Going for Brain-Scale Integration – using FPGAs, TSVs and NOC based Artificial Neural Networks: A Case Study

Conference Paper · September 2014

DOI: 10.1145/2674095.2674098

2 authors:

Mand N. P
KTH Royal Institute of Technology
2 PUBLICATIONS **10** CITATIONS

SEE PROFILE

Johnny Öberg
KTH Royal Institute of Technology
95 PUBLICATIONS **2,513** CITATIONS

SEE PROFILE

Going for Brain-Scale Integration – using FPGAs, TSVs and NOC based Artificial Neural Networks: A Case Study

Nowshad Painda Mand
KTH Royal Institute of Technology
Stockholm
Sweden
+46729088507
pmnow@kth.se

Johnny Öberg
KTH Royal Institute of Technology
Stockholm
Sweden
+4687904127
johnnyob@kth.se

ABSTRACT
With better understanding of brain's massive parallel processing, brain-scale integration has been announced as one of the key research area in modern times and numerous efforts has been done to mimic such models. Multicore architectures, Network-On-Chip, 3D stacked ICs with TSVs, FPGA's growth beyond Moore's law and new design methodologies like high level synthesis will ultimately lead us toward single- and multi-chip solutions of Artificial Neural Net models comprising of millions or even more neurons per chip.
Historically ANNs have been emulated as either software models, ASICs or a hybrid of both. Software models are very slow while ASICs based designs lacks plasticity. FPGA consumes a little more power but offer the flexibility of software and performance of ASICs along with basic requirement of plasticity in the form of reconfigurability. However, the traditional bottom up approach for building large ANN models is no more feasible and wiring along with memory becomes major bottlenecks when considering networks comprised of large number of neurons.
The aim of this paper is to present a design space exploration of large-scale ANN models using a scalable NOC based architecture together with high level synthesis tools to explore the feasibility of implementing brain-scale ANNs on FPGAs using 3D stacked memory structures.
.

Categories and Subject Descriptors
C.2.1[**Computer-Communication Networks**]: Network Architecture and Design *–Packet-switching networks*

General Terms
Design, Experimentation, Performance

Keywords
ANN, TSVs, NoC, Brain-Scale Integration, FPGA.

FPGAWorld'14, September 9-11, Stockholm and Copenhagen.

DOI: 10.1145/2674095.2674098

1. INTRODUCTION
The idea of Artificial Neural Networks (ANNs) was initially proposed by Warren McColluch and Walter Pitts [1] in 1943 when they come-up with "Threshold Logic Unit". It matured enough for practical applications in nearly four decades. It has found numerous practical applications in data intensive domains like video compression, image processing, robotics, pattern recognition, energy physics, and speech synthesis etc [2]. Additionally many neuroscientists have performed a lot of research about information processing performed in the brain which helped them to model brain activities using ANNs. Two main aims of such modeling are either to build a complete processing unit utilizing working principles of brain or use very limited neurons as part of an embedded system. The former is still far away from reality but the latter is already practical in many real-world applications [2][4].

ANNs are inspired by the working principle of brain where billions of neurons are networked together in a complex manner and are processing in parallel unlike Von-Neumann machine which is based on sequential processing. ANNs has distributed memory while serial machines are normally based on centralized memory. Due to inherent parallelism scientists believes that in future it may lead us to new computer architectures.

Implementing an electronic brain is one of the grand challenges [3], and many attempts are being made across the world to construct the first electronic brain capable of emulating a human-sized cortex [4][5]. The majority of ANNs in these projects are emulated by either software models running on Von-Neumann machines, ASICs or a Hybrid of both. The software models are very slow when considering large number of neurons as they are normally executed on sequential machines while ASIC based designs lack plasticity and are not cost-effective for small volumes.

Recently DARPA funded a neural image processor that promised to be 1000 times faster than conventional image processors by using memristors for storing synapses [6]. ANN applications are growing very rapidly and to experiment with their full capability, we need large faster models comprising of huge number of neurons.

FPGAs implementations offer the flexibility of software and performance of ASICs along with the basic requirement of plasticity in form of partial/full reconfiguration with little power penalty and has less density as compared to ASICs. Additionally their rapid prototyping capability along with growing capacity

beyond Moore's law makes them an ideal platform for researchers to experiment with ANN models.

