

Tiled DANNA: Dynamic Adaptive Neural Network Array Scaled Across Multiple Chips

A Thesis Presented for the
Master of Science
Degree

The University of Tennessee, Knoxville

Patricia Jean Eckhart

August 2017

© by Patricia Jean Eckhart, 2017
All Rights Reserved.

Abstract

Tiled Dynamic Adaptive Neural Network Array(Tiled DANNA) is a recurrent spiking neural network structure composed of programmable biologically inspired neurons and synapses that scales across multiple FPGA chips. Fire events that occur on and within DANNA initiate spiking behaviors in the programmable elements allowing DANNA to hold memory through the synaptic charge propagation and neuronal charge accumulation. DANNA is a fully digital neuromorphic computing structure based on the NIDA architecture. To support initial prototyping and testing of the Tiled DANNA, multiple Xilinx Virtex 7 690Ts were leveraged. The primary goal of Tiled DANNA is to support scaling of DANNA neural networks beyond the constraints of a single chip or subsystem. To date, the largest physical DANNA implementations have been limited to a single FPGA chip. By synchronizing the neural network activity occurring within DANNA across multiple chips and designing a custom communication interface, DANNA has efficient and continuous scalability. This is accomplished by partitioning DANNA networks across interconnected tiles of DANNA elements in a two dimensional grid and then transmitting fire events over the chip boundaries. This report provides a brief history of neuromorphic computing, the progression of DANNA development, and a discussion over the design and implementation of Tiled DANNA.

Table of Contents

1	Introduction	1
1.1	Neuromorphic Computing	4
1.2	NIDA and DANNA	5
1.3	Related Work and Existing Technologies	7
1.3.1	TrueNorth	7
1.3.2	SpiNNaker	8
1.3.3	Neurogrid	9
1.3.4	Lake Crest and Knight Crest	9
1.3.5	Darwin Neural Processing Unit	10
2	Single DANNA	12
2.1	Host-Array Communication	13
2.2	DANNA Elements	16
2.2.1	Element Basics	16
2.2.2	Element Attributes	18
2.2.3	Element Orientation and Array Inputs and Outputs	18
2.3	Clocking	20
2.4	Resource limitations	21
3	Design of Tiled DANNA	23
3.1	Overview	23
3.2	Communication Design	24
3.2.1	Host to Tiled DANNA Communication Design	25

3.2.2	Interchip Communication Design	27
3.3	Synchronous Clock Generation Design	28
3.4	Design of Slave Control Signals	29
3.5	Boundary Edge Element Connections	29
4	Implementation of Tiled DANNA	31
4.1	Porting DANNA	31
4.1.1	Prototyping Systems	31
4.1.2	Pin Mapping	34
4.2	Communication Implementation	34
4.2.1	Implementation of Host Communication	34
4.2.2	Implementation of Interchip communication	35
4.3	Implementation of Synchronous Clock Generation	36
4.3.1	Synchronous Clock Generation	36
4.4	Implementation of Slave Control Signals	40
4.5	Defining Element Connections and Behavior	40
5	Testing and Validation	43
5.1	Methodology	43
5.2	Tiled DANNA Test Procedure	46
5.2.1	Independent Mode Testing	47
5.2.2	Tiled Mode Testing	47
5.3	Tiled DANNA Test Results	48
5.3.1	Independent Mode Test Results	48
5.3.2	Tiled Mode Test Results	49
6	Progression and Challenges	51
6.1	ProFPGA	51
6.2	Clock Generation	52
6.3	Clocking Constraints	53
6.4	Clock Buffers	54

6.5	Tiled DANNA Mode	55
7	Future Work	57
7.1	Scaling Tile-to-Tile Interface	57
7.2	Increase Boundary Edge Element Connections	58
7.3	2x2 Matrix of DANNA Tiles	59
8	Conclusion	60
	Bibliography	61
	Vita	65

List of Tables

1.1	Darwin Neural Processing Unit Configurations [32]	11
2.1	DANNA Commands and Opcodes	14
2.2	DANNA COMPRESS NULL and LOAD Opcode and Instructions	15
2.3	DANNA STEP and FIRE Opcode and Instructions	15
2.4	DANNA Opcodes with No Instructions	15
2.5	DANNA Response Packet	16
2.6	Programmable Element Attributes	18
2.7	Element Orientation Scheme	20
2.8	Clocks Necessary to Run a Single DANNA	21
4.1	Clocks Generated on the Master Tile of DANNA	38
7.1	Pin Requirements Under 36, 62, and 72 Bit Bandwidths	58

List of Figures

1.1	History of Neural Networks	3
1.2	First Era of Computing	3
1.3	Second Era of Computing	3
1.4	NIDA Networks Performing Crossover[30]	6
1.5	IBM TrueNorth 16 Chip Board Developed for DARPA Synapse Project[11] .	8
1.6	SpiNNaker Neuromorphic Computing Platform[11]	8
1.7	Neurogrid Neuromorphic Computing Platform[11]	9
1.8	Intel Lake Crest Tile Block Diagram[23]	10
1.9	Darwin Neural Processing Unit Platform[26]	10
2.1	High Level DANNA Communciation Diagram	13
2.2	Operational Logic Flow of Element Behavior	17
2.3	Black Box Diagram of Element	17
2.4	Nearest Neighbor Element Connections[5]	19
2.5	Clocking Scheme of a Single DANNA	21
2.6	Virtex 7 690T 45x45 Element Array Resource Usage	22
3.1	Tiled DANNA Structure	24
3.2	Host to Tiled DANNA Communication Interface	25
3.3	Tiled DANNA Tile-to-Tile Communication Interface	26
3.4	Host to Tiled DANNA Operating in Tiled DANNA Mode	27
3.5	Host to Tiled DANNA Operating in Parallel Mode	27
3.6	Tiled DANNA Boundary Edge Element Connections	28
3.7	Tiled DANNA Boundary Edge Element Nearest Neighbor Connections . . .	30

4.1	ProFPGA Quadmotherboard Prototyping System	32
4.2	I/O Signal Accessibility on ProFPGA 690T Module	33
4.3	Tiled DANNA Prototype	34
4.4	Fire Event Multiplexing and Demultiplexing Across the Tile Boundary . . .	35
4.5	Timing Diagram of Fire Event Data Transfer	36
4.6	Clocks Generated the Master Tile	38
4.7	Timing Diagram for 1 MHz Network Pulse Generation	38
4.8	Clocks Signal Routing of Tiled DANNA	39
4.9	TCL Command Used to Define Slave Clocks	40
4.10	Timing Diagram for Enabling Element Operation	40
5.1	Straight Passthrough Network	45
5.2	Snake Passthrough Network	45
5.3	Timing Grid Network	45
6.1	Initial Synchronous Clock Design	53

Chapter 1

Introduction

Traditional computing methods are no longer sufficient in our advancing technological society. The need for low power devices with the ability to perform complex calculations in nearly real time has become a reality. The traditional approach to computing uses the Von Neumann architecture which stores programs that contain data and instructions in the same space. The instructions are fetched and decoded from the memory and then operations on data specified in the instructions take place. Under this architecture, instruction fetch, decode, data retrieval, and operation cannot be done at the same time. This leads to what is known as the von Neumann bottleneck. No matter how fast you make a processor, the amount of time it takes to fetch and decode an instruction and then retrieve data to perform the operations will limit any computer. To circumvent this bottleneck, multiprocessor systems and graphical processing units have been employed along with implementations of caching, out-of-order execution and prefetching. However, this approach to parallelism comes at a cost in power, both physically and computationally. The need for a new architecture to handle the processing of massive amounts of data and the computational power to do complex operations, while consuming relatively low-power usage has arrived. Technical communities have the knowledge and ability to develop real time applications that can collect large amounts of data in short amounts of time, but there are no hardware platforms that can easily handle these demands. As Venkateswaran et al. have stated[36], the memory access latency and bandwidth constraints in the traditional Von Neumann architecture have demanded a new platform. "The parallel control flow overhead

and resource utilization contention has exacerbated the impact of the bottleneck”. An approach to computing known as neuromorphic, or cognitive, computing has made a couple of attempts at solving this problem throughout the history of computing. These neuromorphic architectures are notable for being highly connected and parallel, requiring low-power, and collocating memory and processing [31]. After every major advancement in computing, neuromorphic computing has also made significant advancements. Figure 1.1 shows the developmental progression of neuromorphic computing over the last 80 years and figures 1.2 and 1.3 depict the developments that occurred in computational devices over a similar time period. Neuromorphic computing involves integrating the software implementation of neural networks onto hardware devices. The remainder of this chapter provides a brief history into neuromorphic computing followed by a description of the progression that has lead to the realization of a neuromorphic computing architecture known as a Tiled DANNA which is fully discussed in the entirety of this thesis.

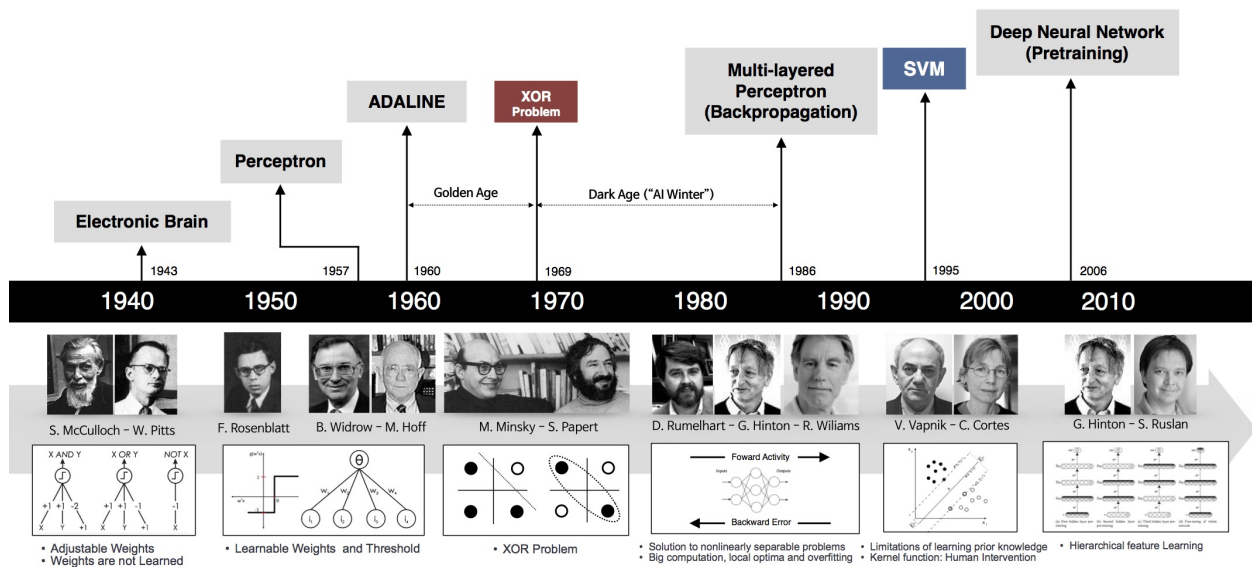


Figure 1.1: History of Neural Networks

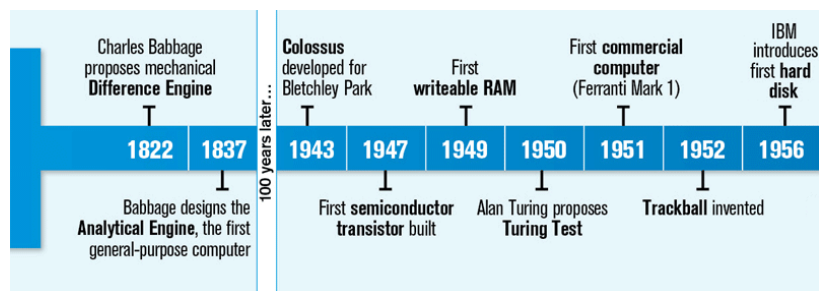


Figure 1.2: First Era of Computing

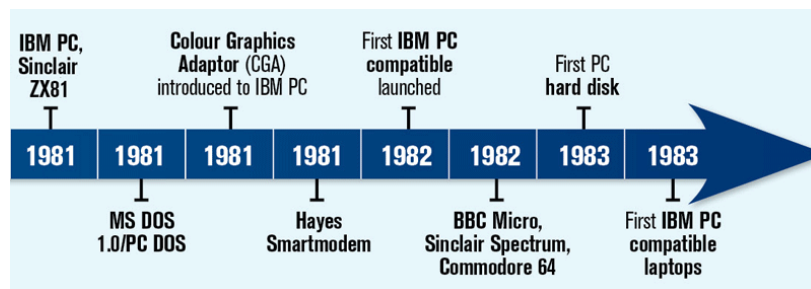


Figure 1.3: Second Era of Computing

1.1 Neuromorphic Computing

A surge in the neuroscience field in the 1940's sparked many studies into how the human computational machine, the brain, works. These studies led to biological breakthroughs and inspired scientists Warren McCulloch and Walter Pitts to design the first artificial neural network in hardware [24]. It came to fruition in 1943 through a design of electric circuits that replicated what the behavior of neurons was thought to be at that time. This collision of neuroscience and computational hardware, along with the advent of the transistor and integrated circuits, gave birth to the field of neuromorphic computing which experienced its first major boom in the 1960's.

During the first boom, the perceptron was invented [28], the nearest neighbor algorithm was designed, and automatic differentiation was discovered, later implemented in backpropagation [29]. The momentum in the field faded as the limitations of neural networks became evident and advancements were made in microprocessors and integrated circuit technology. Neural networks during this era were constructed of perceptrons which had a limiting factor of only being able to classify linearly separable data sets. Minsky and Papert exploited this limitation in their book *Perceptron*[22], Hinton, and Williams solved the limiting problem found in the perceptron by modifying the mathematical model used in the perceptron to use automatic differentiation [29]. Multilayer networks were implemented and led John Hopfield to develop recurrent neural networks. Which are fully connected graphs that can serve as content addressable memory systems implementing automatic differentiation [15]. This second boom in neuromorphic computing brought life and momentum back into the field. Neuromorphic computing has since had steady progression and development. With the development of software systems like Watson [17] and AlphaGO [3], along with the development of sophisticated evolutionary algorithms[30], deep learning [20], and hardware devices like TrueNorth [16] and SpiNNaker [18], the neuromorphic industry seems to be experiencing a third significant surge in advancement that may inevitably replace the Von Neumann architecture, or at least provide another computing paradigm for emerging complex data applications.

1.2 NIDA and DANNA

Artificial Neural Networks (ANN) and Spiking Neural Networks (SNN) are connected compositions of nodes that are used as computational models that mimic the behavior of neurons in the human brain. ANN and SNNs are powerful and informative structures that have been implemented to produce valuable information in numerous fields of research and industries. ANNs are static in nature and are trained using machine learning techniques. SNNs are event driven through input pulses. While ANNs and SNNs have proven to be viable computing constructs for tasks such as pattern recognition, image processing, fault identification, real-time analysis, and deep learning applications [8], the high connectivity and required computational resources make them difficult to implement without a significant amount of processing and computational power.

