Reprogrammable Platforms for High-speed Data Acquisition

Jozsef Ludvig*   James McCarthy†   Stephen Neuendorffer‡   Sonia R. Sachs§

Abstract

Complex embedded systems that do not target mass markets often have design and engineering costs that exceed production costs. One example is the triggering and data acquisition system (DAQ) integrated into high-energy physics experiments. Parametrizable and reprogrammable architectures are natural candidates as platforms for specialized embedded systems like high-speed data acquisition systems. In order to facilitate the design of specialized embedded systems, design strategies and tools are needed that greatly increase the efficiency of the design process. End-user programmability of reprogrammable platforms is essential, because system designers, without training in low-level programming languages, are required to change the base design, compare designs, and generate configuration data for the reprogrammable platforms. This paper presents a methodology for designing and evaluating high-speed data acquisition systems using reprogrammable platforms.

1 Introduction

High energy physics experiments study properties of elementary particles. Accelerator based experiments can currently produce particle energies up to a few TeV while cosmic rays were found up to $10^{21} eV$.

The data aquisition system (DAQ) of a physics experiement captures the data generated by a detector. A DAQ system simulation model includes a number of distinct components:

- An *event generator* is used to simulate properties of the physical events under observation.

- A *detector model* simulates the detector technology and must include its sensitivity and the errors it produces.

- An *analog DAQ front end model* of simulates the analog signal processing and digitization process, including its limitations.

- A *digital DAQ back end model* captures the performance of the low-level data aquisition system, e.g. buffer memory size, dead-time, latency etc..

This framework is valid for the majority of DAQ systems in high energy physics and cleanly defines all interfaces between the physical and technical sub-systems. While detector specific analog front-end circuits are required, generic digital back-ends can serve a variety of experiments. The historically "manual" system integration strategy of these designs can be automated if DAQ is treated like an SoC (System on Chip) design. The key to creating complex designs is a design tool which uses interfaces which are correct by design and transparent to the user.

1.1 The Ptolemy II Modeling Framework

Systems such as the one considered in this paper are *heterogeneous*: they are composed of subsystems with different characteristics which interact in a variety of ways. Many tools have been developed for modeling individual aspects of such system, e.g. data and control flow, analog or digital subsystems. In order to evaluate the complete system, these models need to be composed and the interactions between subsystems can lead to hard-to-analyse, undesirable and unexpected behavior. An in-depth discussion is presented in [3].

In the Ptolemy II framework[5][9], subsystems are hierarchically composed so that the properties of the complete system can be simulated without having to resort to ad-hoc integration of multiple models. These subsystems are able to communicate through both asynchronous, synchronous, buffered, and unbuffered mechanisms.

---

*Lawrence Berkeley National Laboratory, Berkeley, CA 94720. JLudvig@lbl.gov

†Agile Design, Inc., Walnut Creek, CA 94596. jim@agiledesigntech.com

‡EECS Department, U.C. Berkeley, Berkeley, CA 94720. neuendor@eecs.berkeley.edu

§Lawrence Berkeley National Laboratory, Berkeley, CA 94720. SRSachs@lbl.gov

The basic Ptolemy building blocks are called *actors*. Actors atomically execute a task and communicate with other actors through ports.

The model of computation associated with a composite actor is implemented in Ptolemy II as a *domain*. Domains define communication semantics and execution order among actors.

The communicating sequential processes (CSP) domain represents actor processes that communicate by instantaneous rendezvous[4]. The continuous time (CT) domain[10] models ordinary differential equations (ODEs). In the discrete event (DE) domain, actors communicate through events placed on a (continuous) time line. Events have a value and a time stamp, and are processed in chronological order. An actor executed in an SDF model consumes a fixed number of tokens from each input port and produces a fixed number of tokens to each output port. Finite-state machine (FSM) domain[9] entities are states, and inputs to a FSM actor result in state transitions.

1.2 Physics simulation

The DAQ system under investigation was primerily designed for neutrino astrophysics experiments. Since the physics of these experiments is very simple, they are good examples for applying the proposed design strategy. In order to better understand the requirements of the DAQ system, we briefly summarize the physics here.