Small-scale ANN models are already in practical applications [2][4] but large-scale models still face many challenges. ANN models do not scale-up with growing neurons when implemented directly in silicon due to non-linear growth in computational resources and circuitry. Other challenges for large-scale modelling are complexity, plasticity and power consumption. The advent of multicore architectures, Network-On-Chip and heterogeneous 3D stacked ICs along with new design methodologies like high level synthesis, are promising for potentially single-chip or few chips ANN models comprising of millions or even more neurons in the near future.

Conventional bottom-up approach for building such large systems are no more feasible due to increasing integration and testing costs. In the new design methodologies, like high level synthesis, complex systems are directly generated from high level specifications, leaving low-level optimizations to automated EDA tools.

Xilinx Virtex-7H580T [7] is the world first 3D heterogeneous all programmable IC utilizing Xilinx's Stacked Silicon Interconnect (SSI) technology, enhancing inter-die bandwidth manifolds and in near future 3D stacking of ICs using TSVs promises to solve ICs density growth beyond Moore's law. According to the ITRS roadmap, around 2020 it would be possible to integrate multi-layered 2.5D or 3D chips.

In this paper, we sketch an architecture, where the Brain functionality is built using a gigantic network of ~10^6 Brain-Processing-Units (BPUs), partitioned across a couple of thousand chips, using off-chip interconnects. We discuss the limitations in terms of neuron accuracy, memory need per BPU chip, and the number of TSV connections needed and explore the design space of a couple of possible implementations.

2. NEURAL NETWORK BACKGROUND

A neuron is the basic building block of Artificial Neural Network. In its simplest form, an artificial neuron is based on integrate and fire mechanism and the whole brain consist of very complex web of distributed tiny neurons that work in parallel. The distributive parallel processing is considered to form basis for the immense processing power of the brain. Synapses are responsible for inter-neuron communication. The human cortex is believed to consist of 10^{11} neurons with average connections of 1000 to 10,000[8]. Basic integrate and fire model of neuron is depicted in Figure 1 according to McColluch and Walter Pitts [1] proposed model.

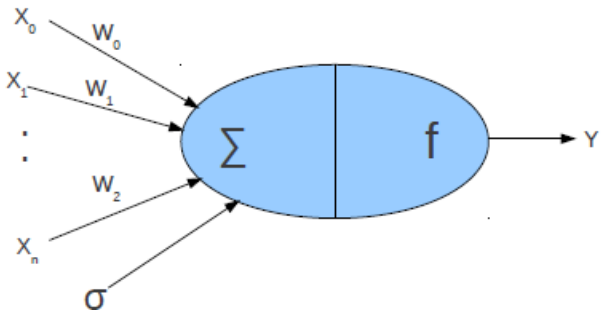

Figure 1: Basic Neuron integrate and fire model

The accumulated sum is passed through a limiting function called activation function. If the sum is above a certain threshold then the neuron fires and communicates this with all the connected outward neurons; otherwise nothing is done. Common activation functions are sigmoid, linear, ramp and sinusoidal etc.

There are two well-known structures of ANN based on their connectivity. The complete network is in the form of layers, where a layer get input from one layer and feed another layer. When such connections are only in forward direction the network is call feed-forward neural network as shown in Figure 2. The layer at start is called input layer, while layers in the middle are called hidden layers and the layer at the exit is called output layer. Network with either one or more than one hidden layer is called multilayer perceptron [9]. The well-known back propagation learning algorithm [10] is applied to such neural network.

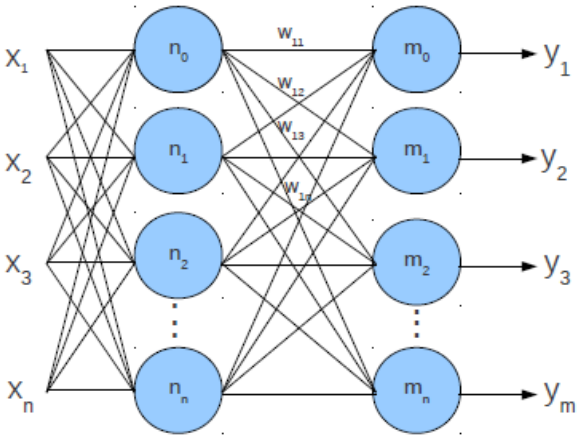

Figure 2: Feed forward neural network

The second types of neural network has connections in backward direction to other layers or to other neurons in the same layer in addition to the connections in the forward direction as shown in Figure 3.This type of neural networks are called recurrent neural networks. Well-known examples of such network are Hopfield Neural Network [11] and Kohonen's selforganizing map [12].