Schuman introduces a neuroscience-inspired dynamic architecture (NIDA) that embeds a spiking neural network inside a geometric space. NIDA networks are trained using evolutionary optimization, and have been proven effective on classification, control, and anomaly detection tasks. NIDA is unique in that it maps the problem into a three-dimensional space. Doing this reduces the problem complexity because the network only has to solve the problem and not figure out the geometric relationship between the inputs [30]. NIDA networks are directed networks composed of only limited parameter neurons and synapses. A unique evolutionary optimization technique is applied to NIDA networks to produce networks trained for specific tasks. During evolutionary optimization networks experience mutation and crossover. Figure 1.4 depicts the networks that result from crossover. NIDA networks are recurrent which allows them to be adaptive during the training process. DANNA translated nicely into a digital representation due to the simplicity and structure of NIDA. These digital elements are designed so that they can be programmed to be either a neuron or a synapse. These elements are then replicated across a field programmable gate array (FPGA) creating Dynamic Adaptive Neural Network Arrays (DANNA) [4].

As previously stated, Dynamic Adaptive Neural Network Arrays (DANNA) incorporate a design that consists of programmable element which can be programmed to be a neuron or a synapse, and a two dimensional structure and communication backbone to allow neural

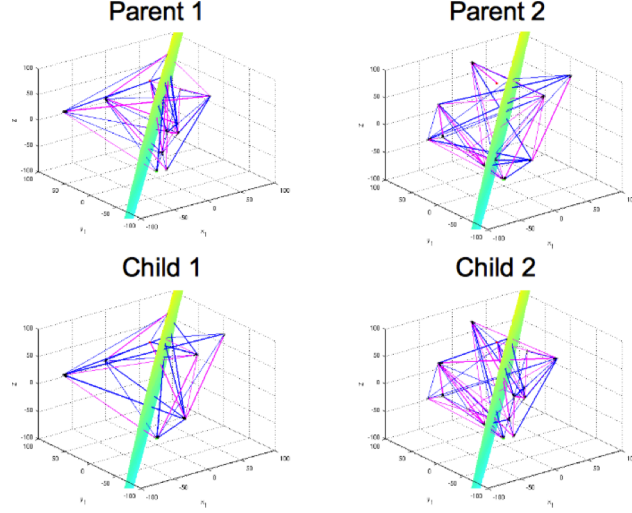


Figure 1.4: NIDA Networks Performing Crossover[30]

networks to be created and dynamically modified [8]. Each element of DANNA has the potential to be connected to 16 of its nearest neighbors. Elements that reside along the edges of the array are used to accept inputs to the array and produce outputs from the array. Activity in a DANNA network is governed by two clocks: a global network clock and an input select clock. The global network clock governs the overall activity of the network. The input select clock runs 16 times faster than the global network clock, and it governs the timing of the sampling of inputs driven from neighbor elements [9]. As the synapse connected to a neuron fires, the neuron accumulates charge. When the accumulated charge reaches or exceeds a programmed threshold, the neuron fires. This firing event triggers activity in the corresponding connected output synapses. A neuron is restricted from firing more than once during a network clock cycle. Synapses have a weight, delay, and a refractory period, all three of which are programmable. A synapse weight can be updated through processes called long-term potentiation and long-term depression[4]. These are processes present in biological neural systems. DANNAs are currently implemented on field programmable gate arrays (FPGAs). All DANNAs are identical in physical composition, and the only specifications necessary to configure DANNA are the size of the array and the location of the array inputs and outputs. Once a DANNA structure is built on to the FPGA, the network topology and parameters can be programmed allowing networks to be loaded and tested [9].

Implementations of DANNA on FPGAs have achieved very high utilization due to high parallelism and rigid control over the elements [7]. Due to resource limitations DANNA has reached its largest potential on a single tile. The following chapters of this report discuss the design and implementation details of Tiled DANNA, a dynamic adaptive neural network array which spans across multiple interconnected tiles by partitioning a DANNA network into sub-networks and tiled across a matrix of tiles. A uniquely designed communication interface allows the firing events between each partition to be transmitted with minimal disturbance to the original behavior of DANNA. The multiple DANNA partitions operate synchronously and act as one DANNA. Tiled DANNA allows the architecture of DANNA to extend itself over a single tile to multiple interconnected tiles. Tiled DANNA can also be configured to operate as multiple DANNAs running in parallel.

1.3 Related Work and Existing Technologies

Neuromorphic computing is steadily advancing due to initiatives like the European Flagship’s Human Brain Project[35] and DARPA’s SyNAPSE project[6] backing the research. The need for low power devices with high computational power is demanded by our data driven and tech hungry society, and the traditional computing method with shared memory for data and instructions is limited. Taking cues from biology, designers have developed several successful neuromorphic systems that are proving to be excellent successors to the von Neumann architecture. The following sections very briefly describe a few of the front runners in the field.

1.3.1 TrueNorth

TrueNorth[16] is comprised of one million programmable neurons and 256 million programmable synapses and is being developed by IBM. The tile is a CMOS integrated circuit consisting of 4,096 neurosynaptic cores which encompass memory, computations, and communication. It was introduced by IBM in 2014. This architecture avoids the bottleneck found in traditional von Neumann computing [16]. The tiles are distributed in two dimensions and communicate over an interchip communication interface. The neuromorphic

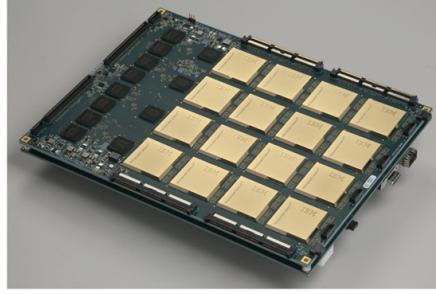


Figure 1.5: IBM TrueNorth 16 Chip Board Developed for DARPA Synapse Project[11]

architecture of TrueNorth is composed of a 256 input lines and 256 outputs neurons connected through directed programmable synaptic connections [21]. TrueNorth has proven itself to be exceptionally proficient at image and speech recognition. TrueNorth has the ability to classify images between 1,200 and 2,600 frames per second while only using 25 to 275 milliwatts of power [10]. This has the potential of enabling low power devices the ability to classify images in real-time from numerous standard video feeds at one time. This far surpasses the leading comparable hardware device, NVIDIA's Tesla P4 GPU.

1.3.2 SpiNNaker

SpiNNaker (Spiking Neural Network Architecture)[18] consists of an array of one million ARM9 cores that use a packet communication infrastructure transmitted through a custom interconnect fabric. Packet transmission is handled solely by hardware allowing for a bandwidth of over 5 billion packets/s [12]. SpiNNaker tiles can be connected together to form massively parallel computing systems based on spiking neural networks. The overall design involves 1,000,000 cores each emulating 1,000 neurons. SpiNNaker is one of the components of the Human Brain Project's neuromorphic computing platform [12].

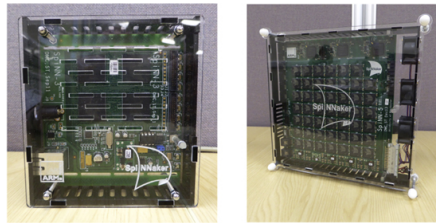


Figure 1.6: SpiNNaker Neuromorphic Computing Platform[11]

1.3.3 Neurogrid

Stanford University is working on developing the Neurogrid [2] which simulates one million neurons connected by six billion synapses through a system composed of sixteen Neurocores. A Neurocore is primarily analog with the exception of the digital logic that is used to implement the axonal arbor of the neuron. This allows the Neurogrid to only require three watts of power. This highly parallel, interconnected architecture attempts to function like the human brain. The fundamental component of Neurogrid is a silicon neuron.

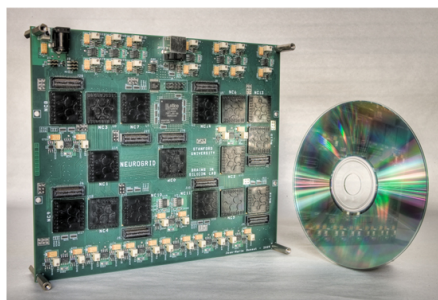


Figure 1.7: Neurogrid Neuromorphic Computing Platform[11]

1.3.4 Lake Crest and Knight Crest

Intel has recently acquired the artificial intelligence start up company Nervana. Intel absorbed the Nervana Engine in the acquisition and released it as Lake Crest. This neuromorphic tile combines a new memory technology called High Bandwidth Memory and a customized numerical format that maximizes the precision that can be stored in 16 bits. The high capacity memory is achieved by die-stacking. A single tile is composed of 8 1GB memory dies making it capable of storing 8GB of data. Six bidirectional high bandwidth links are available to allow tiles to be interconnected in a seamless fashion. “The Nervana Engine featuring high-bandwidth memory, unprecedented compute density, isolated data and computation pipelines, and built-in networking will enable deep learning at a scale never before seen in the industry” [19].

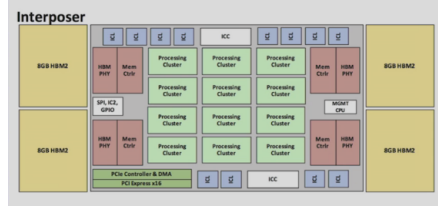


Figure 1.8: Intel Lake Crest Tile Block Diagram[23]

1.3.5 Darwin Neural Processing Unit

A neuromorphic hardware coprocessor named Darwin Neural Processing Unit (NPU) has been developed as a collaboration between research groups from Zhejiang University and Hangzhou Dianzi University in Hangzhou, China. The coprocessor is a highly configurable and is a digital representation of a spiking neural network based on the Leaky Integrate and Fire (LIF) SNN model [32]. The physical neuron units are time-multiplexed to reduce the computational resource requirements and the memory subsystems are reconfigurable. It was prototyped on an FPGA and fabricated as an ASIC. The NPU supports 8 physical neurons which are time multiplexed making the total number of logical neurons configurable. The degree of the time multiplexing can be modified allowing for four different configurations which are shown in table 1.1. The DPU is aimed to be incorporated into low power embedded systems.

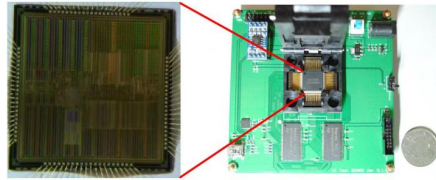


Figure 1.9: Darwin Neural Processing Unit Platform[26]

Table 1.1: Darwin Neural Processing Unit Configurations[32]

Configuration	Max # of Neurons	# of Synaptic Delays	Max # Synapses
1	2048	15	4M
2	4096	7	16.7M
3	8192	3	67M
4	32,768	1	1000M

Chapter 2

Single DANNA

Dynamic Adaptive Neural Network Arrays are neuromorphic systems with a new and distinct architecture that rivals the traditional von Neumann architecture. DANNAs mimic specific behaviors of the human brain to implement programmable elements that form computational structures capable of successfully performing complex control and classification applications. DANNAs consist of adaptive elements with a programmable structure which allows for the creation of a spiking neural network represented using digital logic. They are designed to solve problems using evolutionary optimization [7]. The unique characteristics of DANNA over other neuromorphic systems include a basic neuron model working in conjunction with a high functioning synapse model and dynamic adaptability with configureability of programmable of interchangeable elements [4]. DANNA is currently implemented on the Xilinx XC7VX690T FPGA with 693,120 logic cells, and can support arrays as large as 45-by-45 elements. This is the largest possible DANNA using the Virtex 7 690T FPGA due to resource limitations on the FPGA. By tiling multiple Virtex 7 690Ts configured with DANNA together and enabling them to pass fire events across the tile boundary to elements located on a different tile via a uniquely designed tile interface, DANNA can easily span across multiple tiles allowing the array size to be unconstrained by tile resources. The initial prototyping design and implementation of Tiled DANNA is discussed in Chapters 3 and 4.

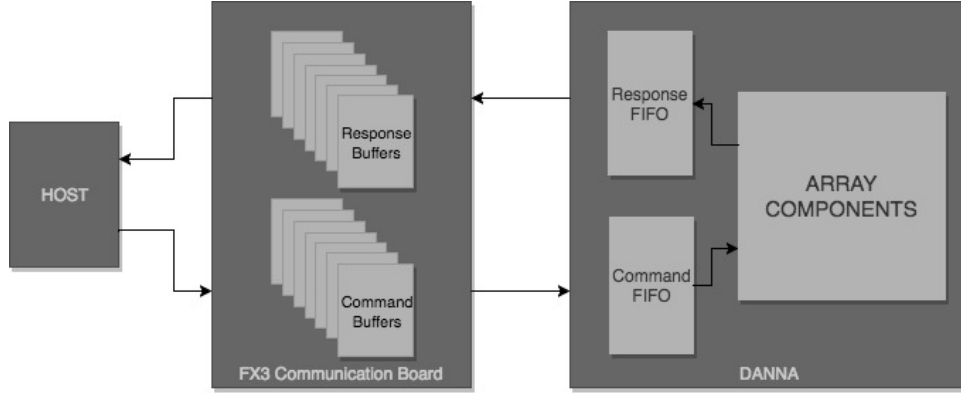


Figure 2.1: High Level DANNA Communication Diagram

2.1 Host-Array Communication

DANNA communicates asynchronously with a host via a Cypress FX3 Superspeed peripheral controller board (FX3). The host interfaces with the FX3 over USB 3.0 or USB 2.0. The communication between the host and DANNA consists of 36 byte command packets and 64 byte response packets. The packets are buffered on the FX3 and a finite state machine (FSM) controlled by DANNA handles the synchronization and transfers between the FPGA and FX3. A 32 bit GPIF interface exists between DANNA and the FX3. The transfers occur at a 100 MHz rate over a 32 bit interface.

Communication between the host and DANNA begins with the host and FX3 establishing two USB endpoints to send and receive bulk transfers. Bulk transfers from the host to the FX3 consist of up to nine 1 KB bursts. This allows up to 256 command packets to be sent to the FX3 without interruption in one transfer. The command packets are held in one of 6 direct memory access(DMA) buffers on the FX3. Each command DMA buffer is 9KB. Two DMA channels are established between the command side of the FX3 and the general purpose interface(GPIF) on the FPGA to ensure no command packets are dropped while the DMA buffer is in transit. The commands are pulled from the DMA buffers over the GPIF interface by the FSM in DANNA at a 100 MHz rate. DANNA takes the commands from the FX3 and feeds them to a 2KB command FIFO. At a 16 MHz rate, commands are pulled one at a time from the command FIFO and decoded by the programming interface. When DANNA fires or is instructed to provide a snap shot of the elements, the programming interface encodes the

output fire weights, the capture/shift data, status flags, and configuration id into a response packet. The response packets are fed to a 2KB response FIFO one at a time with a 16 MHz rate. DANNA then sends the response packets to the FX3 over the GPIF interface at a 100 MHz rate. The GPIF transfers the response packets to the FX3 over one DMA channel to one of six 16KB DMA buffers. Each response DMA buffer holds 16KB. The host receives the response packets from the FX3 in bulk transfers consisting of 16 1KB bursts. Thus up to 256 responses can be received by the host in one USB transfer.

