in this would be {Fn (Xq) – Fn (Xp)}*n. The results of these calculations on the thick curves of the graphs (cumulative packet distribution) presented in Figure 4a and Figure 4b are tabulated in Table 3 for IPv4 and Table 4 for IPv6.

These distributions are used as the stimuli for our simulations. Other geographical patterns could also be gathered and similar analysis would give the required stimulus for Performance modeling using Queuing Systems pertaining to that region. This kind of profiling has inherent advantages over modeling using traditional Internet IMIX as results are much closer to real time internet traffic of a particular region.

The next section describes a block diagram of a simplified Network Processor SoC and an equivalent model using Queues and Serves. Later the above stimulus will be used for simulation of the model.

TABLE 3: IPV4 PACKET DISTRIBUTION ACROSS US AND JAPAN, 2008

US: "Eq-Chic"	**No of Packets in G**	**Percentage of IPv4 packets**
64 - 146	0.8225	47%
147 - 530	0.1225	7%
531 - 1426	0.1225	7%
1427 - 1518	0.6825	39%
Japan: "WIDE"		
64 - 146	2.1774	57%
147 - 530	0.191	5%
530 - 1426	0.2674	7%
1427 - 1518	1.1842	31%

TABLE 4: IPV6 PACKET DISTRIBUTION ACROSS US AND JAPAN, 2008

US: "Eq-Chic"	**No of Packets in K**	**Percentage of IPv6 packets**
64 - 146	41.8	55%
147 - 274	25.08	33%
275 - 658	4.56	6%
659 - 1298	1.52	2%
1299 - 1518	3.04	4%
Japan: "WIDE"		
64 - 146	6.05	55%
147 - 274	3.08	28%
275 - 658	0.66	6%
659 - 1298	0.33	3%
1299 - 1518	0.88	8%