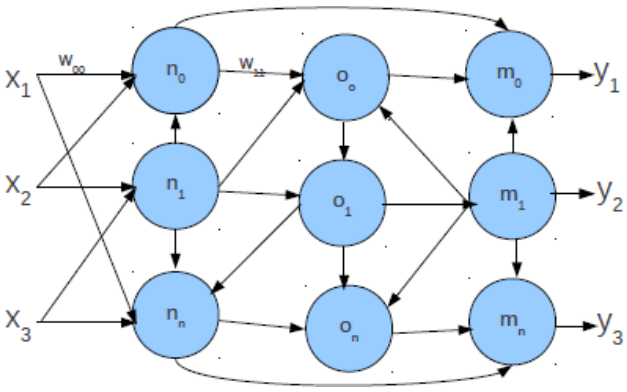

Figure 3: Recurrent neural network

3. BACKGROUND AND RELATED WORK

In its simplest form, the human brain cortex can be viewed/modelled as an Artificial Neural Network consisting of ~10^{11} Neurons, where each neuron is connected to ~1,000 other neurons on average and firing on average at ~100 Hz. The neurons are often organized in mini-columns of ~100 neurons,

where the neurons inside each mini-column share inputs. The mini-columns in turn are grouped together in hyper-columns, where groups of ~100 mini-columns share inputs. The cortex is further divided into six layers. There is one additional layer, whose neural mini-columns span through the other layers in an orthogonal way [13].

There exist many different more or less complicated Neuron Models. In essence, most of them are feeding a weighted sum through a limiting function. In case the output is updated/changed, it is sent through the network to all neurons it is connected to.

One way of looking at the problem is to view it as a gigantic sampled DSP system, where all nodes in the network should be visited every ~10 ms. Thus, if all the neurons in the network fired simultaneously, the network of DSPs should be able to perform $\sim 10^{13}$ neuron calculations per second, i.e., $\sim 10^{15}$GFlops.

Luckily, the firing in a network is quite sparse ($\sim 10^{-2}$), and if only changes in the output of a neuron are sent across the network, the amount of information needed to be sent between hyper-columns is decreased further. Moreover, keeping the neurons' weights and output values in a local memory also reduces the traffic.

Several architectures for implementing a human brain cortex executing at real time have been suggested recently. The majority of such brain emulation projects are based on ASICs, Software or hybrid of both. The SpiNNakar [14] project at Manchester University is based on multiple tiny CPUs on a single chip, communicating through a packet-switch based NOC; and at secondary level, via a network between these chips. The ARM968 processor forms the basis for these CPUs. Each CPU is capable of emulating around 1000 neurons. It utilizes a simplified neuron model while their software based simulator is capable of more complex neuron models. BlueBrain [15] is based on IBM Blue Gene supercomputer with 8000 CPUs to simulated a software based ANN model. The project is able to simulate a few thousand neurons at much slower speed than the actual brain. They work recently on more complex detailed models with some learning algorithms. C2S2[16] at IBM with collaboration of five other universities from different fields succeeded in building a massively parallel simulator utilizing a supercomputer with 147,456 CPUs and 144TB of main memory .They successfully emulated 1.6B neurons with a few trillion synapses. FACETS [5] is a major European project of fifteen groups from seven universities with major ASIC foundation. They succeeded in emulating very limited number of analog neurons incorporating digital communication. They fabricated various chips from time to time and successfully emulated 180,000 neurons with 40 million synapses. NeuroGrid [17] at Stanford University is based on analog programmable CMOS chips. They successfully assembled 16 chips together, each emulating around 65000 neurons.