Communication between the host and DANNA consists of two types of packets: command and response. DANNA currently accepts nine different commands which are listed in table 2.1. All command packets are composed of a one byte opcode and 35 bytes of instructions. Each of the commands, opcodes, and corresponding payloads are shown in tables 2.2, 2.3, and 2.4. All the response packets consist of an eight byte timestamp and 48 bytes of fire weights and array status information. The 64 byte response packet fields are shown in table 2.5

Table 2.1: DANNA Commands and Opcodes

COMMAND	OPCODE
NULL	0000 0000
LOAD	0000 0001
HALT	0000 0010
RUN	0000 0100
STEP	0000 1000
FIRE	0001 0000
RESET	0010 0000
CAPTURE	0100 0000
SHIFT	1000 0000

Table 2.2: DANNA COMPRESS NULL and LOAD Opcode and Instructions

COMPRESS NULL	LOAD
00000000	00000001
8 byte compress value	4 Byte Element Address
27 bytes unused	4 bits unused — 4 bit refractory period
	3 bits unused — 4 bit output port — 1 bit type
	1 byte thresholdweight
	1 byte distance
	2 byte to bitmask input ports
	25 bytes unused

Table 2.3: DANNA STEP and FIRE Opcode and Instructions

STEP	FIRE
00001000	00100000
8 byte step value	32 bytes of fire weights
26 bytes unused	3 unused bytes

Table 2.4: DANNA Opcodes with No Instructions

HALT	RUN	RESET	CAPTURE	SHIFT
00000010	00000100	0010000	010000000	01000001
35 bytes unused	35 bytes unused	35 bytes unused	35 bytes unused	35 bytes unused

Table 2.5: DANNA Response Packet

Byte Number	Bit Fields
0 - 7	Timestamp
8 - 39	Fire weights
40 - 43	Unused
44 - 59	Element snapshot
60	Unused
61	Status flags
62 - 63	Configuration ID

2.2 DANNA Elements

2.2.1 Element Basics

There are two types of programmable elements in the DANNA architecture. Each element of DANNA can be programmed to be either a synapse or neuron. The way the logic blocks and signals are used by the two element types make them distinct. An element is programmed to be either a neuron or a synapse when the DANNA array is configured by load commands from the host. The fields of the load command are shown in table 2.2. Figure 2.2 shows the major components and the logical flow of the element design. Each element has 4 input clock ports, 16 input ports (one for each of its connected neighbors), and 9 control and operational input ports [8]. An element has three output ports: one for its fire output weight; another for a signal signifying the element fired; and a third for a signal containing element status information. Figure 2.3 provides a black box diagram of the element input and output ports.

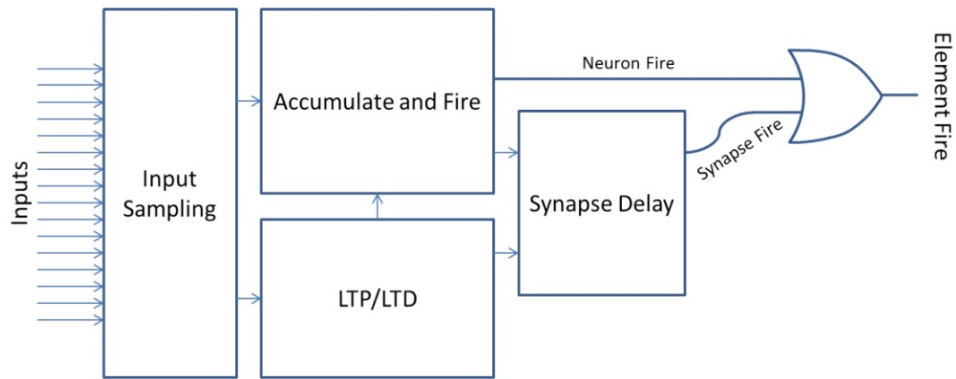


Figure 2.2: Operational Logic Flow of Element Behavior

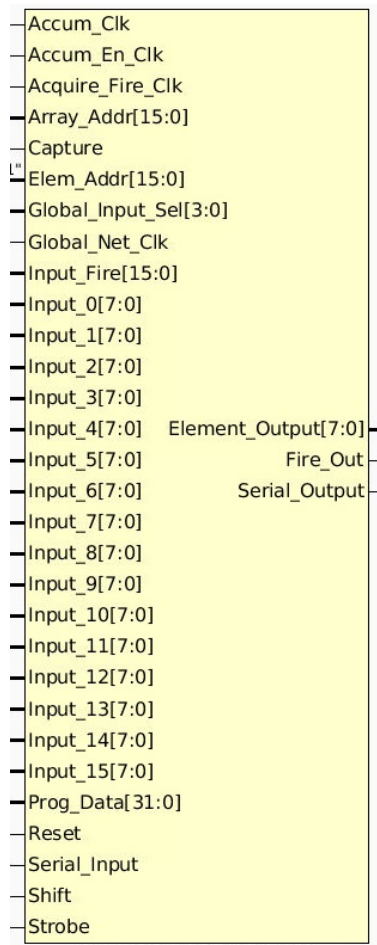


Figure 2.3: Black Box Diagram of Element

Table 2.6: Programmable Element Attributes

Attribute	Size	Type
Programmable Threshold	1B	Neuron
Programmable Distance	1B	Synapse
Programmable Weight	1B	Synapse
Refractory	4 bits	Synapse

2.2.2 Element Attributes

Neurons have a programmable threshold. When a neuron’s input synapse fires, the neuron’s current accumulated charge is increased by the weight of the fire it receives. Once a neuron’s charge reaches its threshold, it fires and the accumulator resets. Synapses have a programmable weight. This weight represents the amount of charge it will add to its connected neuron when it fires. Synapses also have a programmable distancedelay function. The synapse distancedelay represents the neurological charge propagation [5]. The synapse’s weight is output after the number of global cycles equivalent to the synapse’s distance have expired. Synapses and neurons both have a refractory period. An element enters the refractory period after it fires. While in the refractory period the element cannot fire. The refractory period for a neuron is set to one global network cycle. The refractory period for a synapse is programmable to one to sixteen global network cycles. A neuron may still accumulate charge while it is in the refractory period. Synapses can perform long term potentiation and long term depression [8]. When a synapse fires, if it causes its connected neuron to fire, it will potentiate increasing its weight by one. Depression occurs when a synapse fires at the same time its connected neuron fires, reducing its weight by one. Neurons do not potentiate or depress. Synapses only respond to firing events from one connected neuron. Neurons respond to fires from all of its connected elements. Both neurons and synapses make use of the accumulator. Neurons use it to accumulate charge and synapses use it to potentiate or depress its weight. Table 2.6 lists all the element attributes.

2.2.3 Element Orientation and Array Inputs and Outputs

DANNA is a $i \times j$ grid of elements with i rows, j columns, and ij elements. DANNA fire commands are received by a predetermined set of elements in the first column. This column

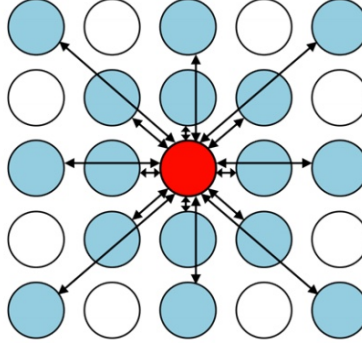


Figure 2.4: Nearest Neighbor Element Connections[5]

is known as the Array Input Edge. DANNA output responses are sent from a select set of elements in the last column of the array. This column is known as the Array Output Edge. Each element is physically connected to 16 other elements, known as neighbors. An element has one neighbor located at each of the immediate cardinal and inter cardinal directions as well as one neighbor located next to each of the eight immediate elements. This creates a star configuration with an inner ring of eight neighbors and an outer ring of eight neighbors see figure 2.4. The elements are numbered in sequential order from 0 to ij and arranged such that the element numbers wrap around to the beginning of the next row. We have also used a rowcolumn based array addressing scheme to enable compatibility when scaling to larger arrays.

An element has 16 input ports which accept the one byte fire values it may receive from a connected neighbor. Input ports 0 through 7 are mapped to the inner ring of neighbors and input ports 8 through 15 are mapped to the outer ring of neighbors. The input ports of connected elements reciprocate, so input port 0 on one element is connected to input port 0 of its neighbor. This connection scheme causes the orientation of the elements to differ from one another. The input port numbering scheme is such that input ports 0, 4, 8, and 12 are either oriented to the North or South, input ports 2, 6, 10, and 14 are oriented to the East or West, and input ports 1, 3, 5, 7, 9, 11, 13, and 15 are oriented to the North-North-East, South-South-East, South-South-West, or North-North-West. Thus, there are four element orientations for the inner ring and four element orientations for the outer ring.

The orientation of each element can be found by determining the parity of the row and column in which the element resides. Table 2.7 provides the relevant element orientation

Table 2.7: Element Orientation Scheme

Orientation	Port Order	Element Location
0	Clockwise	even row, even column
1	Counter Clockwise	odd row, even column
2	Clockwise with 180° rotation	even row, odd column
3	Counterclockwise with 189° rotation	odd row, odd column

information. Under this scheme, all the elements on the array input edge (column 0) will always be of orientation type 0 or 2. This makes input port 6 always facing to the West and always the port to accept inputs into the array.

2.3 Clocking

DANNA requires two communication clocks, three free running array clocks, and four gated array clocks to operate, refer to table 2.8. Three of the gated clocks are enabled when the array receives a run command and the fourth gated clock is enabled when the array receives a run command and it is the beginning of a network cycle. These four gated clocks are used for element control and operation. The gated 32 MHz clock is the Accumulator Clock which supports two accumulate operations during each input port sample (charge update and/or charge reset in a neuron or weight potentiation/depression in a synapse). The gated 16 MHz clock is referred to as the Acquire Fire Clock and it is used to control the firing of the neuron once its charge matches or exceeds its threshold. The Accumulator Enable Clock is a gated 16 MHz phase shifted clock and is used in conjunction with the Accumulator Clock to determine what operation should occur (charge update or charge reset in a neuron, or weight potentiation/depression in a synapse). The array cycle clock and firing clock for a synapse is the Global Network Clock and is a gated 1 MHz pulse. The two communication clocks are necessary for communication with the host. One communication clock is used to transfer data between the host and DANNA. The second communication clock is used in conjunction with an element clock to pull and push data from the programming interface. Figure 2.5 shows which array components use which clocks.

A 100 MHz on-board oscillator is used by a LogiCORE IP Clocking Wizard to generate the two 100 MHz communication clocks. A second LogiCORE IP Clocking Wizard takes the

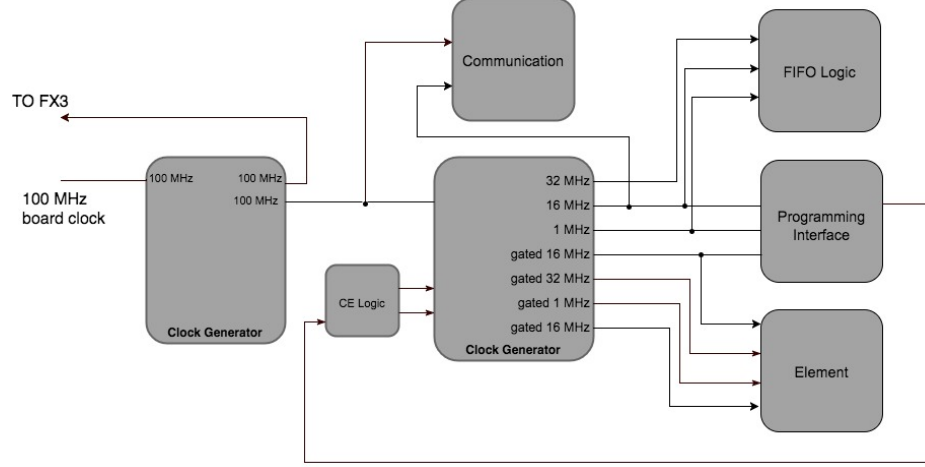


Figure 2.5: Clocking Scheme of a Single DANNA

Table 2.8: Clocks Necessary to Run a Single DANNA

Frequency	Gated	Components
16 MHz	no	Host comm, FIFO logic, programming interface
32 MHz	no	FIFO logic
1 MHz	no	FIFO logic and programming interface
16 MHz	yes	Programming interface and element
32 MHz	yes	Element
1 MHz	yes	Element
16 MHz 90° phase shifted	yes	Element
100 MHz	no	Host communication
100 MHz 180° phase shifted	no	Host communication

generated 100 MHz clock as an input and creates the 7 array clocks. An enable signal is set after every 16 counts of the free running 16 MHz clock. This signal enables a 16 MHz clock for one cycle. This creates a 1 MHz pulse used to signify the beginning of a network cycle. A second signal set by a run command is used to enable four of the element clocks.

2.4 Resource limitations

DANNA is currently being implemented on a field programmable gate array(FPGA). FPGAs allow for easy prototyping of digital systems, but FPGAs put constraints on network size and element complexity. The FPGA technology is based on a lookup tables(LUT) and each FPGA model has a limited number of LUTs. The combinational logic is digitally realized

Resource	Utilization	Available	Utilization %
FF	280439	866400	32.37
LUT	369782	433200	85.36
Memory LUT	20250	174200	11.62
I/O	51	850	6.00
BRAM	1	1470	0.07
BUFG	14	32	43.75
MMCM	2	20	10.00

Figure 2.6: Virtex 7 690T 45x45 Element Array Resource Usage

using these LUTs. As a design approaches maximum LUT usage, the difficulty of routing all the signals without introducing too much latency increases. Designs consuming over 80% of an FPGAs LUTs can often not be routed by the synthesis and implementation tools. Table 2.6 shows the resource utilization of a 45-by-45 element DANNA on a Xilinx Virtex 7 690T. It can be seen that this design consumes 85% of the LUTs. Array sizes larger than 45-by-45 elements cannot be created reliably. The Xilinx Virtex 7 2000T produces similar resource utilization numbers for a 75-by-75 element array. An implementation of DANNA on an application specific integrated circuit(ASIC) or VLSI chip would also experience array size constraints due to available resources on the chip. Multiple interconnected tiles configured with DANNA allow DANNA to surpass the tile resource limitation and scale beyond the borders of a tile.