1.2.1 Neutrino Astronomy Experiments

Neutrino astrophysics experiments are intended to detect high energy neutrinos in the cosmic ray flux. The neutrinos are believed to be generated by cosmic objects like the very large black holes at the center of many galaxies, called Active Galactic Nuclei (AGN's). Neutrinos are very weakly interacting particles and can penetrate galactic dust clouds and even the entire earth without being stopped or loosing energy. They are ideal carriers of information about astrophysical objects which are shielded by large amounts of interstellar matter.

Despite their weak interactions, neutrinos can interact with nucleons and generate high energy muons. These charged particles can be detected more easily. At very high energies ($> 100GeV$) the muon's momentum is approximately half the neutrino energy. At these energies, the neutrino induced muon flux is so small that detectors have to cover an area of one km^2 and a volume on the order of one km^3 or more to detect astrophysical neutrinos.

One of the few known ways to instrument a large detector uses optical detection of Cherenkov radiation.

Cherenkov radiation is electromagnetic radiation generated by charged particles (in this case the muons) moving faster through a polarizeable medium than the local speed of light in the medium. The power spectrum of Cherenkov radiation is proportional to the frequency of the emitted photons and the visible fraction of the Cherenkov spectrum looks blueish. Liquid water and ice are the only Cherenkov media which are available in the required volume. Pure water and ice have very low optical scattering and absorption coefficients for blue and near ultraviolet light and are almost ideal Cherenkov media. Neutrino experiments must be shielded against light and background muons generated in the upper atmosphere by cosmic rays. They are either located in the deep ocean (Dumand, Nestor[11], Antares[1]) or the Antarctic ice shield (IceCube[2]). The very faint Cherenkov radiation is detected by large diameter (8"-12") photo mutiplier tubes (PMTs) enclosed by pressure resistant optical module (OM) spheres.

1.2.2 Cherenkov Radiation

In an optical medium with refraction index n ultra-relativistic charged particles are emitting Cherenkov radiation under a fixed angle of α with $n \, cos(\alpha) = 1$ relative to the momentum vector of the radiating particle. For water and ice this angle is approximately 42 degrees. The particle travels at the tip of the cone shaped Cherenkov wavefront.

A vertical string of detector modules (like the ones used in IceCube) intercepts a Cherenkov cone along a cone section, i.e. a wedge or a hyperbola. This intersection creates a characteristic time profile along the string: the relative timing of photons registered by OMs is a function of the position of the OM and the direction of the particle track. If the distance between the particle track and the string is called r, the projection of the nearest point onto the string coordinate (here chosen to be z) is named z_0, the angle of the track relative to the string (the zenith angle) is called θ, and c denotes the speed of light, the equation for the hit time $t(z)$ becomes:

$$c \, t(z) = (z - z_0)\cos(\theta) + \frac{\sqrt{r_0^2 + (z - z_0)^2 \sin^2(\theta)}}{\tan(\alpha)} \quad (1)$$

While this geometric model is an oversimplification of the physics, it is sufficient for an initial exploration for systems engineering purposes. A Ptolemy model calculating the timing is shown in figure 1.

This model creates the coordinates of eight equidistant modules and the hit times for given particle parameters. These times are used by the top level DE *discrete event simulation* model of the system[1].

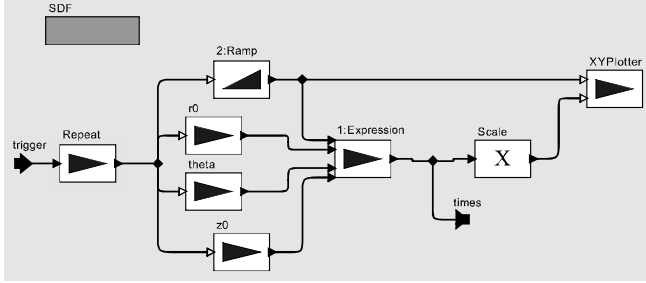[1] The Ptolemy *synchronous data flow* (SDF) Cherenkov model