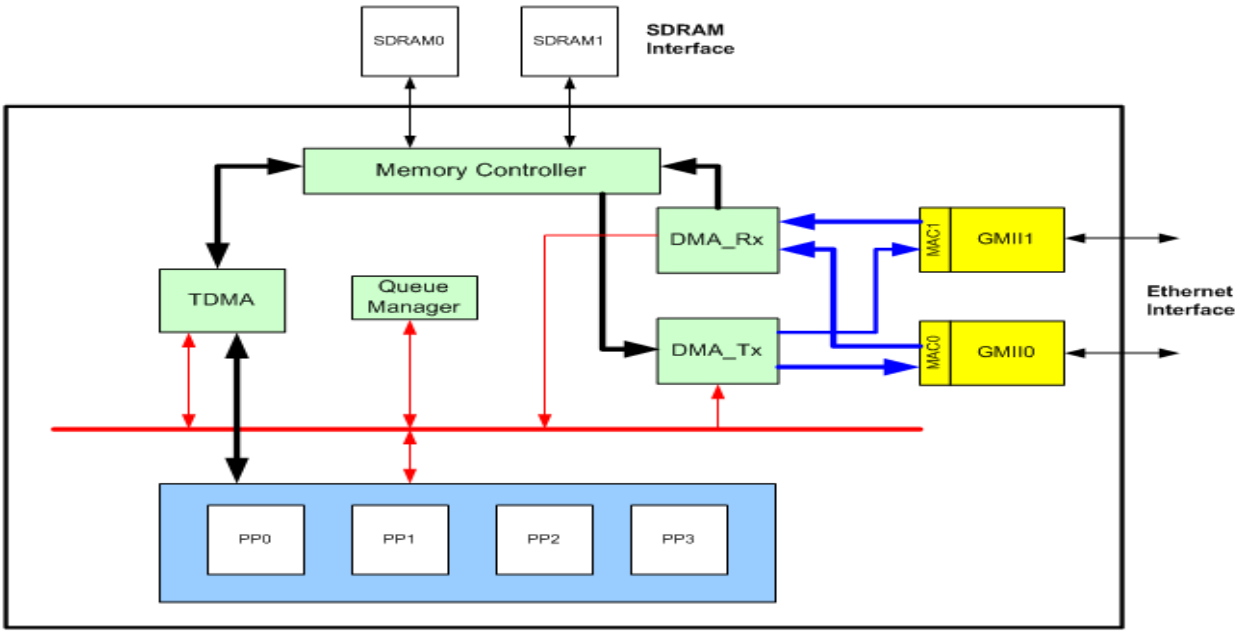

Fig 5a Block Diagram of a Typical Network Processor SoC

V. PERFORMANCE MODELING OF A NETWORK PROCESSOR USING QUEUING SYSTEMS

A. *Typical Network Processor Data Path*

Figure 5a shows a typical architecture of a data path in a Network Processor. It consists of two Gigabit Media Independent Interface (GMII) ports with capability to send or receive packets at the rate of 2 Gbps. It has an interface to the external memories (SDRAM0/1) which is the packet buffer. Many other interfaces and functionalities are omitted for the sake of simplicity. The entire data path design works based on logical queues. Queues explained here should not be confused with the Queuing Systems that was explained in the previous chapters.

The maximum number of queues that can exist is based on the particular design or application. Each queue represents a collection of packets, which are actually physically stored in any one of the external SDRAMs. Queue Manager manages different logical queues. Any master can access the queue only through the centralized Queue Manager.

Write master adds logical entries to the queues after storing the packet physically to the external memory. Read masters will read the queue entry to understand the SDRAM address in which the packet is stored. It then reads the packet from SDRAM and removes the entry in the queue.

DMA_Rx is the write master and DMA_Tx is the read master. Transfer DMA (TDMA) can perform both write and the read from the queue. Packets are accessed from the external memory through the Memory Controller.

The Packet Processor Cluster (PPC) contains Packet Processing Engines (PPn). Each Protocol Processor (PP) can work independently to process the Ingress Packet. All PPs use only the first few bytes of the packets to do its processing.

A simplified data flow is as follows. DMA_Rx receives the Ethernet packet and segments it to the internal format, stored in the internal memory. It does a basic parsing and then stores the packet into the external memory (SDRAM). It also stores some additional useful information about the packet into the first segment of every packet other than the payload data. Because of the fixed segment size, this additional information along with un-occupied bytes of the segment (in case if payload data is less for that segment) constitutes the overhead bytes for that packet. After parsing, DMA_Rx will get to know to which queue the packet has to be stored. It adds an entry to that particular Ingress queue. As soon as an entry is available in the Ingress queue, any PP in the PP cluster can request TDMA to fetch the Packet from the SDRAM for processing. After it finishes the processing, it will request TDMA to store the processed packet to the SDRAM. It also adds entry to the Egress Queue. DMA_Tx will sense the