Chapter 3

Design of Tiled DANNA

3.1 Overview

Due to the highly parallel nature and integrated timing network of the DANNA architecture, the size of DANNA on a single tile is limited by tile resources. DANNA is currently implemented on a Xilinx Virtex7 690T and Xilinx Virtex7 2000T and has been constrained to a 45-by-45 element array and a 75-by-75 element array, respectively. Tiled DANNA surpasses this limitation by configuring multiple tiles with partitions of a larger DANNA in a two dimensional grid and enabling the DANNA partitions to act in unison as one large DANNA. This allows DANNA to scale to array sizes that are not bounded by tile resource limitations. Each tile in the grid of DANNAs contain a partition of the overall array. The tiles of DANNA implement a communication interface which enables the elements located on the edges of the partitions to pass fire events to one another. This permits DANNA to efficiently scale to unconstrained sizes by spanning across multiple tiles. Figure 3.1 shows the general structure of Tiled DANNA. The master tile of Tiled DANNA, referred to as $\text{DANNA}_{0,0}$ in figure 3.1, is designed to generate the synchronous communication and element clocks necessary for network operation across the multiple tiles.

This master tile approach was chosen to avoid using a separate communications and clocking tile for a Tiled DANNA implementation. As the size of the arrays implemented on Tiled DANNA scale to larger sizes we may chose to use a separate clocking component for these functions. The interface between tiles involves transmitting the synchronous clocks

from the master DANNA partition to the slave DANNA partitions as well as connecting the boundary edge elements. The connected elements can then fire across the tile boundaries while remaining in lockstep, giving the multiple partitions of DANNA the means to operate in unison creating one massive DANNA. The multiple tiles of Tiled DANNA can also be configured as independent DANNAs. Therefore, Tiled DANNA can also operate in a mode where the multiple DANNAs are running in parallel under one host. The remainder of this chapter walks through the design details of Tiled DANNA.

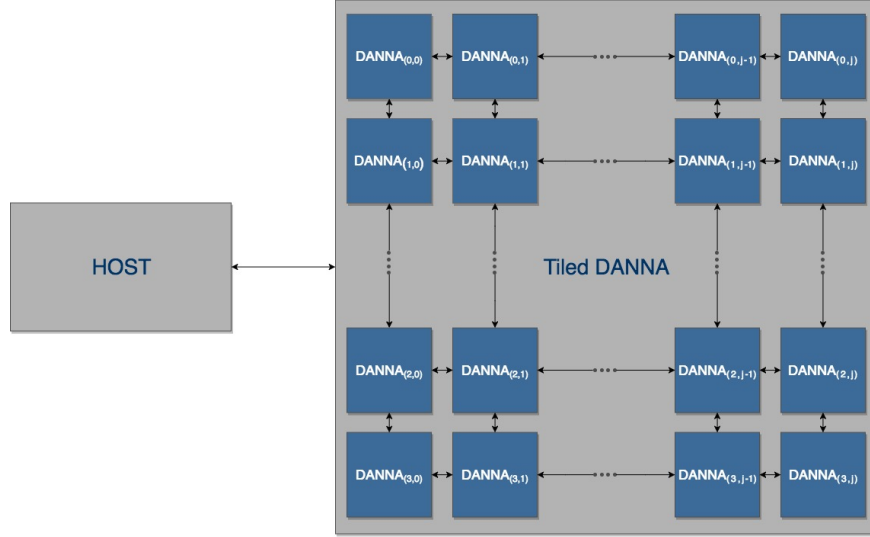


Figure 3.1: Tiled DANNA Structure

3.2 Communication Design

There are two communication interfaces in Tiled DANNA: host-to-array and tile-to-tile. The host-to-array interface employs point to point asynchronous communication over a 32 bit interface at a 100 MHz clock through multiple peripheral communication boards to send and receive packets to and from Tiled DANNA as shown in 3.2. The host-to-array interface could have also been designed such that the host broadcast transmissions and received over multiple lines each with an address specifying distinct tiles. The point-to-point interface was chosen for simplicity. Minimal changes were necessary in the software support and firmware to implement point to point communication. A single broadcast with multiple receives would have required major changes to the original DANNA host to array communication interface.

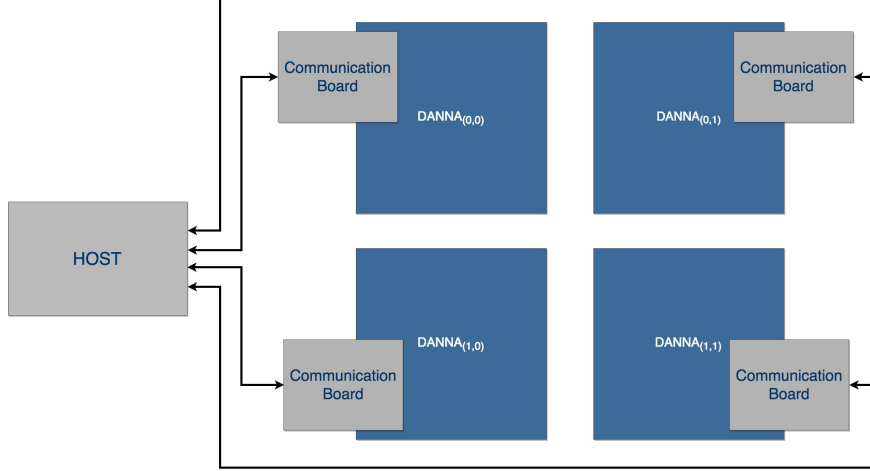


Figure 3.2: Host to Tiled DANNA Communication Interface

The tile-to-tile interface uses synchronous communication across two unidirectional signal lines which transmit 36 bytes over a 36 bit interface at 16 MHz rate as shown in figure 3.3. One bidirectional signal line could have been implemented, but to maintain lockstep array operation across the multiple DANNA partitions with minimal disturbance to the original DANNA behavior, unidirectional signal lines were the better choice. DANNA produces 32 outputs from its edge elements. Bidirectional signaling would not support 32 edge elements to be transmitted and received by the two connected tiles within one network cycle.

The details of the communication between the host and Tiled DANNA are identical to the details of communication between the host and DANNA. This is fully described in Section 2.1. The implementation of the intercommunication between partitions is described in section 3.2.2. The initial prototyping effort of Tiled DANNA was implemented on a structure consisting of two tiles. Figures 3.4 and 3.5 depict the initial prototyping effort of Tiled DANNA for both modes of operation. Any reference to Tiled DANNA from this point forward refers to this implementation.

3.2.1 Host to Tiled DANNA Communication Design

As stated, Tiled DANNA can operate in one of two modes: Tiled Mode or Parallel Mode. Both modes of operation require one primary host with multiple established USB 3.0 interfaces. Tiled mode requires the USB connection to the $\text{DANNA}_{(0,0)}$ tile to transmit

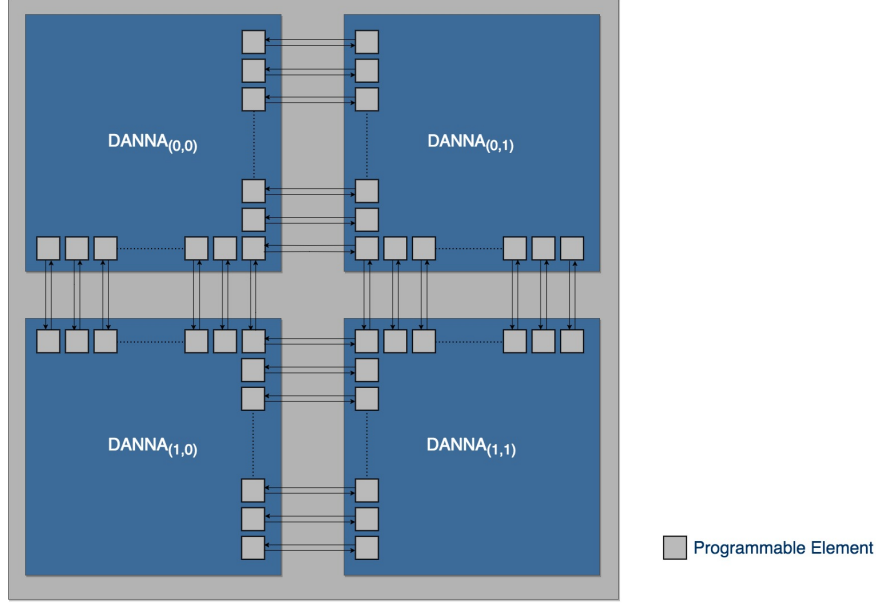


Figure 3.3: Tiled DANNA Tile-to-Tile Communication Interface

the 36 byte command packets to Tiled DANNA. The USB connection to the $\text{DANNA}_{(0,1)}$ tile will only receive the 64 byte response packets from Tiled DANNA, refer to figure 3.4. This allows the elements that reside on the input edge of the master DANNA tile to be configured to accept the external array inputs, and the elements located on the output edge of the slave DANNA tile are configured to produce the array outputs for the response host. The implementation details are discussed in 4.2.1.

Tiled DANNA configured to operate in parallel mode is designed to have each USB interface transmit and receive packets to and from its corresponding FPGA. See figure 3.5. The communication to and from each DANNA array and its corresponding USB interface is operationally identical to the description found in chapter 2.1. The communication is asynchronous over a 32 bit interface running at a 100 MHz rate. Buffering on a peripheral communication board, controlled by a finite state machine in the FPGA, manages the communication flow. Each USB interface will send commands and receive responses from its corresponding FPGA. As Tiled DANNA scales while using USB 3.0, the number of necessary USB interfaces will be equal to the number of tiles in the system. The implementation details are discussed in figure 4.2.1

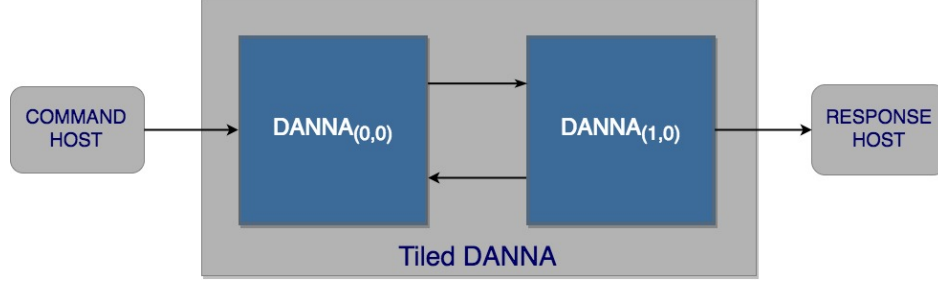


Figure 3.4: Host to Tiled DANNA Operating in Tiled DANNA Mode

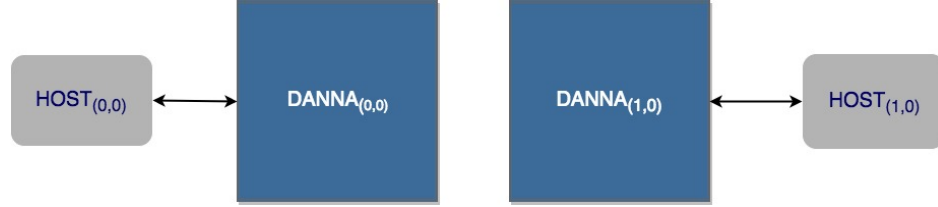


Figure 3.5: Host to Tiled DANNA Operating in Parallel Mode

3.2.2 Interchip Communication Design

Tiled DANNA has the ability to connect elements physically located on one tile to elements physically located on a second tile allowing one DANNA to span multiple components. The elements making the connections across the tile boundaries are known as boundary elements. As shown figure 3.6, they are located on the output edge of the master array and the input edge of the slave array. Elements located on the tile boundaries of corresponding rows have a one to one connection. All fire events that occur in boundary elements on one tile during one network cycle are transmitted to the boundary edge elements of the connected tile over the course of the next network cycle at a 16 MHz rate. Each boundary edge can contain up to 32 elements. The 32 eight bit fire values and 32 one bit fire signals from the boundary elements on one tile need to be transmitted to the 32 corresponding edge elements on the connected tile. This requires the interface to transfer 36 bytes of data in two directions. The synchronous interface between the master and slave requires the data transfer in both directions to be completed within one network cycle. The implementation details of interchip communication are covered in section 4.2.2 and the implementation details of the synchronous global clock are found in section 4.3.1.

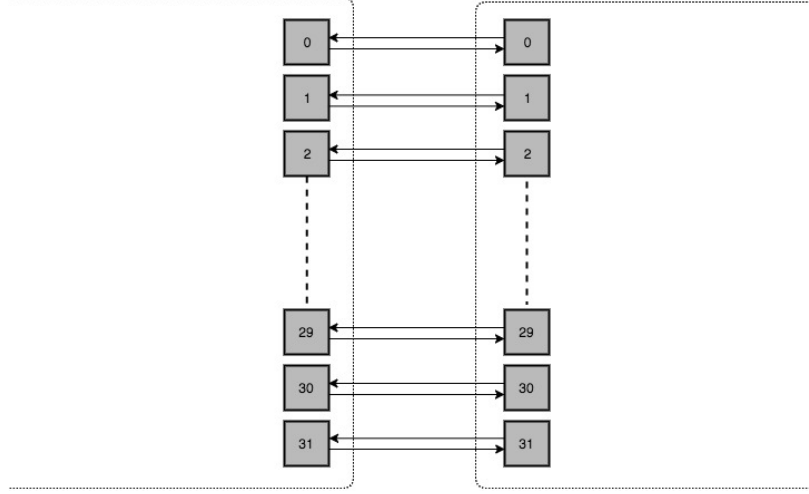


Figure 3.6: Tiled DANNA Boundary Edge Element Connections

3.3 Synchronous Clock Generation Design

Recurrent spiking neural network such as DANNA are event driven networks and the timing of events is imperative to fundamental operation. Nine different clocks are needed to run DANNA: two host-to-array communication clocks and seven synchronous element clocks. Four of the seven element clocks are gated, enabling the DANNA network to be controlled by the host computer (e.g. run, step and halt). The frequency and phase of the nine clocks are listed in table 2.8. The clocking structure of Tiled DANNA is quite unique in that the clock network will not only cross clocking domains (communications and element operation) within one tile, but the clock network spans multiple tiles, enabling lockstep operation across the entire network. To accomplish this, the necessary clocks are generated on the master tile and then transmitted to the slave tile. This was an implementation choice to limit the number of components, but larger scale systems may choose to implement a separate clock component. Both the master and slave use the generated clocks to maintain communication with its connected USB 3.0 interface, to run the array components, and to create the gated element clocks. All seven element clocks are synchronous, and allow for lockstep array operation across both tiles. The implementation details are fully described in section 4.3.1.