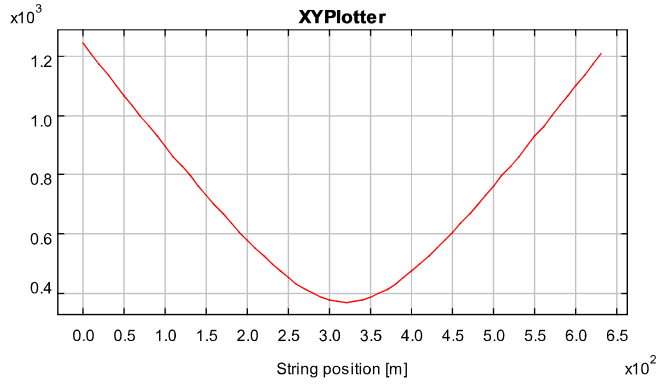Figure 1: Model to Calculate the Timing of the Cherenkov Cone

Figure 2: The Cherenkov Cone Timing[ns] vs. String Position for a Horizontal Track 100 Meters from the String

1.3 Detector simulation

Photomultiplier tubes are stochastic amplifiers: a photon can create a single photoelectron at the surface of the photocathode which is then accelerated towards an electron multiplier chain by a strong electrostatic field where it creates multiple secondary electrons. PMTs with 10-14 secondary electron multiplier stages have an average gain of 10^5 to 10^6. Because very few (3-5) secondary electrons are generated by the primary photoelectron, the amplitude of the final anode current pulse fluctuates strongly. The resulting *pulse height distribution* (PHD) is the most important performance characteristics of a PMT[2].

Figure 4 shows a histogram of a pulse height distribution generated by the model in figure 3. The model

---

itself executes in zero time. The calculated times are converted into timed discrete events by the *timedDelay* actors of the system model shown in figure 9.

[2]A good PHD is nearly Gaussian with width $\sigma^2_{PHD} \approx 1$. Individual photoelectrons are also delayed by a random drift time due to the inhomogenity of the accelerating electrostatic field. The average drift time ($\approx 20ns$) and the time spread ($\approx 2ns$) of the drift time distribution can be modeled with a normal distribution $N(t_{drift}, \sigma^2_{drift})$.
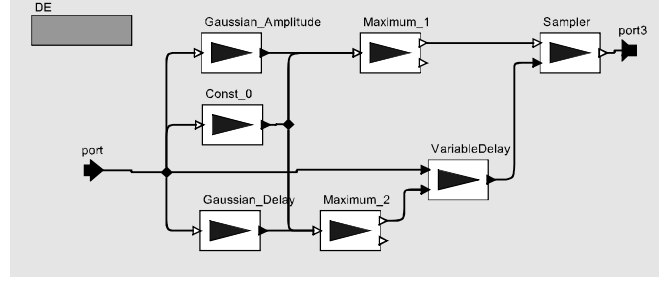

Figure 3: Discrete Event Model of the Stochastic Amplification Properties of a PMT

agrees reasonably well with the measured distribution in the data sheet of the photomultiplier R5912 from Hamamatsu ([6]).
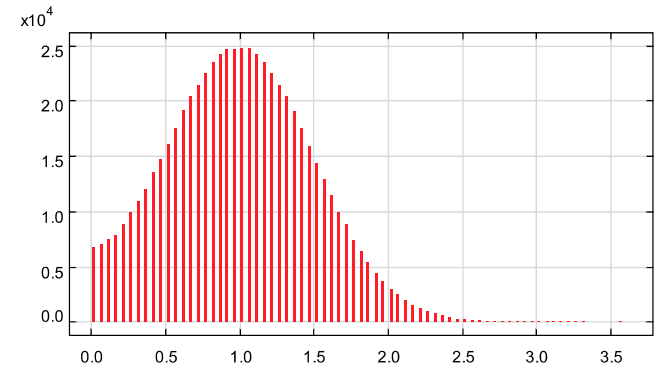

Figure 4: Pulse Height Distribution Calculated by the PMT Model in figure 3(Number of Events vs. Pulse Height)

1.3.1 Time dependent output waveforms

The normalized, time dependent output current waveform of a PMT can be approximated with an electrical model of a bandwith limited Dirac pulse. The PMT waveform generation and the simulation of the analog front-end circuit are very similar and have been integrated into a single CT model.

1.4 Front end

A typical DAQ frontend consists of a pre-amplifier, a signal shaper to limit the bandwidth of the circuit and improve the signal-to-noise ratio of detector signals, a sample/hold (S/H) circuit, and an analog-to-digital converter (ADC)[3].