IFAT [18] [19] and NeuronDyn [20] at University of California San Diego incorporated analog neuron chips. Their fabricated chip, IFAT, successfully emulated 2400 neurons. They successfully deployed this chip in a few embedded applications. Later they increased its capacity to 65,000 neurons. IFAT chip in conjunction with Xilinx Spartan-6 FPGA succeeded in emulating 250,000 neurons. They also fabricated more realistic and detailed chip, the NeuroDyn, emulating only 4 neurons and 12 synapses with detailed 384 parameters controlled from Matlab software on a workstation. Very recent project EMBRACE[21][22] employed mix-signal approach utilizing scalable NOC along with analog neuron model to design ASIC base ANNs model with aims of high speed and low power consumption. Again very limited numbers of neurons are successfully modeled till date using RTL–level models. BioRC[8] is based on carbon nanotubes and graphene with the ultimate goal of fabricating synthetic cortex. However, it is just at the very starting phase and will take considerable time to prove its feasibility.

4. ARCHITECTURE

Our vision is to describe the ANN model using high level neural net model and then to synthesize any ANN' architectural model using state of the art high level synthesis tools along with its design-space exploration and finally map the synthesized model on a NOC based FPGA platform which will give us size, performance and power requirement metrics of various models.

We envision a digital architecture, as shown in Figure 4, below. The neuron calculations are divided into chips, where each chip contain a number of Brain Processing Units (BPUs), connected together using a Network-on-Chip structure with both on-chip and off-chip connections. Each BPU perform exactly one Neuron

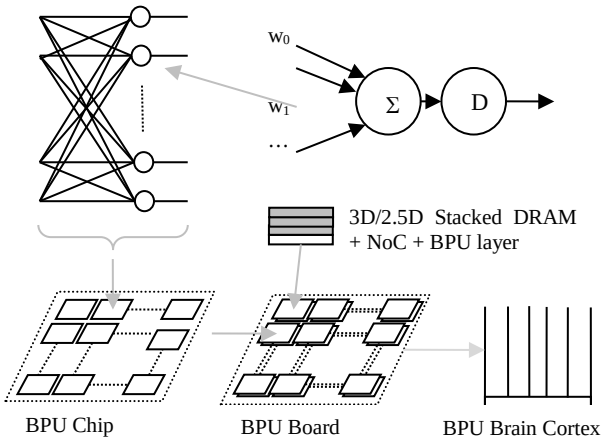

Figure 4: Simple ANN organized as mini-columns, communicating with DRAM through TSVs and across an on- and off-chip hierarchical NoC

calculation per BPU cycle. The BPU cycle is determined by the bandwidth of the memory containing the input-weight pairs of the calculations associated with the BPU. To get sufficient bandwidth, we envision the BPU chips to be implemented as a chip stack, with a single computation chip connected to a set of TSV DRAM memories stacked on top of it. The chips are then organized into arrays on a couple of boards; with a mother board to implement the global interconnect between BPU boards and to external sensor arrays.

For communication between the BPUs, a packet-switched based NOC will be employed while an off-chip protocol similar to the one used by S. Furber [28] and S. Gao [29] will be used for communication among various BPU boards.

5. DESIGN SPACE EXPLORATION

To be able to build a full size brain, we have to fetch on average 1,000 weights and multiply them with 1,000 outputs for every neuron firing. We have 10^{11} Neurons, with about 1% of them firing every 10 ms. Thus, we need a weight-input memory consisting of $10^3*10^{11}=10^{14}$ input-weights, and we have to fetch $10^3*10^{11}*1\%/10[ms]=10^{14}$ input-weights per second.

The number of BPUs needed to implement the function is also dependent on the speed of the DRAM containing the input-weight-data, the amount of DRAM possible to have in a chip stack, and the number of TSVs that is possible to put on the top layer of each chip.

Regarding the first issue, studies show that the DRAM access times do not shrink with technology [23][24]. On the other hand, the addressing of a standard DRAM today involves a complicated procedure of time-multiplexing of address and data in order to save pins. However, since TSV technology promises to provide many more pins, we are not necessarily restricted by pin out, since we would be able to provide both ROW and COLUMN addresses simultaneously. In addition, we could also pipeline the address decoding, increasing the speed at the cost of a larger area for address decode circuitry. Or, we could read out several lines of data simultaneously, and provide them serially to the computational unit. Thus, the DRAM access time should not be an issue.

Studying the second issue, we see that the DRAM storage we can have is limited by two things, 1) the number of dies we can stack on top of each other and 2) the number of TSVs that we can have to access the memory stack. The first is a function of maturity of technology, which we cannot do anything about, but stacks of 16 layers with memory sizes of 1Tbit and beyond, have been studied [25]. The second is also the limiting factor of the third issue.