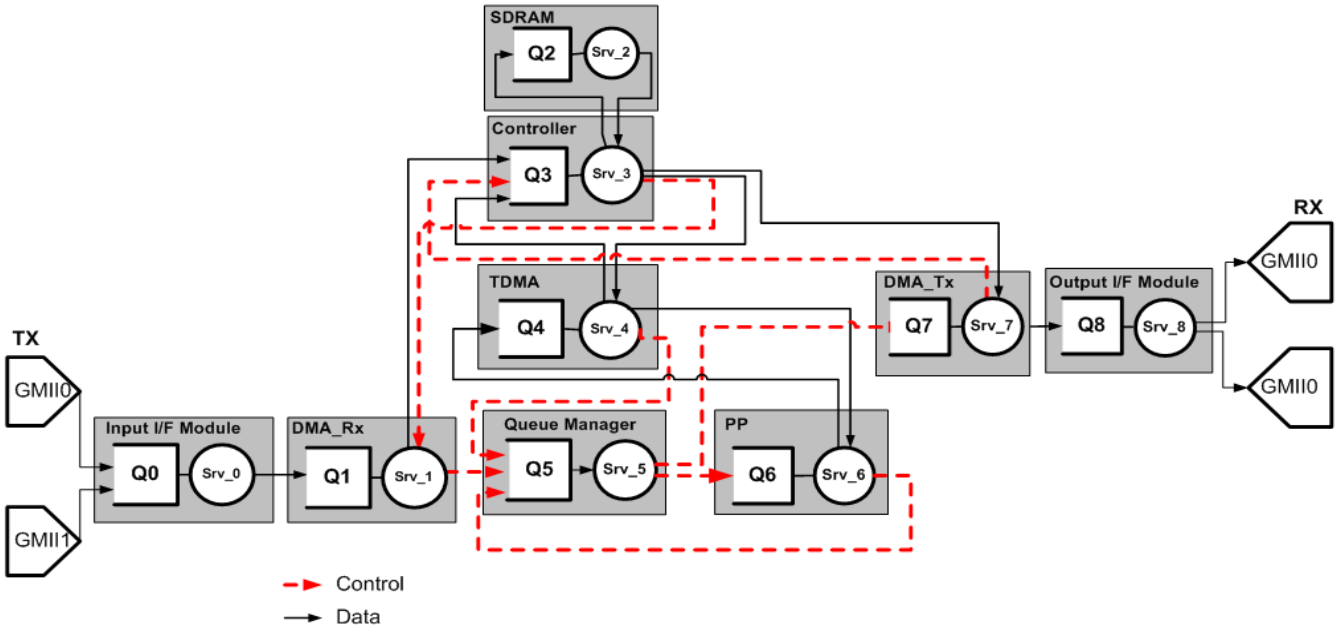

Fig 5b Representation of Control and Data Path of Network Processor SoC using Queuing Systems

packet availability from the Egress Queue, and will fetch that particular packet from the SDRAM. It transmits the packet to the GMII and will remove the entry from the Egress Queue.

B. SystemQ Model of the Network Processor Data Path

Figure 5b represents the functionally refined Network processor data path using SystemQ. Each module is represented by a queue and server combination. "Qn" and "Srv_n" represent Queue and Server respectively. Traffic Sources and Sinks (TSS) are connected to the DMA_Rx and DMA_Tx respectively. These TSS are on/off sources, which are capable of generating and prototyping real time random bursty Ethernet traffic. The TSS can be constrained to generate different peak rates and sustainable rates. The thick lines represent the data flow and the dotted lines represent control flow. For a data write operation, thick lines represent the write request with data and the dotted lines represent the write response. For a data read operation, dotted lines represent the read request and thick lines represent the data with acknowledgement response.

Ethernet packets arrive at Q0 of the Input I/F module and are delivered to server Srv_0. It resides in server Srv_0 until the service time set for that server expires and then gets queued to the Q1 of DMA_Rx. In Srv_1, the complete packet is segmented into multiple smaller segments and each segment is stored into SDRAM. The thick line connecting Srv_1 to the SDRAM controller is the request to store the segment into SDRAM. After the successful storing of the segment into SDRAM controller is the request to store the segment to SDRAM, Srv_3 sends an acknowledgement to Srv_1, which is denoted by the red dotted arrow from Srv_3 to Srv1. Srv_1 initiates the next segment transfer to SDRAM only after receiving the acknowledgement for the previous segment. After storing all the segments of the packet to the SDRAM for that packet, Srv_1 indicates to Queue Manager's queue, Q5 that the packet is ready to be processed by PP. Queue Manager maintains two parallel logical queues within Q5, Ingress queue and Egress queue.

Ingress queue contains the information about the packets that has to be processed by the PP, and Egress queue contains the information about the packets that has to be read and transmitted by DMA_Tx. Once the packet information arrives in the Ingress queue of Q5, this will be indicated to PP which is the dotted line from Srv_5 to Q6. PP then requests TDMA to fetch a packet form the SDRAM for processing, which is denoted by the thick line from Srv_6 to Q4. After PP finishes the processing, it stores the packet in the SDRAM with the help of TDMA. Srv_6 of PP will send a request to TDMA to store the processed packet. After the processed packet gets successfully stored, the Srv_4 will indicate this to Srv_6. Srv_6 adds the packet information to the Egress queue of Q5. This is indicated by the dotted line from Srv_6 to Q5.

Srv_5 of Queue Manager will indicate to DMA_Tx about the availability of processed packet in the SDRAM that has to be transmitted, which is indicated by the dotted line from Srv_5 to Q7. Srv_7 will then request packet from Q3, which will be received from Srv_3. In Srv_7, the segments are

assembled to packets and will be transmitted to the Output I/F Module. For simplicity only one PP and one SDRAM are shown in the figure 5b. For the actual simulation, two SDRAMs and four PPs are used.

C. *Simulation setup*

The service times for all the servers are estimated based on the individual module specification. It equals the mean packet latency of the module, summing up the estimated latencies of all functional sub-blocks inside that module. Two TSS are used to represent two Gigabit Ethernet ports. The sustainable rate of each TSS is set to 1Gbps, and the peak rates are varied from 1Gbps to 2Gbps in steps of 100 Mbps for the individual TSS.

The internal segment size of the packets is derived based on the system level specification. The simulated time period for each simulation is 500 milliseconds.