---

[3]Cherenkov detectors carry most information in the timing of the signal. Timing can be extracted from a PMT pulse by sam-

The PMT waveform and preamplifier model in figure 5 uses a single fourth order low-pass filter with $5ns$ time constant, corresponding to $\approx 60MHz$ pre-amplifier bandwidth. The preamplified and shaped waveforms
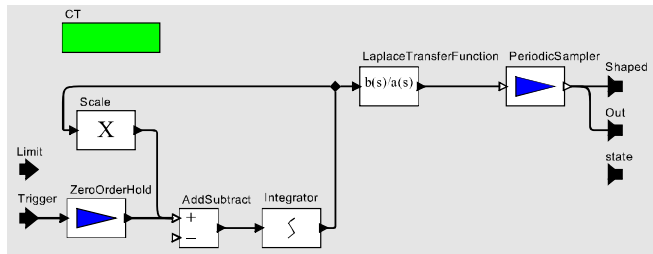

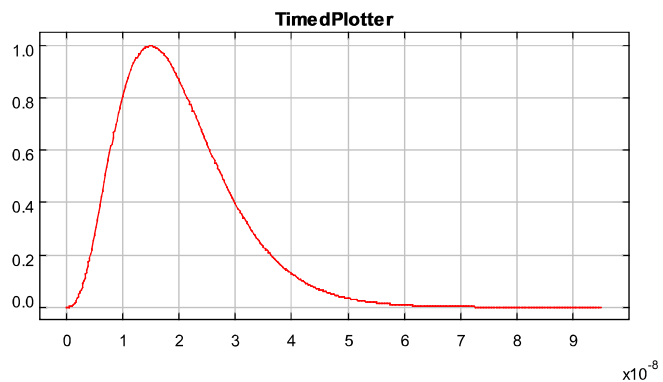Figure 5: Continuous Time Model of the PMT Pulse Shape

Figure 6: PMT Pulse Shape from the model in figure 6(Relative Current vs. Time [ns])

are sampled at 10ns intervals and the samples are discretized to the ADC resolution of 12 bit, i.e. an integer between 0 and 4095.

An important issue for mixed DE/CT simulations is efficiency. While a DE simulation is only evaluated at times at which at least one variable is changing, CT models are evaluated continuously at time intervals which are set by the differential equation solver of the simulator. The use of a periodic sampling actor as a S/H stage model also requires the simulator to evaluate the model every 10ns. This is very inefficient since the average hit rate of a PMT is 1kHz and the PMT pulse is only 100ns long. A free running CT model would create an unnecesary

computational overhead. By embedding it into a *modal model*, a *finite state machine* determines when the CT model execute[4]. This allows low overhead simulation without losing relevant information. This technique is shown in figures 7 and 8. .
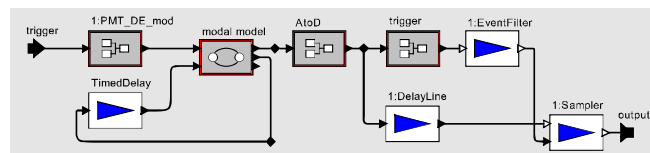

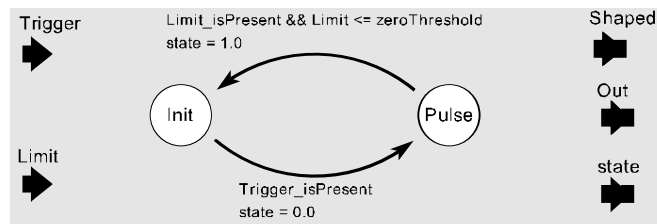Figure 7: Model of the PMT and DAQ Front End

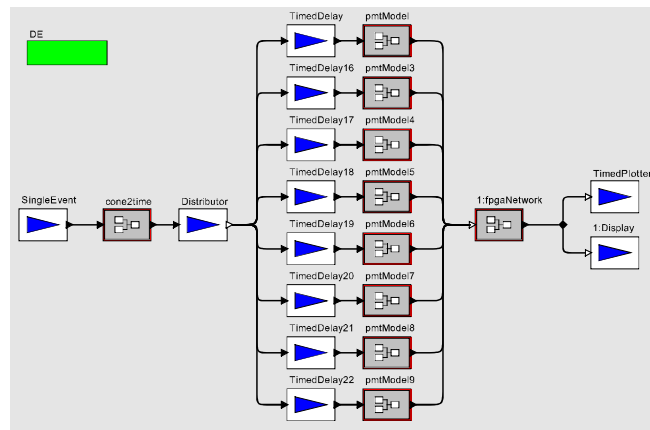Figure 8: Controller of the PMT and DAQ Front End