3.4 Design of Slave Control Signals

Tantamount to synchronous operation is a synchronous network cycle across both tiles. Within one network cycle sixteen subcycles occur. This allows the 16 element input ports to be polled and the components within the elements to function accordingly. When DANNA is given a run command the global network cycle following the run command begins array operation. When operating in Tiled mode only the master DANNA tile is fed commands. To enable the other DANNA tiles to operate, the master must send a control signal allowing the other tiles to be enabled to run. Also at the beginning of the global network cycle immediately following a run command a counter is enabled that maintains the timestamp of the network cycles. A second control signal is necessary to send to the other DANNA tiles to allow the timestamp counter to be enabled.

3.5 Boundary Edge Element Connections

Each element of DANNA typically connects to 16 neighbors. Therefore, each element has 16 input ports. One input port for each of its connected neighbors. When an element fires, it broadcasts its fire weight. In a single tile implementation six of the sixteen input ports of boundary edge elements are unused. Boundary edge element connections are shown in figure 3.7 Thus, each input edge element could potentially accept six input signals from across the tile boundary. The boundary edge elements are currently designed to only use one of the six off tile neighbor connections. The five other input ports are left disabled. The interface between the tiles is designed to be 36 bit operating at a 16 MHz rate. If more connections were enabled across the tile divide the interface would need to be wider or run at faster rate. The boundary edge elements are connected in a one to one, reciprocating East – West connection as shown in figure 3.7.

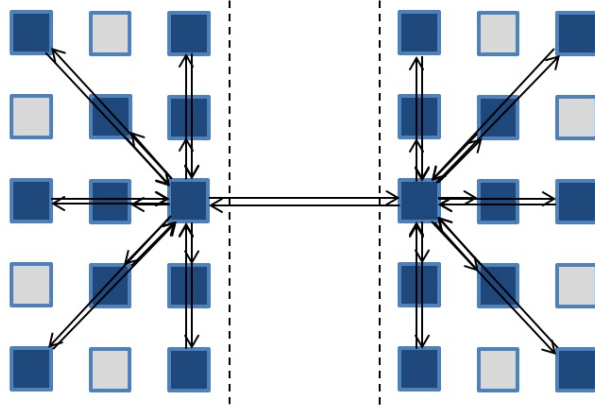


Figure 3.7: Tiled DANNA Boundary Edge Element Nearest Neighbor Connections

Chapter 4

Implementation of Tiled DANNA

Due to the highly parallel nature and integrated timing network of the DANNA architecture, the size of DANNA on a single tile is limited by tile resources. DANNA is currently implemented on a Xilinx Virtex7 690T and Xilinx Virtex7 2000T and has been constrained to a 45-by-45 element array and a 75-by-75 element array, respectively. Tiled DANNA allows DANNA to break this constraint by spanning across multiple tiles configured with partitions of DANNA. In Tiled DANNA, DANNA partitions are distributed across multiple tiles laid out in a grid, allowing one DANNA to function seamlessly, while spanning the multiple tiles. The following chapter covers the implementation process involved in realizing Tiled DANNA.

4.1 Porting DANNA

4.1.1 Prototyping Systems

To allow the DANNA architecture to scale across multiple tiles, the original DANNA design needed to be ported to a new prototyping system. The prototyping system was required to allow for synchronous clocks across the multiple tiles and provide sufficient connectivity to support and maintain the nearest neighbor connections of DANNA across the connected tile boundaries. The Tiled DANNA design requires the use of 80 signals for interchip communication and 54 signals for host-to-array communication. The host-to-array connection and tile-to-tile connections are to be made through independent connectors.

DANNA communicates with the host using a Cypress FX3 communication board and USB 3.0. The prototyping system was required to provide a means for the host to maintain communication through the Cypress FX3 communication board.

Few manufacturers offer systems that contain interconnected FPGAs on a printed circuit board. HiTech Global manufactures the HTG-747 with four Xilinx Virtex 7 2000Ts on one board[13] while Onix[Platforms] and Prodesign manufacture modular host systems designed to prototype large ASIC designs that may require multiple FPGAs. The Onix system can provide scalability to support three Xilinx Virtex Ultrascale 440 FPGAs and Prodesign[ProFPGA] produces a line of ProFGPA prototyping systems capable of supporting four Xilinx 690T or 2000T FPGAs. The HTG-747 provides two FMC connections to each of the FPGAs. This will limit the connectivity of Tiled DANNA by allowing only one FPGA to be connected to another. More than two FPGA connections would need to be made through a daisy chain of HiTech Global boards. The available Onix system documentation was not sufficient to make a valid system comparison. Prodesign produces a line of motherboards with the capability to interconnect multiple FPGAs. Their concept involves modular systems that provides access to nearly all the I/O signals on an FPGA through extension sites and specially designed interconnect boards. Figure 4.1 provides images of the ProFPGA quadmotherboard prototyping platform.

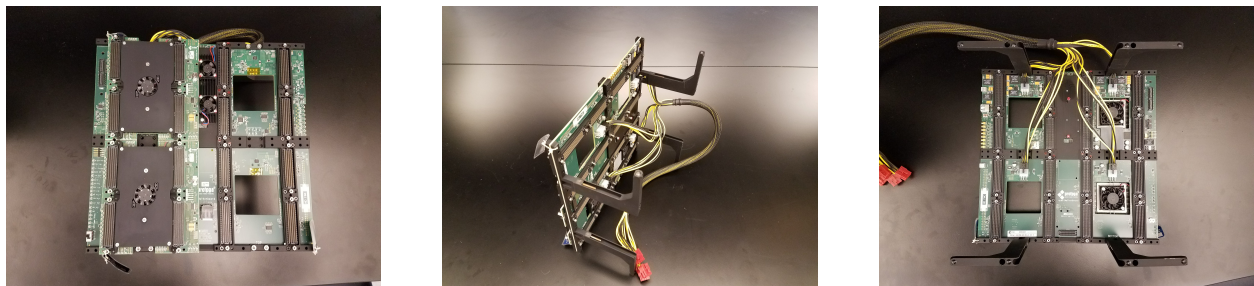


Figure 4.1: ProFPGA Quadmotherboard Prototyping System

The Prodesign ProFGPA quadmotherboard along with two Xilinx Virtex7 690T modules was chosen to implement the initial prototype of Tiled DANNA. Each of the 690T FPGA modules on the ProFPGA quadmotherboard are designed with six extension sites that allow for access to a total of 738 I/O signals. The quadmotherboard can hold up to four Xilinx 690T FPGAs. The Prodesign system also provides 8 clock signals that are guaranteed to be

fully phase synchronous through the use of their provided software and logic components. The motherboard has the capability of communicating with the FPGAs over PCIe, USB 2.0, and Ethernet, but not USB 3.0. Four of the six extensions provide access to 148 I/O signals, one provides connectivity to 98 I/O signals, and one accesses 48 I/O signals. Figure 4.2 provides information regarding the available I/O pins on each of the extension sites.

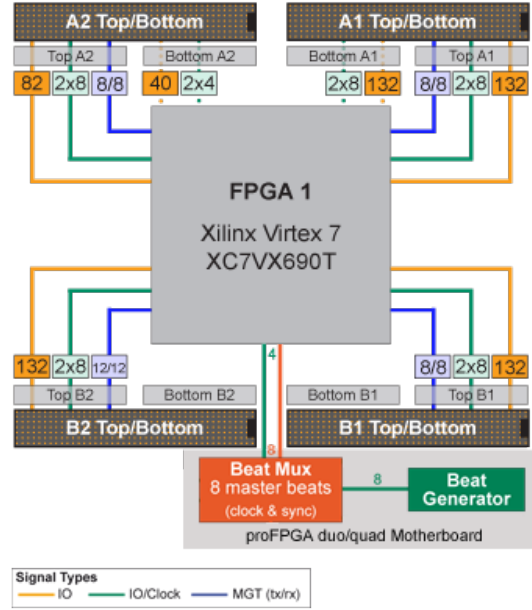


Figure 4.2: I/O Signal Accessibility on ProFPGA 690T Module

The extension sites containing connectivity to 98 I/O signals were chosen to incorporate interconnect boards enabling the Cypress Superspeed FX3 Communication Board to continue to bridge the communication between the Host and DANNA. ProFPGA extension cables were used to connect corresponding extension sites that provide access to 148 I/O signals. This implementation only makes use of three of the extension sites. This leaves open the opportunity to scale DANNA further on the ProFPGA quadmotherboard. The figure below shows the ProFPGA quadmotherboard fully connected to the two Xilinx 690T modules.

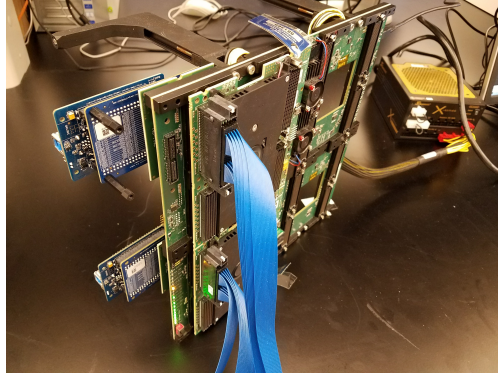


Figure 4.3: Tiled DANNA Prototype

4.1.2 Pin Mapping

Previously DANNA was implemented on the HiTech Global HTG-777: Virtex 7 690T FPGA FMC Module[14]. The first step to porting DANNA to the proFPGA quadmotherboard was to determine which pins on the ProFPGA quadmotherboard were to be used to access the I/O signals on the FPGA that DANNA was using to communicate with the host. A translation of the pin names from the HiTech Global board to the ProFPGA quadmotherboard was done to accomplish this.

4.2 Communication Implementation

4.2.1 Implementation of Host Communication

As described in section 3.2, Tiled DANNA is capable of operating in two modes. Tiled DANNA can run as multiple DANNAs operating in parallel or run as one DANNA spanning multiple FPGAs. Under both modes of operation, the host is required to communicate with multiple USB devices and have the ability to differentiate between the communication from the different tiles. This parallel communication is established through the firmware loaded on to each of the FX3 communication boards connected to the individual tiles. The firmware loaded on to each FX3 behaves identically and required no change from the original DANNA design to allow Tiled DANNA to be implemented. The communication flow is described in Section 2.1. However, the USB descriptor used to bind to each USB 3.0 interface is subtly

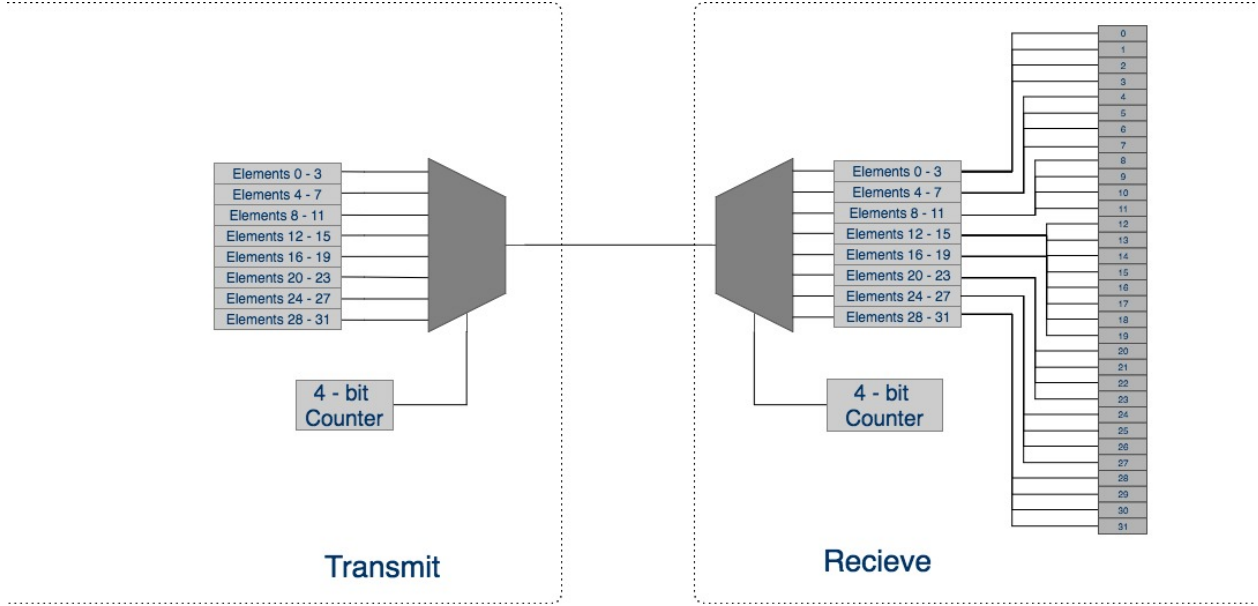


Figure 4.4: Fire Event Multiplexing and Demultiplexing Across the Tile Boundary

different. A USB descriptor contains a field reserved for the product identification number of the USB device. By default, all Cypress FX3 Superspeed communication boards have a product identification number of F1. To make each FX3 device unique, the F1 product identification number was changed from F1 to F6 on the FX3 connected to the FPGA that receives the synchronous clocks. With each FX3 identified uniquely, the primary host of the system is able to bind to the multiple USB ports separately and establish communication with each FPGA virtually simultaneously.

4.2.2 Implementation of Interchip communication

The interchip communication of Tiled DANNA supports the communication of 32 boundary edge elements located on each of the connected DANNA partitions. The communication consists of transmitting all the fire weights and fire signals from the boundary edge elements on one FPGA to the boundary edge elements on the connected FPGA. The communication between boundary elements is continuous. The boundary edge elements fire weight and fire signals are pulled from the elements four at a time at a 16 MHz rate and fed to an 8-1 mulitplexer. Refer to figure 4.4.

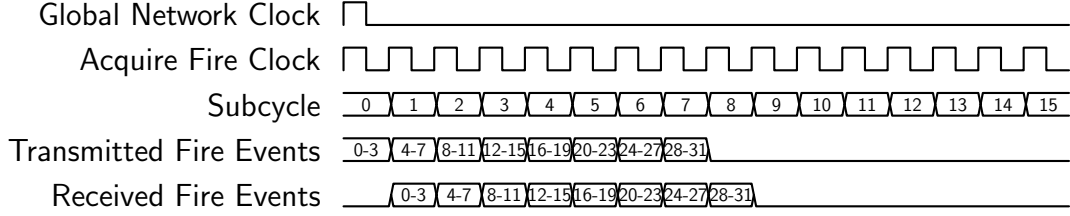


Figure 4.5: Timing Diagram of Fire Event Data Transfer

This creates a 36 bit interface transmitting four element fire weights and four element fire signals at one time. A timing diagram of the fire event transfer is shown in figure 4.5. The 36 bytes are decoded on the receiving FPGA and stored in a latch. On the next network global cycle the latched values are routed to the registers of the input ports of the appropriate boundary elements. A counter synchronous to both FPGAs is used to keep the transmission aligned to connected elements.