At the end of the simulation, the maximum sizes of all the queues are observed. The maximum size of the queue implies the maximum size of the memory that would be required to buffer that data for that particular module. Hence this simulation results can be used to arrive at the size of internal and external memories for different traffic conditions.

In this particular simulation model, the internal memory requirement is derived from the filling level of queues Q0, Q1,Q3,Q4,Q5 and Q8 and the external memory requirement is derived from the filling levels of Q2,Q6 and Q7. The fill level of Q6 indicates the number of packets that are stored in the SDRAM and queued for PP processing. The fill level of Q7 indicates the number of processed packets that has to be transmitted towards TX side.

D. *Simulation parameters*

Simulations are run with different packet length mixes. Apart from the Standard IMIX proportion, the real time traces from US and Japan (IPv4 and IPv6) are also used as stimuli in the simulations to arrive at the accurate resource requirement pertaining to that region.

For each internet packet length mixes (including Standard IMIX), the peak rate is varied from 1Gbps to 2 Gbps in the steps of 100 Mbps. The sustainable rate is kept as constant to 1Gbps. TSS is configured to generate traffic of ON/OFF type. TSS can be configured with the minimum and maximum burst size in terms of packets. In this experiment, both are set to 100 packets. Hence a burst of 100 packets will be generated to achieve the specified peak rate. To be more realistic, different values can be configured for the minimum and maximum burst size. Hence the TSS routine can generate random burst sizes between the minimum and maximum burst sizes specified.

Protocol processor takes 1000 cycles to process one complete packet. In reality, the processing time would vary based on the packet type and its contents. In the further refinement steps, the processing time can be a varying factor based on the traffic class.

In this architecture, the service time of the Protocol Processor, which represents the speed and the amount of processing governs the external memory requirement. DMA_Rx is the governing factor for the internal memory requirement, for which the service time is set as 64 cycles based on its functionality.

The service time of the other servers are arrived based on the individual module specification and the specified operating frequency. Since one of the intentions of the simulation is also to get the maximum size of all the memories, infinite memory sizes are used to avoid packet discards or back pressure. Ingress rate and the egress rate are also observed for the different traffic conditions

VI. Analysis of Simulation Results

The various parameters captured at the end of our simulation include SDRAM Load, Rx and Tx Logical Queue Sizes of both DMA_Rx and DMA_Tx (Figure 5b). We also captured the Queue Size build up at the "Controller" and "PP" blocks of our Model. This simulation is carried out with different internet mixes - IPv4_Japan, IPv4_USA, IPv6_Japan and IPv6_USA. The memory size requirement is arrived based on the maximum queue size of the Queues for the entire simulation time. The rate at which the queue size grows is a function of the processing speed of the corresponding server.

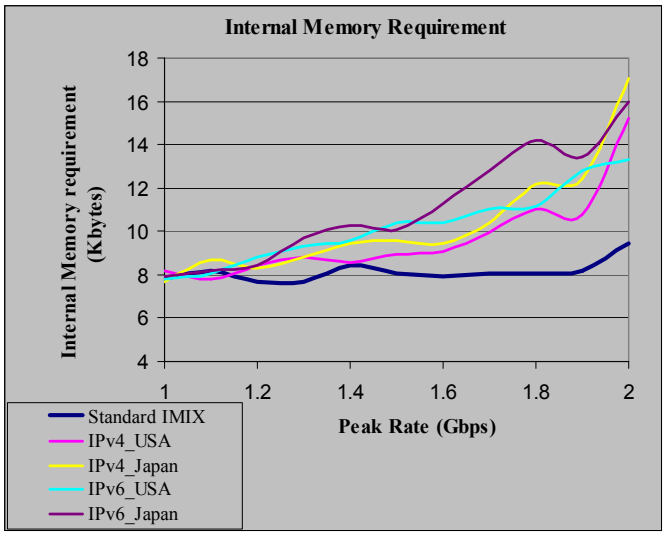

Fig 6a Internal Memory Characteristics showing the contrast between Standard IMIX and Real Time IP Traffic Profiles

Figure 6a shows the Internal Memory Requirements plot as a function of different Peak Rates. For the sake of simplicity, the memory requirements for all the modules are summed up and plotted in the Internal Memory Requirement. These include Queues of DMA_Rx, TDMA, Controller and Shared Segment Buffer (not shown in Figure 5b). On the other hand, knowing the individual modules buffering

requirement will be helpful in determining the memory sizes of all the modules in the architecture.