Figure 9: Complete System Model

2 Back End Model

High-speed data acquisition systems are platforms for acquiring, processing, and moving data around. They may use different digital signal processing algorithms,

---

pling it with high resolution (12 bit) at a high rate (100MSPS) and fitting a parametrized analytical expression for the expected waveform to the sampled data. For PMT pulses three free parameters (amplitude, offset and a pulse time) are used and three samples are sufficient to calculate the pulse parameters. A PMT pulse with 15-20ns width gives 3-5 non-zero samples at 100MSPS sampling rate. The statistical error for the time parameter of the fit is usually less than 20% of the sampling period, or about 2ns.

[4]This embedded modal model contains a state machine with two states, an inactive *Init* state and a *Pulse* state which activates the CT PMT model. The transition from the *Init* to the *Pulse* state is triggered by the presence of a trigger signal, the transition back into the *Init* state occurs as soon as the output of the CT model falls below a given threshold.

data rates, data formats, and network topologies. The problems to be solved by DAQ system designers are: *(i)* how to get the data; *(ii)* how to store it; *(iii)* how to move it efficiently to a destination for processing; *(iv)* how to process the data; *(v)* how to test the correctness of the result; and *(vi)* how to ensure the quality of the process within given constraints.

A reconfigurable platform is an economic way to solve these problems because it can be adapted to a new application without the need to redesign physical hardware.

In this section we describe a reconfigurable hardware platform for DAQ applications and an efficient programming model in the Ptolemy II framework. We also show how an application model is mapped to the platform model. One of the problems of interest is to create a reusable hardware design strategy for a variety of applications that operates efficiently within the platform specific bandwidth, latency, and resource limitations. Given the high level of abstraction provided, user-level programming can include all components of a heterogeneous architecture that entails fabrics of FPGAs, CPUs, memory, network links, algorithms, firmware and software while hiding low level details. Using the Ptolemy II framework, users can model their applications, implement and test algorithms, and assess the performance of the initial implementation.

2.1 The network of FPGAs

The communication of data and control messages between the DAQ front end (analog electronics and ADC), the digital signal processing and the final storage (on a CPU farm) is accomplished with a static network of transmitting and receiving nodes[5].

The messaging uses a custom datagram protocol which can be implemented in a small fraction of available FPGA logic.

2.1.1 Network transmitter node

The transmitter circuit is a synchronous design driven by a clock signal CLK and can be triggered by a transmit enable signal $TxEnable$. After being triggered it forms a data packet containing a destination address

---

[5]For the application in question, the platform has up to 64 input channels with 12 bit ADC running at 100 MHz. The digitized signals from eight analog channels are fed into one FPGA which writes them into 27 Mbit DDR SRAM waveform memories. Digital signal processing algorithms inside the FPGA are used to extract trigger information such as event energy and timing. Processing of the $8 * 12bit * 100MHz = 1.2GByte/s$ data stream is continuous and dead-time-free. Eight FPGAs are connected in a ring topology which is favorable due to the simplicity of the board design and the nearly optimal electrical line length, which helps minimizing noise problems in the analog section.

$DestAddr[27..0]$, a source address $SourceAddr[27..0]$ and the information on its parallel inputs $IN[n - 1..0]$. Flow control codes are used to indicate packet start, stop and idle line states of the protocol.

The following sequence of 8 bit wide output words form a packet which is submitted on one of the network subnets:

$$
\begin{aligned}
&IDLE = 0 \text{ or } PACKET\_STOP = 64 \\
&PACKET\_START = 1 \\
&DestAddr[27..20], [20..14], [13..7], [6..0] \\
&SrcAddr[27..21], [20..14], [13..7], [6..0] \\
&IN[6..0], [13..7], ... \\
&IN[7 * ceil(n/7) - 1..7 * (ceil(n/7) - 1)] \\
&PACKET\_STOP \\
&IDLE \text{ or } PACKET\_START
\end{aligned}
$$

2.1.2 Network receiver node

Receiver nodes listen to packets on their subnet and store message data in a parallel output register when they identify their own address in the $DestAddr$ field of a packet header. In a DAQ application most messages are short (8-16 bytes) and most nodes are specialized to transmit or receicve a single, fixed format message type.