Boundary edge elements located in corresponding rows on the two tiles are connected. The boundary edge elements are numbered 0 through i where i is the row in which the boundary element resides. If an element has not fired during a network cycle its fire weight and fire signal are zero. With this bandwidth and transmission rate the tile-to-tile interface can support the transmission of fire events between 64 elements during one network cycle, while the initial implementation established communication between 32 elements. This transmission scheme allows all the fire events to be transmitted over the course of one network cycle allowing the DANNA partitions to remain in lockstep, but results in a one cycle delay between the fire events that occur across the tile divide.

4.3 Implementation of Synchronous Clock Generation

4.3.1 Synchronous Clock Generation

Tiled DANNA requires two communications clocks and the seven element clocks. Array operation requires three free running clocks with frequencies of 32 MHz, 16 MHz, and a 1 MHz pulse and four gated clocks with frequencies of 32 MHz, 16 MHz, 16 MHz 90° phase shifted and a 1 MHz pulse. The master FPGA is required to generate and distribute the synchronous clock across all tiles of Tiled DANNA. To accomplish this a LogiCORE Clocking

Wizard IP block integrated in the Vivado Design Suite was used. The LogiCORE Clocking Wizard creates a verified clocking network within one tile by using dedicated clocking logic known as a mixed-mode clock manager (MMCM) and phase locked loops (PLL).[\[34\]](#). The clocking wizard chooses the appropriate board oscillator and clocking primitives, multiplies or divides the input clock to the desired user specified clock frequencies and performs necessary phasing shifting in reference with the input clock. The resulting synchronous phase aligned clocks are distributed throughout the entirety of one FPGA tile through dedicated clock lines with relatively no jitter. The clocking wizard takes a differential 100 MHz signal from the proFPGA quadmotherboard and generates the six free running clocks listed in [table 4.1](#). The 1 MHz global network clock is not a clock with a 50% duty cycle. This clock is a pulse with a width of 32 MHz that occurs every 1 MHz. The clocking wizard cannot create clocks with a frequency lower than 10 MHz[\[33\]](#), so one cycle of a 16 MHz clock is enabled after 16 cycles. This creates a 1 MHz pulse with a 32 MHz width to serve as the global network clock. The timing diagram in [table 4.7](#) shows how the 1 MHz pulse is created in reference to the element subcycles.

Table 4.1: Clocks Generated on the Master Tile of DANNA

Frequency	Clocks Generated
32 MHz	Accumulator Clock
16 MHz	Acquire Fire Clock
16 MHz 90deg phase shift	Accumulator Enable Clock
1 MHz	Global Net Clock
100 MHz	System Communication Clock
100 MHz 180deg phase shift	Communication Bus Clock

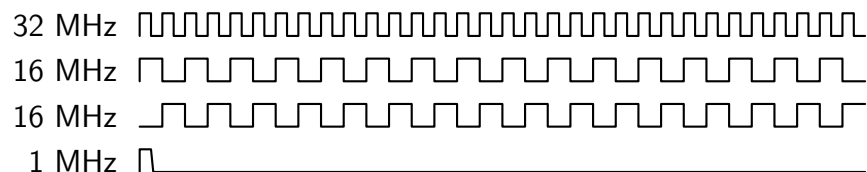


Figure 4.6: Clocks Generated the Master Tile

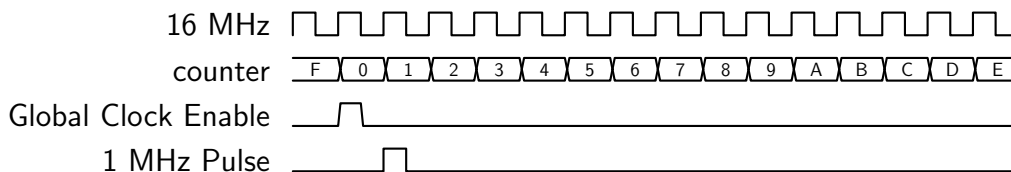


Figure 4.7: Timing Diagram for 1 MHz Network Pulse Generation

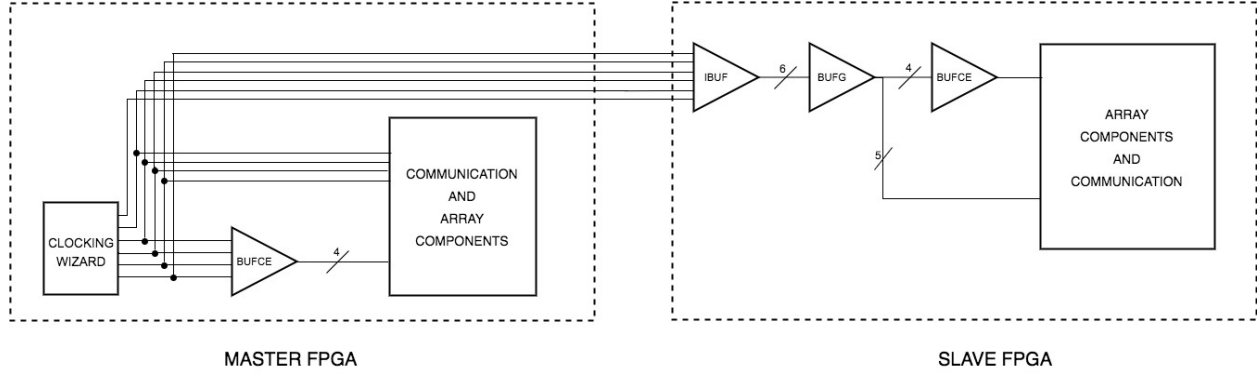


Figure 4.8: Clocks Signal Routing of Tiled DANNA

All six clocks are routed through output pins of the master FPGA and used directly by the master DANNA partition of Tiled DANNA. By passing these clean clock signals off the master FPGA through dedicated I/O clock pins, across interconnect cables, and into the slave through dedicated I/O clock pins the six free running clocks remain synchronous when they reach the input pins of the slave partition of DANNA. The six clocks that come into the slave from the master are first routed through global input buffers and then through global clock buffers. The global input buffers bring the six free running clocks into the clock network built into the fabric of the FPGA. Once in the clock network, the six clocks were each required to be routed through one of the 32 global clock buffers to guarantee that the clocks would remain synchronous when distributed to the logic components of the array. Figure 4.8 depicts the clocks signal routing. Not only do the clocks on the slave need to be buffered appropriately, the clock signals need to be defined appropriately for the Vivado Design Suite software tools to ensure the clocks are edge aligned and routed such that the clock latency remains within bounds of time allowed by the clock tree. The software tools are given the `create_clocks` command in the constraints file. This command has parameters to tell the software tools which port the clock signals will be coming from, the frequency of the input clocks, when the rising and falling edge should occur, and at which point in the design the signals begin to be used as a clock.^[1] The six commands used in Tiled DANNA are shown in figure 4.9.

```

create_clock -period 31.250 -name ac -waveform {0.000 15.625} [get_ports {outclks[0]}]
create_clock -period 62.500 -name afc -waveform {0.000 31.250} [get_ports {outclks[1]}]
create_clock -period 62.500 -name aec -waveform {15.625 46.875} [get_ports {outclks[2]}]
create_clock -period 1000.000 -name gnc [get_ports {outclks[3]}]
create_clock -period 10.000 -name sys_clk -waveform {0.000 5.000} [get_ports {outclks[4]}]
create_clock -period 10.000 -name busclk -waveform {5.000 10.000} [get_ports {outclks[5]}]

```

Figure 4.9: TCL Command Used to Define Slave Clocks

These commands allowed the software tools to appropriately align the input clocks, and with the previously described buffering, the DANNA clocks remain synchronous across both tiles allowing the DANNA partitions to run in unison.

4.4 Implementation of Slave Control Signals

All the elements of DANNA run synchronously under a global network cycle. A network cycle contains 16 subcycles that allow for all 16 input ports to be polled for firing events. A counter maintaining the timestamp of each network cycle runs in alignment with the global network clock. When the array receives a run or a step command from the host a signal is sent to all the elements to initiate element operation on the next network cycle. Another signal is also set to start the counter for the timestamp. Since only the master partition will be receiving commands, the control signals were sent from the master to the slave. Figure 4.10 provides the timing diagram which indicates when the control signals are set.

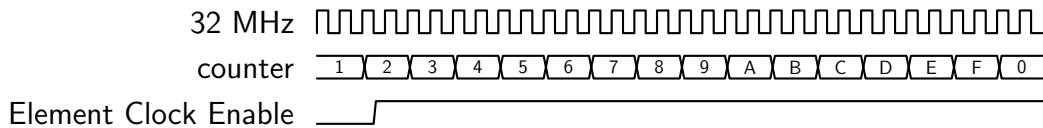


Figure 4.10: Timing Diagram for Enabling Element Operation

4.5 Defining Element Connections and Behavior

DANNA participates in external communication through I/O edges. The DANNA input edge is defined by all the elements that are located in the first column and the DANNA output edge is defined by all the elements that reside in the last column. Tiled DANNA involves taking

numerous DANNAs and tiling them in a grid fashion. Thus, a grid with divides between the FPGA tile boundaries is formed. The elements that lie along this divide are known as boundary elements. Elements located in corresponding positions on different tiles make identical connections, but each element has a unique physical address. Elements within Tiled DANNA will exhibit identical behavior as elements in single DANNA. However, since there is a one cycle delay occurring with the transmission of fire events across the tile boundaries, the elements located on the boundary will be delayed by one network cycle on receiving the signal indicating potentiation or depression should occur. This causes the behavior of the boundary elements to be unique to the rest of the elements. To allow the boundary elements to operate as close to a non boundary element, potentiation and depression is prevented from occurring. This is accomplished by programming the element to expect the potentiation and depression signal from an element connection that is undefined.

The programmable elements of DANNA are connected to 16 of its nearest neighbors. The 16 neighbors of an element consist of the elements located in the immediate and inter cardinal directions of the element, creating an inner ring of eight. An element is also connected to the elements that lie in the immediate and inter cardinal direction of the inner ring of neighbors, creating an outer ring of eight elements. The connectivity scheme used for the 16 connections is to have the element connect to its eight nearest neighbors and then connect to eight additional neighbors one layer out in the same direction as the inner eight connection as illustrated in Figure 2.4[5]. Each programmable element has 16 input ports, one for each of its connected neighbors. Element connections in DANNA reciprocate, so if an element is connected to a neighbor through input port x where x is some number between 0 and 15. Then the neighbor is connected to the element through input port x as well. The element orientation and numbering scheme is fully described in subsection 2.2.3. The element input ports that always face directly to the East or West of DANNA are either input port 2 or input port 6, depending on the parity of the column number in which the element resides. If the column is even then the element is of orientation type 0 or 2 and input port 6 faces to the West. The element accepts fire events from the neighbor located to its immediate left and input port 2 faces East and accepts fire events from the neighbor connected to its immediate right. If the element resides in an odd column then the element is of orientation type 1 or 3

and input port 2 faces West and is connected to the neighbor immediately to the element's left and input port 6 faces to the East and is connected to the neighbor immediately to the right.

This means that all the boundary edge elements on the slave will always have input port 6 facing to the West, enabling input port 6 to be the port to accept the fires from across the tile divide. However, the boundary edge of the master may fall on an even or an odd column number. If the boundary edge on the master falls in an even column then input port 2 faces to the East, enabling it to accept connection from across the tile divide. Likewise, if the boundary edge falls in an odd column input port 6 will face to the East and will be enabled to receive inputs from across the divide. This causes a discrepancy in the element connection scheme as the input port connections may not always reciprocate when connecting the master and slave. For example, in an array of size 45-by-45 the boundary edge will fall on column 44. This will force the boundary elements of the master to connect with the slave through non reciprocating input ports. The master will fire into input port 6 of the slave, but the slave will be firing into input port 2 on the master. This is not typical element connection behavior.

Chapter 5

Testing and Validation

The testing and validation of Tiled DANNA involves running a series of automated tests designed to stress various aspects of the tiles configured with DANNA to verify Tiled DANNA hardware configurations are fully functional. The objectives of testing Tiled DANNA included verifying the clocks across the tiles were synchronous, the independent tiles of DANNA can operate independently, and that the tiles of Tiled DANNA can function in unison to allow DANNA to span across the multiple tiles to produce cycle accurate responses.

5.1 Methodology

The testing procedure involves configuring the FPGAs with a DANNA consisting of predetermined number of rows and columns of programmable elements. Using custom support software, the DANNA elements are loaded with a network designed for a specific application or function. Again using custom software, a series of commands are issued to run and initiate fire events on the hardware configuration of DANNA. The input fire events are directed to the appropriate input elements at a specific time with a specific weight. The response packets are produced by the hardware are captured and compared with the response packets that are generated from the event simulator. The event simulator is a software simulated version of DANNA and behaves exactly like the hardware. Identical simulator and hardware output verifies the hardware configuration is functioning as expected. There

are four different types of test networks used to verify the functionality of the Tiled DANNA: application, straight passthrough, snake passthrough, and timing grid. The application test networks result from evolutionary optimization of a specific application. Unfortunately, there are no application test networks large enough to test the hardware configurations of Tiled DANNA operating in Tiled Mode. The straight passthrough network is designed to verify synchronous operation. The snake passthrough network is designed to ensure the connections across the tile borders are functioning correctly. The timing grid network is a fully connected network that stress the functionality of long term potentiation and long term depression as well as appropriate element behavior. Examples of the three test networks are shown in figures [5.1](#), [5.2](#), and [5.3](#).