From the Internal Memory Requirement plot, we can clearly see the thick line of Standard IMIX is below the plot of IPv4 and IPv6 plots of Japan and USA. This clearly shows that the approximations taken by IMIX do not reflect the actual Memory Requirement needs of Real Time IP Traffic. IMIX correctly approximates at lower Peak Rates up to 1.17 Gbps but deviates for higher Peak Rates. The difference between the Real Traces and IMIX accumulates as we approach the usually targeted 2 Gbps QoS requirements. A maximum of 8KB difference is seen in IPv4_Japan with respect to IMIX. IPv4_USA, IPv6_Japan, IPv6_USA also show significant difference of 5KB, 7KB and 4KB respectively.

Figure 6b shows the External Memory Requirements plot as a function of different Peak Rates. It is apparent to sum up the Rx Queue size of DMA_Tx and PP Queue Size under External Memory Requirements. This External Memory Requirement is influenced only by the packet processing bottleneck. The requirement would be higher if we take the Egress line rate congestion also into consideration.

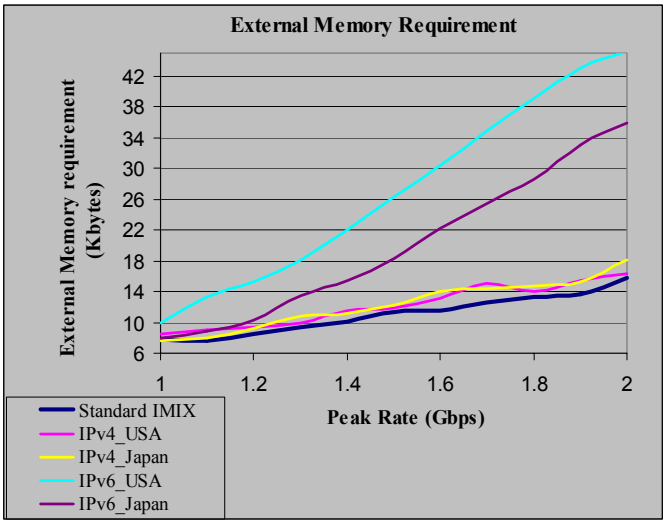

Fig 6b External Memory Characteristics showing the contrast between Standard IMIX and Real Time IP Traffic Profiles

From the External Memory Requirement plot we can observe that the memory requirement approximations by IMIX are much below the Real Time IP Traffic. IMIX approximations closely follow IPv4_USA but show variation of 2 KB for IPv4_Japan. The scenario in IPv6 is more evident with 21 KB in case of IPv6_Japan and a maximum of 30 KB in case of IPv6_USA. This effect is because of the more concentration of shorter length packets. The shorter the length of the packet, the more the overhead bytes, hence it demands higher memory bandwidth and storage space. This overhead is because of the fixed segment size that has been used.

The complete exercise depicts a range of Memory Requirement variation from 2 KB to 30 KB which would substantially affect the performance throughputs. The mixes covered in our modeling and simulation is restrained to IPv4 and IPv6 protocols in Japan and USA. These internet mixes are just taken as examples, but for thorough study, more internet mixes can be considered. The additional advantage is to optimize the architecture based on the regions for which the architecture is targeted. An example of this would be the comparison between IPv6 data between Japan and USA. A SoC designed for USA would need ~ 30 KB of SDRAM, whereas in a SoC targeted for Japan would require ~ 20 KB of External Memory. This saving of ~10 KB immediately reflects in Cost and Power consumption of the total SoC.

Also plotted in Fig 6c is the SDRAM bandwidth requirement as a function of Peak Rates and different traffic mixes.

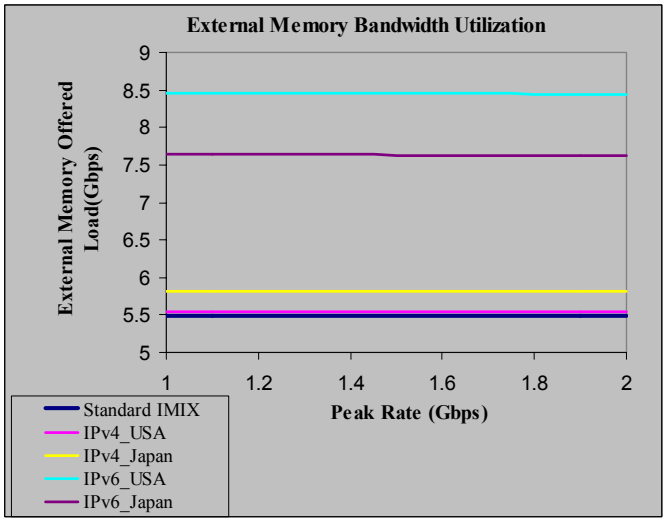

Fig 6c External MemoryBandwidth Utilization showing the contrast between Standard IMIX and Real Time IP Traffic Profiles