2.1.3 Network router and combiner structures

Figure 10 illustrates the network structures of an FPGA. The datagram network uses independent data paths for

Figure 10: Network structures within a FPGA.

reception and transmission, is collision free and loss-less. A simple high performance packet routing strategy with static routing tables is used to transport messages between nodes of the hardware network. *Router circuits* split nets into subnets. A router reads every packet destination address and reroutes the packet to the internal

network if the packet's destination address matches the address of the local subnet.

Data from multiple subnets are merged by *combiner circuits* into one output stream. Since the transmitter architecture is non-blocking, a combiner uses FIFOs on its inputs to buffer incoming data until the output line becomes available. The finite buffer depth (multiples of 256 byte FPGA block memory size) of the practical implementation limits the peak data rate at the input of the combiner circuit or buffer overflow and data loss will occur. Due to the stochastic nature of the data sources, the network begins to drop packets when the input rate exceeds $\delta * R$, for R, the maximim data rate and δ, a constant < 1. One of the goals of modeling and simulating this network is to find safe limits for the network load, depending on the chosen subnet topology and the stochastic properties of the data. Figure 11 illustrates the structure of the combiner circuit.

Figure 11: Combiner circuit.

2.2 Ptolemy II network model

In the following we model the data flow across the network to collect statistics on the queue lengths used in combiner nodes.

2.2.1 Transmitter Model

The transmitter model is a hierarchical model that combines the Synchronous Data Flow (SDF)[8] and the Finite State Machine (FSM)[7][9] models of computation. The high-level model is shown in figure 12. The top-level model, executing in an SDF domain, takes as inputs event data and a global clock, which in turn are inputs to a finite state machine. The transmit controller has a single state with transitions depending on whether event data, a clock signal (trigger), or both inputs exist at the time(s) execution of the transmit node is scheduled. This controller makes sure that data is only taken
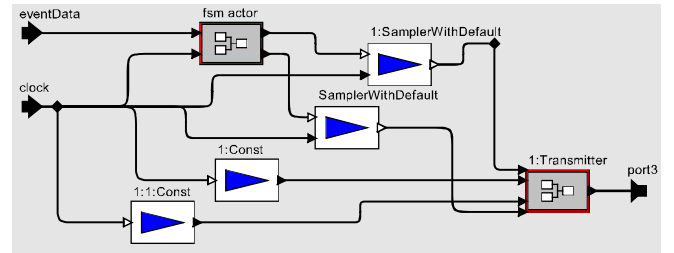

Figure 12: Model of the Transmitter Node

into the transmitter when a valid clock signal is present. The controller generates a data variable and a boolean variable to indicate whether or not the data is valid.

Each controller output drives a *SamplerWithDefault* actor that generates the most recent input token when its trigger port receives a token. The output of the samplers and the (constant) source/destination network addresses are fed into the SDF transmitter model shown in figure 13.
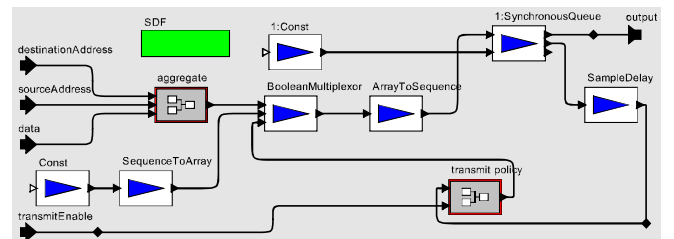

Figure 13: Model of the Transmitter Composite Actor.

The packet data is first combined with the source address, destination address, *PACKET_START*, and *PACKET_END* flags to form a complete packet. This task is performed by the *aggregate* SDF model. The transmission policy, modeled by the SDF *transmit policy* model, takes as input the queue length of the last cycle of the transmitter. If the queue length is zero and if *transmitEnable* input is true, the policy generates a true output. The policy output is used by the *BooleanMultiplexor* to select either the assembled packet or a sequence of *IDLE* flags for transmission. The data packet or the sequence of *IDLE* flags is stored in a synchronous queue.