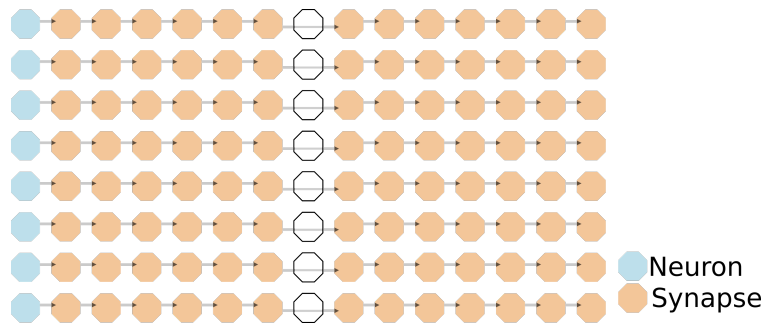


Figure 5.1: Straight Passthrough Network

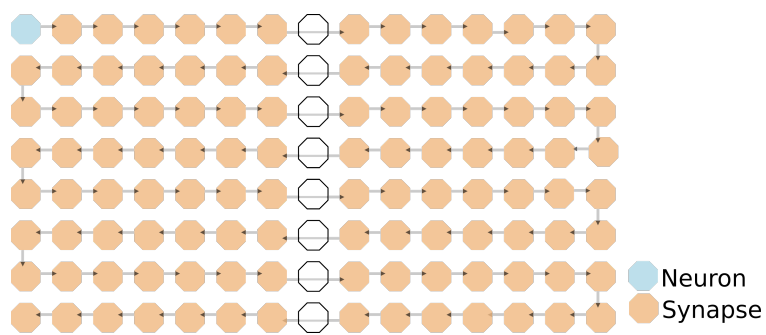


Figure 5.2: Snake Passthrough Network

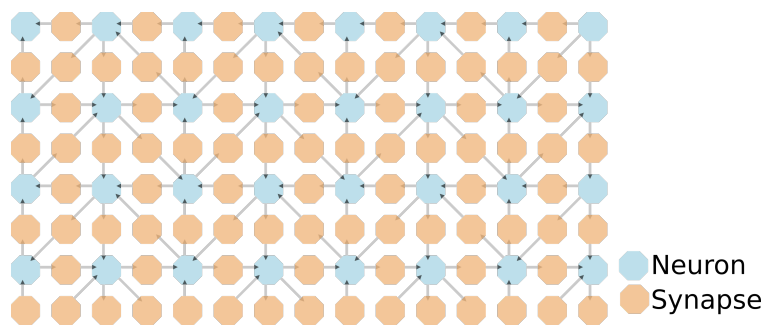


Figure 5.3: Timing Grid Network

5.2 Tiled DANNA Test Procedure

The Tiled DANNA prototyping effort began with porting the original DANNA design from the HTG-777 to two proFPGA XC7VX690T FPGA modules connected to a proFPGA quadmotherboard. First, it was important to verify the clocks across the multiple tiles were generated correctly. Along with clock verification testing, we verified each tile of Tiled DANNA was operating correctly as an independent DANNA. Finally, Tiled DANNA operating in tiled mode was tested to verify synchronous operation across multiple chips. The four different types of network configurations previously described were used to test various aspects of Tiled DANNA functionality.

The straight passthrough networks are constructed such that each row begins with a neuron connected to a synapse and the remainder of the row consists connected synapses. The neuron is programmed with a maximum threshold. Thus every fire into the neuron will cause the neuron to fire. The synapses are programmed with a distance of one, so there will be no delay in when the fire events will begin. There are three test sets applied to the straight passthrough network. One test set issues a fire every cycle on every input. Firing every cycle on every input shows that the signals are propagating through each row as expected. The expected output from this test should show all the elements firing every cycle. The second test set issues one fire on one row every cycle. The row to be fired on increments by one every cycle. Doing this staggers the fires into the array. This will result in only one output element firing every cycle. The third test set issues one fire on a single row for a very large number of cycles. This test is designed to stress the host to array communication. This test will cause the network to produce a response packet every cycle for 500,000 cycles and will push the communication buffers to the maximum to ensure that no response packets are being dropped.

The snake passthrough network is designed to stress both directions of the tile-to-tile interface as well as fire propagation. This network consists of a single neuron on row one connected to a string of synapses that connect through the network from one side to the other, down to the next row, and back across the network. This snaking pattern allows fires to propagate over the tile borders in both directions. Only one fire is asserted on the input

neuron. Monitoring the output packets from the master tile and the slave tile, the fire can be tracked as it traverses the network. This snake passthrough network is only used to test Tiled DANNA operating in tiled mode.

The timing networks are fully connected networks. The network is composed of a four neuron-synapse connections. The four neuron-synapse connections create a 4-by-4 grid that is replicated across a network. This network construct allows the behavior and timing of element firing events to be verified. There are 8 test sets used on each timing grid network. Each test set consists of one fire into all of the inputs every x cycles where x varies in each of the 8 test sets as 1, 2, 4, 6, 8, 16, 32, 64, 128. All the timing grid tests run for 2,000 cycles.

5.2.1 Independent Mode Testing

To verify the partitions of Tiled DANNA can function as independent DANNAs operating in parallel, hardware configurations of 10-by-10, 15-by-15, 25-by-25, 32-by-32, and 45-by-45 element arrays were loaded on to each of the 690T modules attached to the proFPGA quadmotherboard one at a time. The automated tests for the applications, straight passthrough, and timing grid networks were ran on each of the various size hardware configurations of both the master and slave tiles. The master was tested first. After verifying the master tile of DANNA was functioning appropriately, verification of the slave tile functioning as an independent DANNA running off the clocks provided from the master tile was done. Successful results from these tests only verify that tiles can function independently and the clocks are generated and transmitted by the master correctly. These test do not verify that the clocks are synchronous across both tiles. The results from these tests are discussed in section 5.3.

5.2.2 Tiled Mode Testing

To verify the clocks were synchronous across the multiple tiles of DANNA, each tile was configured with a partition of a larger DANNA array. The tiles were loaded with networks that stress the connections and functionality of the tiles operating in tandem to ensure the array was clocking and communicating as desired.

To accomplish this, one set of load commands was given to both tiles through a single stream of load commands. Each tile only accepts the load commands that contain element addresses that reside within it. As described in section 4.5 the boundary elements are disabled from potentiating and depressing because the potentiation/depression signal will be delayed by one network cycle. The straight passthrough and snake passthrough, and timing grid networks along with their corresponding test sets are used to test synchronous array operation. The results are discussed in section 5.3

5.3 Tiled DANNA Test Results

5.3.1 Independent Mode Test Results

The first test of Tiled DANNA was to verify one tile of Tiled DANNA was functioning appropriately. To accomplish this the master tile was configured with five different array sizes and the previously described networks were used to test the master tile functionality. As discussed in section 6.2, the initial Tiled DANNA implementation introduced a counter that did not exist in single DANNA. The first round of testing indicated that the initial implementation was not functional. The communication clocks were misaligned with the global network clock resulting in the halt packet not being received. This left the host in an indefinite state. To remedy this the counter was removed and the single DANNA logic was adopted. This modification allowed the smaller array sizes to function. However, the medium size arrays would only pass the straight passthrough test. Through monitoring of array response packets it was clear the array was not functioning as expected. Some output elements were firing at the wrong times. It was determined that the elements were not being loaded correctly and at incorrect times. Eliminating the feedback clocks on the master tile alleviated this issue. Making these two modifications to the initial implementation on the master the automated tests were successful on the small and medium size arrays on the master tile. The Vivado Design Suite synthesis and implementation tools were unable to route the largest array sizes. This indicated too many resources were being utilized by the

design. As described in section 6.4, unnecessary global clock buffers were removed from the design. This allowed all hardware configurations to pass all test on the master tile.

The slave tile output showed problems similar to the master tile, however for different reasons. The results from the straight passthrough network on the slave hardware configurations indicated that the array was firing at the wrong times and with the wrong elements. As described in section 6.3 the clocks were not generated locally and needed to be specified for the Vivado synthesis and implementation tools. A change to the clock definitions was made. This allowed the small and medium size arrays to pass the straight passthrough and timing grid tests. Only the larger arrays were left producing incorrect responses for the straight passthrough and timing grid networks. Introducing input buffering on the clock signals reduced the burden on the global clock buffers allowing the larger hardware configurations to pass the network tests.

5.3.2 Tiled Mode Test Results

Verification of synchronous clocks across multiple chips and the functionality of the tile-to-tile interface is made through testing Tiled DANNA while operating in tiled mode. The straight passthrough test is used to ensure the array spanning two chips is running in lockstep and is producing fires at appropriate times. One test involves firing on all rows at once and one test involves firing on each row independently.

The results from this test indicated all rows with the exception of the last four rows were firing appropriately. Element monitoring through the response packets from boundary edge elements on the master indicated the boundary edge elements on the master were firing appropriately, but the last four rows of Tiled DANNA were producing no responses. The Vivado Design Suite debugging tools allowed the signals carrying the transmission from the master boundary elements to the input ports of the slave boundary elements to be probed. These probes indicated the fires were being received by the slave, the signals were reaching the appropriate element input ports, but the elements were not firing. The commands to load and run DANNA are sent in the same file. By separating the load commands from the run commands it was discovered that the slave tile was not able to load all of the elements before the master started operating. To remedy this, the halt command was sent to the slave

after the load commands. This forced the host to wait until the slave was finished loading all the commands before issuing the run and fire commands to the master tile. The output from Tiled DANNA spanning multiple chips matched the output from the event simulator. These successful results from the straight passthrough network verified the clocks across the multiple chips are synchronous and the tile-to-tile communication interface from the master tile to the slave tile was functioning as expected.

The next test ran was the snake passthrough test. Monitoring the response packets from the master tile allowed the fire events occurring across the interconnect to be tracked. The response packets from the master indicated the boundary edge elements on the master and slave were firing as expected as the fire signal propagated through the array and overall array output matched the output from the event simulator. Successful results from this test verify the tile-to-tile interface running in both directions is operating correctly and is fully functional.

Chapter 6

Progression and Challenges

6.1 ProFPGA

As previously stated in subsection [4.1.1](#), the proFPGA quadmotherboard produces 8 synchronous clocks which are accessible to all FPGA modules attached to the motherboard. These 8 clocks run at fixed frequencies. The Prodesign software along with an on-board Virtex 7 can divide or multiply these 8 clocks to any user defined frequency between 4 MHz and 400 MHz. The FPGA modules attached to the motherboard gain access to these user defined clocks using the provided proFPGA components. The user design is to incorporate the proFPGA control module alongside one of the two clock synchronization modules. One clock synchronization module guarantees to bring synchronous clocks to run within each FPGA module, but does not guarantee that those clocks will be fully phase synchronous across the FPGA modules. To obtain fully phase synchronous clocks across all FPGA modules, the advanced clock management module must be used. An attempt to bring this module into Tiled DANNA was made. However, after contacting Prodesign it was discovered that the Prodesign hardware being used was loaded with outdated firmware and the advanced clock manager provided by ProFPGA was incompatible with quadmotherboard. The attempt to incorporate the proFPGA components was abandoned, and the master tile of DANNA was responsible for generating the necessary communications and array clocks. The only signal being pulled from the quadmotherboard is one 100 MHz clock.

6.2 Clock Generation

Initially two 100 MHz, 32 MHz, 16 MHz 90° phase shifted, and two 16 MHz clocks were generated using the clocking wizard on the master FPGA. These six signals were to be used by both the master and the slave to generate the two communication clocks and the 7 element clocks. On the master, the six clock signals were routed to six bidirectional I/O ports that were mapped to dedicated clocking pins. A tcl command in the masters constraints file directed the synthesis and implementation tools to reference the six I/O ports as the primary source point for the generated clocks and not the clock management blocks generated them. The slave made a one to one pin connection with the master clock signals through an interconnect cable. Thus, the six clock signals from the master were routed directly to the slave through dedicated clock pins. The slave constraints file also included the same command for the synthesis and implementation tools to reference the input ports as the primary source for the clock signals. All six clocks on both the master and the slave were routed through global clock buffers before being used by the communication and array logic. This was the initial design and implementaion of Tiled DANNA. The results are discussed in subsection 5.3. The master and slave design did not respond. The global network clock of DANNA is a 16 MHz pulse that occurs every 1 MHz. The clocking wizard cannot generate a clock with such a low frequency. To create the 1 MHz pulse, a counter in the master and another in the slave were added to the design to enable one of the 16 MHz clocks for one cycle after every 16 cycles. Timing diagram 4.7 shows when the enable signal is set on the master and slave. For DANNA to be functional, this counter not only needed to be synchronous between the master and slave, but also to be synchronous with the existing DANNA logic. To reduce complexity, the newly added counter was removed from the master and the slave. The preexisting counter used to generate the 1 MHz pulse enabling signal in the original DANNA architecture was used. This counter was used in the master to enable one of the 16 MHz clocks generated by the clocking wizard for one cycle after every 16 cycles. This 16 MHz wide pulse that is generated every 1 MHz in the master replaced one of the 16 MHz clocks being transmitted to the slave.

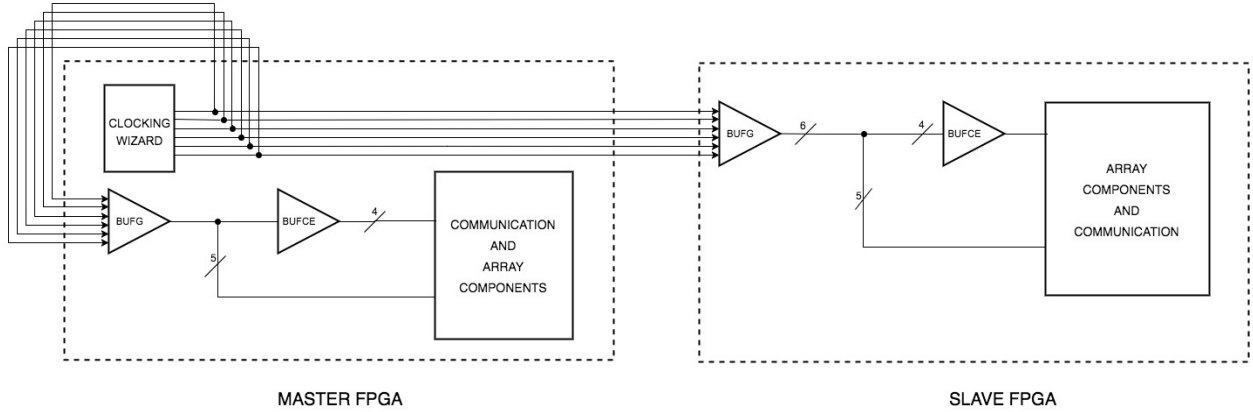


Figure 6.1: Initial Synchronous Clock Design

Eliminating the counter and transmitting the 1 MHz pulse from the master allowed the master to 10x10 function fully. However, array sizes larger than 10x10 would not function on the master and the slave produced incorrect responses. It was determined the elements were not being loaded correctly. This was remedied after the port definitions of the clock signals on the master were changed from bidirectional ports to output ports. The master 10x10, 15x15, 25x25, and 32x32 arrays passed their respective tests. Only the 10x10 array passed the tests on the slave. No changes were made to the commands in the master and slave constraints file. Under this design and implementation the Vivado software tools were unable to generate a configuration that did not introduce too much latency for a 45x45 array.

6.3 Clocking Constraints

The constraints file of a Vivado Design Suite project contains information that speaks to the software tools in regard to signal definitions and relationships to ensure signals are routed appropriately. As previously stated, a command was given in the constraints file of both the master and the slave. This command was the create clocks command, and it creates clock objects or redefines previously defined clock objects for the timing engine to assist in synthesis and implementation.^[1] The create clocks command was used to overwrite the definition of the clock objects generated by the clocking wizard in the master. The same create clocks command in the slave created six new clock object. The source point

for the six clocks on both the master and the slave were the input ports the clock signals were passed through. As previously described, the six clocks were initially routed through bidirectional ports on the master, and then later changed to output only ports. When the port declaration for the six clocks changed, the create clocks constraint should have been removed from the constraints file. It was not, yet the master design was still able to generate 10-by-10, 15-by-15, 25-by-25, and 32-by-32 array sizes that passed the test suite with the unwarranted create clock command. The slave design will always need the create clock command because the primary source point is from the input ports. The synthesis and implementation tools need to be told where the signals are originating from and what they will look like to appropriately synthesize and route them. With the initial command only defining the source point and period, the 10-by-10 DANNA slave design was the only array size that was functioning appropriately. The 15-by-15 DANNA became functional when the waveform definition was made by specifying when the clock edge transitions were to occur in the create clocks command.