From this plot, we can observe no variation in the Offered Load of the External Memory within a Traffic Mix for varying Peak Rates. More significant is the variation in the offered load for different Traffic Mixes. IPv4_Japan and IPv4_USA are little closer to the Standard IMIX offered load, but IPv6_Japan and IPv6_USA show more deviation of 2 Gbps and 3 Gbps respectively. Higher Offered Load can be achieved by either using a single high speed grade SDRAM or using multiple low/medium speed grade SDRAMs. This approach of bandwidth analysis helps in determining the number of external SDRAMs at the conceptual stage. Further, as we did not observe drops in the throughput for different Peak Rates with two SDRAMs in the model; these characteristics were not plotted.

VII. Conclusion

Significant efforts are put in modeling of Network Processors but the current practice of simulating the models with a standard IMIX would not result in optimized architectures as the Internet Traffic patterns across different geographical regions vary. In order to adhere to strict QoS across different boundaries, tuning of Architectures with respect to Hardware Resources becomes essential. We have developed a basic Network Processor SoC model and also presented a methodology to evaluate the performance of a Network Processor Data Path at the concept phase profiling the Memory Requirements for different regions.

The methodology accounts the geographical trends of internet usage and come with internet traffic profiles for each region. These profiles were used to simulate our Network Processor Data Path and demonstrate the variation of Internal and External Memory requirements for architectures targeted to specific regions. Similar simulations could be carried out to identify the patterns of additional performance parameters such as number of processing engines, pipelining network of PPs, and scheduling algorithms etc.

The benefits of carrying out this methodology are seen in terms of saving of chip area and cost without sacrificing on the throughput requirements. Our methodology of performance modeling can be applied for any communication device during the conceptual phase to arrive at refined and optimized architecture.

VIII. References

[1] S. Sonntag, M. Gries, and C. Sauer, "SystemQ: A queuing based approach to architecture performance evaluation with SystemC", Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS V), volume 3553 of LNCS, 434–444. Springer-Verlag, July 2005.

[2] S. Sonntag, M. Gries, and C. Sauer, "SystemQ: Bridging the Gap between Queuing –based Performance Evaluation and SystemC", Journal of Design Automation for Embedded Systems, Springer, The Netherlands, January 2007.

[3] S. Sonntag, M.Gries and C.Sauer, "Performance Evaluation of Packet Processing Architectures Using Multiclass Queuing Networks", Proceedings of 39th Annual Simulation Symposium, ANSS 2006.

[4] "Cisco Visual Networking Index – Forecast and Methodology, 2007 - 2012" and "Approaching the Zettabyte Era", White Papers, Cisco Systems Inc., June 2008.

[5] S. McCreary and K. C. Claffy, "Trends in Wide Area IP Traffic Patterns – A view from the Ames Internet Exchange", Cooperative Association for Internet Data Analysis, CAIDA, USA.

[6] "Mixed Packet Size Throughput", The Journal of Internet Test Methodologies, 16–18, Agilent Technologies Inc., Sept. 2004.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. ACM Transactions on Computer Systems, 18(3):263–297, Aug. 2000.

[8] T. Murata. Petri Nets: Properties, analysis and applications. Proceedings of IEEE, 77(4): 541-580, Apr. 1989.

[9] K. Keutzer, S. Malik, A. R. Newton et al. System-level design: Orthogonalization of concerns and platform-based design. IEEE Trans. On CAD, 19(12):1523–1543, Dec. 2000.

[10] J. D. Houle, K. K. Ramakrishnan, S. Rita, Y. Muart and K. Shiv, "The Evolving Internet - Traffic, Engineering, and Roles", A Study of Performance and Economic Models, AT&T Knowledge Ventures, September 2007.

[11] http://www.caida.org/home/about/

[12] http://www.internetworldstats.com/stats.htm

[13] http://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml

[14] http://mawi.wide.ad.jp/mawi/

[15] http://www.physics.csbsju.edu/stats/KS-test.html