2.2.2 Combiner Model

The combiner model is also a hierarchical model using the SDF and FSM models of computation. The following figure 14 depicts the top level model of a combiner node. The top level SDF model takes two data inputs which are stored in two queues, modeled by *SynchronousQueue* actors. For each cycle of the model, the output of one queue is selected by a FSM controller. *SampleDelay*
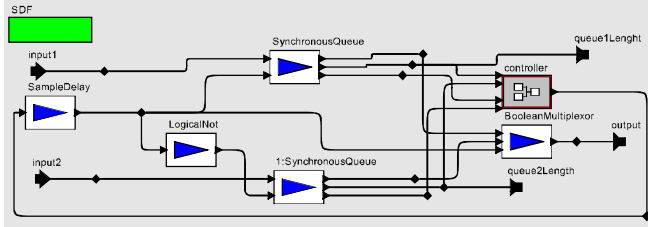
Figure 14: Model of the Combiner node.

actors are used to break dependency cycles in directed loops of SDF models. The *SynchronousQueue* actor also generates a queue length output and allows the controller to look ahead to the next queued value. The combiner controller looks at the queue length of both queues and decides to empty the one that has the most number of packets.

The model of the network of FPGAs is formed by connecting the output ports of one FPGA model to the input port of the next FPGA in the ring. Figure 15 depicts the Ptolemy II model of this network of FPGAs.
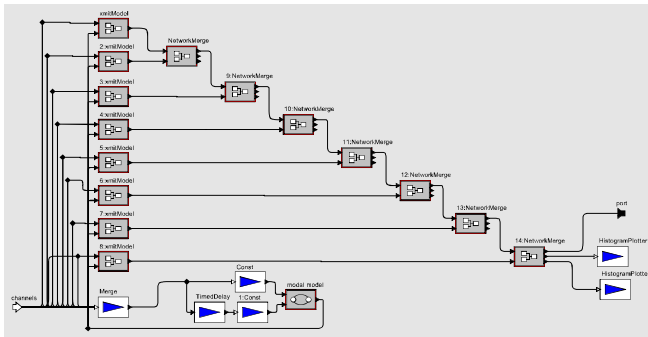

Figure 15: Model of a network of FPGAs

2.2.3   System Performance Discussion

The performance of the system is closely related to the performance of the configurable hardware platform[6]. The main limitation of the non-blocking protocol is the

---

[6]The two implementations of the system under design at Lawrence Berkeley National Laboratory are an eight channel and a 64 channel DAQ board. The eight channel board uses Altera Mercury FPGAs running at a clock frequency of 100MHz. This board can achieve 100MByte/s on any part of the network internal to one FPGA and implements four sub-networks in each of the two chips used for data aquisition. This results in an aggregate data rate of 400MByte/s per chip. The external ring bus topology uses the SerDes circuits and LVDS drivers of the Mercury family to achieve 1GByte/s. With two FPGAs collecting data from 8 channels this is a very well balanced network design. The 64 channel board will be using Altera Apex II chips and further enhance the performance of the 8 channel system.

finite FIFO depth resulting in occasional packet loss. In a high energy physics DAQ environment, most of the events are simply noise and contain no interesting physics information. Packet losses are tolerable if they are limited to a small fraction of the data. However, the system design has to make sure that packet losses do not occur frequently under "normal" circumstances, i.e. if the quality of data generated by the detector is as good or better than specified by the design requirements. The simulation model can help to address these questions by histogramming FIFO congestions. Figure 16 shows the distribution of the number of bytes in the FIFO of the last combiner circuit, which is usually the most congested. The highest observed congestion is well below the physical FIFO length of 256 Bytes, meaning that for the chosen simulation parameters no packets were lost.
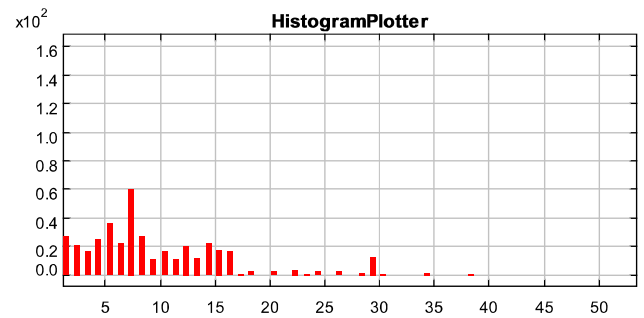

Figure 16: Histogram of Queue1 Length

2.3   Mapping the model to hardware

In the previous sections the modelling and general structure of the system design were discussed. The important result was that a complete and sufficiently realistic system model could be expressed within the Ptolemy II framework. It remains to be shown how this simulation approach lends itself in a natural way to design automation.