Thus, in the end all boils down to the number of TSVs as the limiting factor. The ITRS roadmap from 2012 [26] gives us the information in Table 1.

Table 1:3D stacking TSV diameter as per ITRS

Global Level, W2W, D2W or D2D 3D-stacking	2009-2012	2012-2015
Minimum TSV diameter	4-8 μm	2-4μm
Minimum TSV pitch	8-16 μm	4-8 μm

If we are conservative and say that we cannot have a TSV pitch lower than 8μm, we still get a TSV density of $1cm^2/64\mu m^2$ =1,562,500 TSVs/cm^2. However, we will never need more than $\log_2(10 \text{ ^ } 15)$=50 bits to address any part of the memory, and we will never need to access more than 1,000 input-weights in order to do one calculation. Since the number of the address and resulting output bits are so tiny compared to the input-weights, we can safely ignore them in our further exploration. We then end up with the following equation and Table 2:

$$NrNeuronCalcsPerCycleCm2 = \frac{TSV_{density}}{resolution * NrInputWeightPairsPerNeuron}$$

Table 2: Weight resolution vs Neuron calculations

Input-Weight Resolution [bits]	64	32	16	8
#Neuron Calculations/cycle/cm^2	24.4	48.8	97.7	195.3

Since the cycle time is determined by the DRAM access time only, we can use this information as the other parameter in a design space exploration to figure out the number of needed BPU chips, the number of BPU array boards, etc., as a function of the input-weight resolution. In the following calculations, we assume that the maximum chip size is a die with the side 1.5 cm, with the area 2.25 cm^2. We then end up with Figure 5 (see also Table 3):

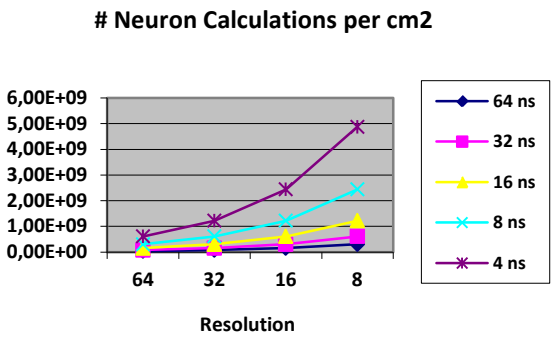

Figure 5: Neuron calculations vs resolution

Table 3: Design Space Exploration - Conservative

DRAM Access Time [ns]	Resolution (#bits per Input-weight pair)	#Neuron Calculations per second per cm^2	Area needed for entire brain [cm^2]	#Chips (2.25 cm^2)
64	64	$3.8*10^8$	$26*10^5$	$12*10^5$
	32	$7.6*10^8$	$13*10^5$	$5.8*10^5$
	16	$15.3*10^8$	$6.6*10^5$	$2.9*10^5$
	8	$30.5*10^8$	$3.3*10^5$	$1.5*10^5$
32	64	$7.6*10^8$	$13*10^5$	$5.8*10^5$
	32	$15.3*10^8$	$6.6*10^5$	$2.9*10^5$
	16	$30.5*10^8$	$3.3*10^5$	$1.5*10^5$
	8	$61*10^8$	$1.6*10^5$	$7.3*10^4$
16	64	$15.3*10^8$	$6.6*10^5$	$2.9*10^5$
	32	$30.5*10^8$	$3.3*10^5$	$1.5*10^5$
	16	$61*10^8$	$1.6*10^5$	$7.3*10^4$
	8	$122*10^8$	$8.2*10^5$	$3.6*10^4$
8	64	$30.5*10^8$	$3.3*10^5$	$1.5*10^5$
	32	$61*10^8$	$1.6*10^5$	$7.3*10^4$
	16	$122*10^8$	$8.2*10^4$	$3.6*10^4$
	8	$244*10^8$	$4.1*10^4$	$1.8*10^4$
4	64	$61*10^8$	$1.6*10^5$	$7.3*10^4$
	32	$122*10^8$	$8.2*10^4$	$3.6*10^4$
	16	$244*10^8$	$4.1*10^4$	$1.8*10^4$
	8	$488*10^8$	$2.0*10^4$	$9.1*10^3$
2	64	$122*10^8$	$8.2*10^4$	$3.6*10^4$
	32	$244*10^8$	$4.1*10^4$	$1.8*10^4$
	16	$488*10^8$	$2.0*10^4$	$9.1*10^3$
	8	$977*10^8$	$1.0*10^4$	$4.6*10^3$
1	64	$244*10^8$	$4.1*10^4$	$1.8*10^4$
	32	$488*10^8$	$2.0*10^4$	9,102
	16	$977*10^8$	$1.0*10^4$	4,551
	8	$1,950*10^8$	5,120	2,276