6.4 Clock Buffers

The DANNA slave design did not initially include input clock buffers. The design pulled the clocks from the input ports and directly into global clock buffers. It was found that the global clock buffers were not sufficient when bringing clocks in from an input port on larger DANNAs. The larger designs required the input clock signals to be routed through input clock buffers. After incorporating the input clock buffers, the 25-by-25 DANNA was functional on the slave design. 10 global clock buffers from one clock region were being used by the master design. There are only 12 global clock buffers located in each clocking region. This pushed the design close to maximizing its clocking resources making it unable to be routed by the implementation tools. Eliminating all the global clock buffers from the master allowed the 45-by-45 DANNA to meet timing requirements and become fully functional as independent DANNA arrays.

6.5 Tiled DANNA Mode

The fire events transmitted over the tile-to-tile interface occur in groups of four at a 16 MHz rate. With 32 boundary edge elements full transmission across the interface is complete within 8 cycles of the 16 MHz clock. The transmission over the tile-to-tile interface is synchronized under the global network clock and a four bit counter. A network cycle in DANNA consists of 16 subcycles that occur at a 16 MHz rate. The transmission of the first four element fire events are to occur on subcycle 0, and each proceeding subcycle transmits the next consecutive set of four element fire events. There exists a four bit counter in the DANNA design that runs off the free running 16 MHz acquire fire clock. Initially, this counter was used to enable transmission of the fire events over the tile-to-tile interface. Count zero of this counter was used to transmit the first set of four and count one on this counter was used to receive the transmission from the master. Also, in the initial implementation of the tile-to-tile interface, the transmission of fire events was unintentionally gated by the 16 MHz clock. The output packets of Tiled DANNA with the initial tile-to-tile interface resulted in rows 8 through 26 producing fire events a cycle early, rows 0 through 7 producing fire events at the correct time, and rows 27 through 31 not producing fire events at all. It was discovered that count zero of this counter is not in line with subcycle 0. Count two of this counter occurs during subcycle 0. The counter that was initially used is latched in the design. The four bit latched value became the new four bit signal used to enable transmission. Subcycle 0 occurs when this latched value is 1. After the signals are decoded by the receiving side they are held in a latch until the next network cycle. Latching the signals allowed for rows 8 through 26 to now produce response packets at the appropriate times. Rows 27 - 31 were still not producing responses. Monitoring of the output from the master tile indicate that the transmission from the master is occurring as expected and using Vivado's provided debugging tools it is apparent the signals for the fire events of rows 27 through 31 are making it through the master slave interface as expected. However, the boundary elements of rows 27 through 31 are not producing any output.

The host manages how the commands are sent to the array. The host parses the network configuration file. It sends all the load commands to the master tile through the master usb

interface and then all the load commands to the slave tile through its usb interface. Once the loads are issued to both devices, the commands to run the array are sent to the master tile. The slave usb interface provides no signal to the host to indicate it has completed loading all the commands. This flow caused the last four rows of load commands to be lost in the communication buffers. To remedy this, a halt command is sent to the slave after the load commands are issued. This causes the host to wait for the halt response before issuing the run and fire commands allowing the slave to load all of its elements before array operation.

Chapter 7

Future Work

7.1 Scaling Tile-to-Tile Interface

Tiled DANNA is one DANNA that spans across multiple tiles, thus a tile-to-tile transmission interface is required to send the fire events between elements that are connected across a tile divide. Currently, the tile-to-tile communication involves a synchronous 36 bit interface running at a 16 MHz rate. By transmitting element fire events using the described interface, four element fire weights and four element fire signals can be transmitted every cycle of a 16 MHz clock. With this bandwidth at this transmission rate the interface can guarantee that the fire events from potentially 64 elements will cross the tile boundary and be received by connected elements with a delay of exactly one network cycle. The tile-to-tile interface can be modified in several ways to allow more element connections. The bandwidth can be increased and/or the transmission rate can be increased. The interface can also become asynchronous operating at much higher rate.

An asynchronous communication interface would not constrain the fire events between tiles to a network clock. Fire events between boundary elements can be transmitted as they occur, and not on every network cycle, under an asynchronous implementation. This would result in lower power consumption. However, when network activity across the tile boundary increases the probability of receiving the fire events across the tile divide in one network cycle diminishes.

Modifications to the synchronous interface can also be made to increase boundary element connections. The bandwidth, transmission rate, or both can be increased. If the interface was to remain 36 bits wide and the rate of transmission was increased to 32 MHz, 128 element connections can be guaranteed to cross the tile divide with only a one network cycle delay. If the bandwidth was doubled and the 16 MHz transmission rate remained the same, 128 element connections could also be made. However, this is not possible due to pin limitations. As table 7.1 shows under a 72 bit interface the number of required pins would be 152. Each 690T module connected to the ProFPGA quadmotherboard has the potential to make 148 interconnections through one extension site. The 72 bit interface exceeds this by four pins. Thus, the maximum bandwidth possible due to pin limitations is 62 bits. Under this bandwidth with a 16 MHz transmission rate 112 element connections can be made. If both the bandwidth and transmission rates were increased to 62 bit and 32 MHz respectively, the interchip communication has the potential to support 224 element connections.

Table 7.1: Pin Requirements Under 36, 62, and 72 Bit Bandwidths

Signal	36 bit interface	62 bit interface	72 bit interface
Clocks	6	6	6
DANNA Control	2	2	2
Fire Weights	32 x 2	56 x 2	64x2
Fire	4 x 2	7 x 2	8x2
Pin Requirement	80	134	152

7.2 Increase Boundary Edge Element Connections

The current implementation of the tile-to-tile interface is 36 bits operating at a 16 MHz rate supports 32 boundary edge elements. The communication involves transmitting four eight bit element fire weights and four one bit element fire signals in one transmission. There are two signal lines so the transmission can occur simultaneously in both directions. For all the elements in DANNA to remain in lockstep the transmission across the tile divide needs to be fully complete within one network cycle. Under the described interface the necessary fire

event information for 32 elements is completely transferred in .5 MHz, exactly half of one network cycle. This means the current interface design can support twice as many element connections then are currently defined. Without making any changes to how the interface operates 64 element connections can be supported. Implementing this change would only involve configuring DANNA to span at least 64 rows.

7.3 2x2 Matrix of DANNA Tiles

The initial prototyping effort consisted of implementing Tiled DANNA on two FPGA tiles configured with DANNA in a 1-by-2 tile structure. The next step is to span DANNA over four tiles connected in a 2-by-2 structure. This involves making vertical connections between two 1-by-2. Currently, the elements that are located on the last row of DANNA do not have any southern input ports activated. Therefore, the six input ports located in the South-West, South, or South-East of the inner and outer layers are not defined connections. Likewise the northern input ports of the first row of a DANNA are not actively connected. So the input ports located in the North-West, North, and North-East of both the inner and outer layer are open connections. Thus, if a 1-by-2 implementation of Tiled DANNA was tiled below another 1-by-2 implementation of Tiled DANNA, the set of boundary elements would increase to include the elements located on the first and the last row of DANNA. The tile-to-tile interface would occur on the South side of one tile of DANNA and the North side of a second tile of DANNA. Under the element orientation scheme implemented on DANNA, this makes input ports 0 and 4 the two input ports externally accessible. The Tiled DANNA implementation that is looking to make connections to its North will always have input port 0 accessible to the tile divide, but the Tiled DANNA implementation looking to its South may have input 0 or 4 accessible to the tile divide depending on the parity of the row number. Even rows will have input port 4 facing externally and odd rows will have input port 0 facing externally. Once the orientation is determined the element connections can be defined to appropriately configure the FPGA for the appropriate partition of DANNA.

Chapter 8

Conclusion

The initial prototyping effort has proven to show that a Tiled DANNA structure can support multiple DANNA networks running simultaneously or one large network running across multiple tiles. To accomplish this clocks were successfully generated by one tile and transmitted to a second tile configured with DANNA. This second tile was able to run solely off the clocks generated and provided by the first tile. Tiled DANNA has also demonstrated that cycle accurate behavior for a single array can be seen across multiple tiles. Using specifically designed test networks and boundary element monitoring fire event propagation can be seen across the rows of tiled DANNA from input to output. The output packets that are received by the host from the hardware match those produced by the software event simulator verifying cycle accurate operation. By maintaining nearest neighbor communications between elements the scalability of Tiled DANNA is only limited by the signaling capability from the master clock subsystem. The same mechanisms used to connect two tiles together can be translated and replicated to numerous tile connections. As the clock signals are required to traverse larger and larger distances the ability to support multiple tiles will become difficult to maintain due to the load and latencies that will be introduced onto the clock signals.

Bibliography

- [1] (2013). *Vivado Design Suite User Guide: Using Constraints*. Xilinx, ug903 edition.
- [2] Boahen, K. (2006). Neurogrid: emulating a million neurons in the cortex. In *Conf. Proc. IEEE Eng. Med. Biol. Soc*, page 6702.
- [3] Churchland, P. S. and Sejnowski, T. J. (2016). *The computational brain*. MIT press.
- [4] Daffron, C., Chan, J., Disney, A., Bechtel, L., Wagner, R., Dean, M. E., Rose, G. S., Plank, J. S., Birdwell, J. D., and Schuman, C. D. (2016). Extensions and enhancements for the DANNA neuromorphic architecture. In *IEEE SoutheastCon 2016*, Norfolk, VA.
- [5] Daffron, C. P. (2015). DANNA a neuromorphic computing VLSI chip. Masters Thesis, University of Tennessee.
- [6] DARPA (2017). Darpa synapse program.
- [7] Dean, M. E., Chan, J., Daffron, C., Disney, A., Reynolds, J., Rose, G. S., Plank, J. S., Birdwell, J., and Schuman, C. D. (2016). An application development platform for neuromorphic computing. In *International Joint Conference on Neural Networks*, Vancouver.
- [8] Dean, M. E., Schuman, C. D., and Birdwell, J. D. (2014). Dynamic adaptive neural network array. In *13th International Conference on Unconventional Computation and Natural Computation (UCNC)*, pages 129–141, London, ON. Springer.
- [9] Disney, A., Reynolds, J., Schuman, C. D., Klibisz, A., Young, A., and Plank, J. S. (2016). DANNA: A neuromorphic software ecosystem. *Biologically Inspired Cognitive Architectures*, 9:49–56.
- [10] Feldman, M. (2016). Ibm finds killer app for truenorth neuromorphic chip.
- [11] Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of Neural Engineering*, 13(5).
- [12] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

- [13] Global, H.
- [14] Global, H. Htg-777: Virtex 7 fpga fmc module.
- [15] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.
- [16] Hsu, J. (2014). Ibm’s new brain [news]. *IEEE Spectrum*, 51(10):17–19.
- [17] Kelly III, J. and Hamm, S. (2013). *Smart Machines: IBM’s Watson and the Era of Cognitive Computing*. Columbia University Press.
- [18] Khan, M. M., Lester, D. R., Plana, L. A., Rast, A., Jin, X., Painkras, E., and Furber, S. B. (2008). Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2849–2856. Ieee.
- [19] Kloss, C. (2016). Nervana engine delivers deep learning at ludicrous speed!
- [20] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- [21] Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673.
- [22] Minsky, M. and Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press.
- [23] Mujtaba, H. (2017). Intels lake crest chip aims at the dnn/ai sector 32 gb hbm2, 1 tb/s bandwidth, 8 tb/s access speeds, more raw power than modern gpus.
- [24] Pitts, W. (1942). Some observations on the simple neuron circuit. *The bulletin of mathematical biophysics*, 4(3):121–129.

- [Platforms] Platforms, O. A. P.
- [26] Press, S. C. (2015). China successfully developed 'darwin,' a neuromorphic chip based on spiking neural networks.
- [ProFPGA] ProFPGA. Fpga based prototyping solution.
- [28] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.
- [29] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE.
- [30] Schuman, C. D. (2015). *Neuroscience-Inspired Dynamic Architectures*. PhD thesis, University of Tennessee.
- [31] Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. arXiv:1705.06963.
- [32] Shen, J., Ma, D., Gu, Z., Zhang, M., Zhu, X., Xu, X., Xu, Q., Shen, Y., and Pan, G. (2016). Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *Science China Information Sciences*, 59(2):1–5.
- [33] Suite, V. D. (2015). *LogiCORE IP Product Guide: Clocking Wizard v5.1*.
- [34] Suite, V. D. (2016). *Clocking Wizard: LogiCORE IP Product Guide*. Xilinx, pg065 edition.
- [35] Union, E. (2017). The human brain project.
- [36] Venkateswaran, N., Krishnan, A., Kumar, S. N., Shriraman, A., Sridharan, S., et al. (2003). Memory in processor: A novel design paradigm for supercomputing architectures. In *ACM SigArch Computer Architecture News*, volume 32, pages 19–26. ACM.

Vita

Patricia Eckhart is originally from Brooklyn, New York, but claims Norris, Tennessee as her home town. After high school, she obtained her first Bachelors of Science degree in Secondary Education with a Concentration in Mathematics from Tennessee Technological University. Patricia's first year teaching was spent at Campbell County High School in Jacksboro, Tennessee and her next nine years of teaching were spent at Bearden High School in Knoxville, Tennessee. While teaching at Bearden High School, Patricia was named 2008 Teacher of the Year and was classified at the highest level under the Tennessee teacher evaluation model several years in a row. She earned her first Masters of Science Degree in Educational Theory and Practice through an online program with Arkansas State University. In the Fall of 2011, Patricia returned to school part-time challenging herself and moving forward with the intention of becoming a college professor. A spark was lit inside her when she took a required physics class. This prompted Patricia to alter her course and leave the teaching profession. She returned to school full-time to enter the engineering field. She attended Pellissippi State Community College and obtained an Associate of Science degree in Electrical Engineering Technology which led to a laboratory technician position at the Spallation Neutron Source. Through encouragement from her family and co-workers at the Spallation Neutron Source, Patricia continued her education at the University of Tennessee, where she graduated Magnum Cum Laude with her second Bachelors of Science degree in Computer Engineering in the Fall of 2015. She then obtained the Bodenheimer Fellowship and a research assistantship, giving her the opportunity to extend her education with a second Masters of Science degree in Computer Engineering to be conferred in August of 2017.