2.3.1   Digital logic co-generation

Ptolemy simulation models are ideal for co-generation and design automation of the digital back-end design. The DAQ models using the SDF and FSM domains are synthesizable. SDF actors peforming integer and logic operations can be trivially mapped onto digital circuits. The semantics of the SDF model are identical to the data flow implemented by pipelined digital circuits, therefor the mapping to a circuit essentially becomes an identity operation. A similar relationship exists between a

Ptolemy II FSM model and a digital implementation of a state machine. Apart from naming conventions, each Ptolemy state machine can be mapped directly onto a VHDL description. The Ptolemy group is currently working on generating digital logic from SDF and FSM models.

2.3.2 Network synthesis

The most relevant difference between a "classical" DAQ system and the presented design is the exclusive use of datagram packets on a custom network to represent and transport non-local information. This networked approach maps abstract data structures, i.e. the objects a system designer and users care about, unambigously to an implementation on the hardware level. The presented implementation is simple, correct by design, reasonably efficient in terms of bandwidth and resource usage and can be co-generated easily.

The Java network co-generator maps a list of ports to VHDL code for transmitter and receiver blocks. The release version of this code will add *PacketTransmitter* and *PacketReceiver* actors to the Ptolemy actor library. These actors can be used within Ptolemy to simulate the packet network to co-generate VHDL, Java, and C++ code. Finally, Ptolemy relations are equivalent to a physical netlist of interconnections and can be translated to VHDL mapping files which are connecting multiple network circuits and other co-generated function blocks to a complete DAQ system design.

References

[1] The ANTARES Collaboration. A deep sea telescope for high energy neutrinos. Technical Proposal 99-01, DAPNIA, March 1999.

[2] The IceCube Collaboration. Icecube: a kilometer-scale neutrino observatory. Proposal to the national science foundation, November 1999.

[3] Johan Eker, Jörn W. Janneck, Edward A. Lee, Jie Liu, Xiaojun Liu, Stephen Neuendorffer, Sonia Sachs, and Yuhong Xiong. Taming heterogeneity—the ptolemy approach. *IEEE Proceedings, Special Issue on Modeling and Design of Embedded Software*, scheduled for publication June 2002.

[4] C. Hoare. A theory of CSP. *Communications of the ACM*, 21(8), August 1978.

[5] John Davis II, Christopher Hylands, Bart Kienhuis, Edward A. Lee, Jie Liu, Xiaojun Liu, Lukito Muliadi, Steve Neuendorffer, Jeff Tsay, Brian Vogel, and Yuhong Xiong. Heterogeneous concurrent modeling and design in java. Technical Memorandum UCB/ERL M01/12, Electronics Research Lab, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley California, USA, March 2001.

[6] HAMAMATSU PHOTONICS K.K. Photomultiplier tube r5912. Data sheet, HAMAMATSU PHOTONICS K.K., September 1998.

[7] B. Lee and E. A. Lee. Interaction of finite state machines with concurrency models. *Proc. of Thirty Second Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California*, November 1998.

[8] E. Lee and D. Messerschmitt. Synchronous Data Flow. *Proceedings of the IEEE*, pages 55–64, September 1987.

[9] E.A. Lee, J. Liu, X. Liu, J. McCarthy, S. Neuendorffer, S. Sachs, and Y. Xiong. *Designing Systems with Ptolemy II.* Kluwer, February 2001.

[10] Jie Liu. Continuous time and mixed-signal simulation in Ptolemy II. Memo M98/74, UCB/ERL, EECS UC Berkeley, CA 94720, July 1998.

[11] B. Monteleoni. Nestor a deep sea physics laboratory for the mediterranean. In *Proceeding of the 17th International conference on Neutrino Physics and Astrophysics (NEUTRINO'96)*, Helsinki, Finland, June 1996.