With a more aggressive TSV pitch values, (4um pitch), we get values that are 4 times larger in terms of calculation speed and 4 times smaller in terms of area. For 1 ns DRAM access time, we would then get the Table 4:

Table 4: Design Space Exploration

Input-Weight Resolution [bits]	64	32	16	8
# Neuron Calculations/cycle/cm^2	97.7	195.3	390.6	781.3
# Chips	4,551	2,276	1,138	569
# BPU Boards (100 chips/board)	45.6	22.8	11.4	5.7

We should also ask ourselves what the required amount of DRAM per chip would have to be, in order to be able to implement the functionality as a function of the desired number of chips. We then get the Table 5 below:

Table 5: Weight resolution vs DRAM size vs #of chips

Input-Weight Resolution [bits]		64	32	16	8	4
DRAM Size [Byte/brain]		$8*10^{15}$	$4*10^{15}$	$2*10^{15}$	10^{15}	$5*10^{14}$
#Chips	100	$8*10^{13}$	$4*10^{13}$	$2*10^{13}$	10^{13}	$5*10^{12}$
	1,000	$8*10^{12}$	$4*10^{12}$	$2*10^{12}$	10^{12}	$5*10^{11}$
	10,000	$8*10^{11}$	$4*10^{11}$	$2*10^{11}$	10^{11}	$5*10^{10}$
	100,000	$8*10^{10}$	$4*10^{10}$	$2*10^{10}$	10^{10}	$5*10^{9}$

As we can see, the implementation will be memory limited, i.e., the bottleneck will be how many memories that it will be possible to stack together on top of one single BPU die, together with the number of TSVs that we are able to use for interconnect. Solutions that we deem feasible to implement according to the ITRS roadmap [26] are marked in gray.

5.1 BPU IMPLEMENTATION

The second question we have to ask ourselves is how many BPUs that are needed per chip in order to implement the functionality listed in the previous section. In one single BPU cycle, all bits of input-weights of a single neuron calculation are fetched from memory. It means that for one calculation, $1000 * precision * resolution\ [bits]$ should be fetched from memory. The way the computation is implemented is of no concern, i.e., it does not matter if the calculation is performed in a single calculation unit fetching all bits of a single neuron at once, or if there are 1000 BPU calculations going on in parallel, each calculation fetching one input-weight pair in parallel with the others. The amount of bits being fetched from the memory is the same.

In order to determine the amount of cycles needed to perform one BPU calculation, we should implement a single neuron calculation, together with a NoC connecting each local calculation unit to its neighbors.

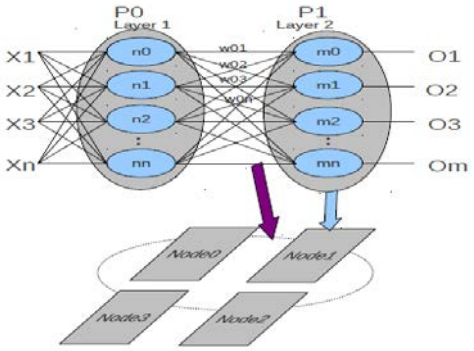

Figure 6: Mapping ANN to NOC platform

When it comes to how to implement the actual calculations, there are several models to choose from. We have selected the multi-layer perceptron model [8] as the ANN model for our proposed architecture. In this model, the neurons are grouped into mini-columns consisting of 100 neurons each and each mini-column is connected to another mini-column in a serial fashion (the number of connections are 10 times less, but the number of neurons are 10 times more, so the total number of calculations is the same). A packet-switched NOC is selected to connect the BPUs together. The nodes calculate the new output value, based on the inputs provided through the NoC. In case the new value differs from the present output value, it should be broadcast to the target nodes, and then stored locally at the target node together with that input's associated weight, so that the node can retrieve the value together with the input in one single cycle.

5.1.1 Vivado HLS

High level synthesis is one of the promising technological innovations where we can directly synthesize behavioral models from specification leaving low level optimizations to the automated tools.

Vivado is a state-of-the art High-level synthesis tool, integrated with the Xilinx tool set. It is used to synthesize each layer of ANN using its behavioral model in C++. The main advantage here is that it is possible to simulate the model at high-level while still leaving the low level optimizations to the tool. The method speeds up both simulation and synthesis.

The basic proposed ANN model is shown in Figure 6, and the proposed synthesis methodology is described in Figure 7. Each mini-column is performed in sequence over a number of cycles. The exact number of cycles is to be determined by this investigation.

5.1.2 NOC Generator

The NoC generator is a state-of-the art tool for generating network-on-chip systems on FPGAs. The scalable multi-core NOC platform is generated from an XML configuration file[27]. The basic configuration includes target technology, NOC topology, kind of router and interconnection scheme. A new design can be generated within a couple of seconds by changing the parameters. The tool automatically creates a multicore implementation and places all SW at respective nodes, thereby reducing time for design-space exploration. The code is then compiled towards the selected FPGA and target FPGA vendor tools are run to have the system implemented.

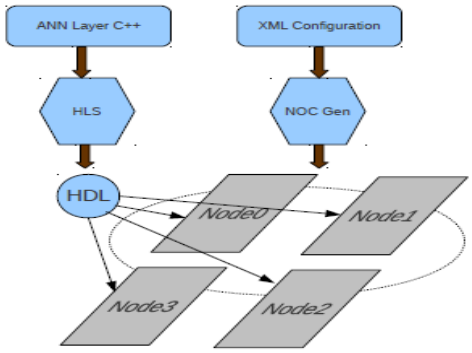

Figure 7: Design flow

6. EXPERIMENTAL RESULTS

From the previous section, we have the maximum throughput we can ever hope to achieve. If we can produce more data than this in a single FPGA, the design is pin/TSV/memory bandwidth limited, i.e., the only thing that stops us from building it is the number of connections between the memory and the FPGA/ASIC. If not, it is area limited, and we will need more than one FPGA/ASIC chip in the stack to fulfill the functionality. Table 6 shows the values we strive for per chip as a function of the resolution

6.1 Vivado HLS

A parameterized C++ behavioral model of single layer ANN is synthesized by using the HLS tool with target device as ZedBoard having Zynq7000 all programmable SOC XC7Z020-CLG484-1 7th series FPGA. New points in the design-space are created by changing the number of neurons.

Two cases are considered. In the first case, the weights are modelled as being read from an input port. All weights are considered double precision, while the inputs/outputs are of integer (actually Boolean) types for simplicity. Other type combinations are also permissible. The results from this case are presented in Table 4 which shows that latency increases with number of neurons.

Table 6: Nr of Neuron vs Resource requirement and latency

# Neu	BRAM (18K)	DSP48E	FF	LUT	Lat (clk-cc)
1	0	14	13591	15871	616
2	0	28	13610	16251	668
4	0	28	13619	16336	768
8	0	28	13628	16421	968
10	0	28	13628	16770	1068
20	0	28	13637	16892	1568
30	0	28	13642	17043	2068
40	0	28	13646	17013	2568
50	0	28	13652	17180	3068
60	0	28	13652	17170	3568
70	0	28	13655	17254	4068
80	0	28	13655	17135	4568
90	0	28	13655	17319	5068
100	0	28	13662	17311	5568

We don't need to go beyond 100,since that is the size of a minicolumn. We conclude from the experiments that the size of a calculation molecule is around 17300 LUTs when I/Os are sixteen bit integers and weights are double precision floating point.

Although we did a serial pipeline implementation, we should point out here that for the sake of this exploration, the result is irrespective of if we chose to build a parallel solution or not. Either we make a gigantic N-input calculation molecule to produce one data every clock cycle, or we make N serial calculation molecules produce one result every N clock cycles. The area of the two solutions will be approximately the same. However, the parallel solution can be pipelined to use a faster clock cycle.

In the second case the weights are modelled to be synthesized as on chip BRAM. This experiment gives us a sense of how the memory demand increase with increasing number of neurons, see Figure 8.

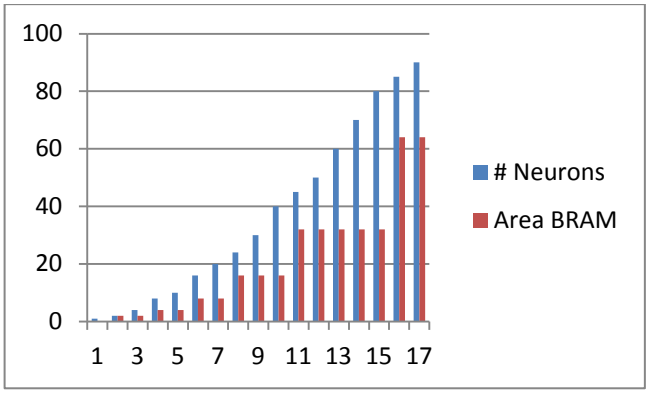

Figure 8: No. of neurons vs Area

The area results and critical path of the design are of course highly dependent on the chosen data type. A design space exploration of the size of the calculation molecule with respect to the data type is shown in Table 5 below.

Table 7: Weight Data type dependencies

Weight Data Type	Area [LUT]	Delay [clk-cc]
64-bit Double Serial	13807	5563
64-bit Integer Serial	15862	5005
64-bit Integer fully parallel	64000[1]	1
32-bit Single Serial	7201	5563
32-bit Integer Serial	8957	5004
16-bit Integer Serial	5555	5004
8-bit Integer Serial	3504	5004
4-bit Integer Serial	2652	5003

From the experiments, we can deduce that we need 5,568 clock cycles to calculate 100 Neuron outputs, consuming 100 times 100 input-weight pairs. If we do this in a serial fashion, adding two input-weights per clock cycle, using a 100 MHz clock, this means

[1] Estimate based on our structure as the HLS tool was unable to synthesize it.

that it takes 5,568 us to calculate 100 NeuronCalculation (actually, one minicolumn in Lanser's terminology[8]), or 17,960 NeuronCalculations per second (179.6 minicolumns/s) per BPU.Considering the real time constraint of each neuron firing ~100 times/sec it means performance of 179.6 neurons per BPU.

6.2 NOC Generator results

The NoC has been synthesized separately from the ANN to obtain scalability metric. The results indicate that a NoC node consumes around 750 LUTs per node. Thus, a BPU Node using double precision floating point operations will be around 18,000 LUTs large.

6.3 Discussions

From Table 2, we get the maximum possible delivery data rate. However, what also need to know the maximum possible consumption rate. From Table 7 we can derive that value. It is shown in Table 8, below.

The fully parallel, fully pipelined designs using integer data types are the only designs that come close to the required throughput, so in the next analysis, we will disregard all serial and floating point solutions.

From Table 7, we see that the only BPU+NoC node that is fast enough is the fully pipelined 64-bit integer solution that is ~65000 LUTs large. The largest FPGA today contains roughly 800,000-1,000,000 LUTs. This means that we can fit ~15 BPUs on an FPGA of that size, compared to the 54.9, which are the conservative TSV limit through which we can deliver data, i.e., we can at maximum consume 27% of the data that we can ever hope to deliver.

Thus, we can conclude that the design will be area limited, and not TSV pin limited, and that we will require 3,66 times (1/fraction) more chips than what we got when we did our design space exploration in Section 4. The results are summarized in Table 8 below.

Table 8: Weight resolution vs Neuron calculations (conservative TSV pitch, and integer data types)

Input-Weight Resolution [bits]	64	32	16	8
Production rate #Neuron Calculations/cycle/chip	54.9	109	220	439,4
Consumption rate #Neuron Calculations/cycle/chip	15	31	62	125
Fraction of required rate	27%	28%	28%	28%

From Table 3, we get the most feasible design, arranging the DRAM to produce one set of weights every 10 ns. The data is reproduced in Table 9, below, with the # chips column compensated for the area limitation factor of 3.66.

The two most feasible designs we have marked in gray. It will need at minimum 66,000 chips if we use 8-bit resolution for the weights. Each chip will have a memory of 1.5 GByte

Table 9: Most feasible design

DRAM Access Time [ns]	**Resolution (#bits per Input-weight pair)**	**#Chips (2.25 cm^2)**	**DRAM Size (Bytes per chip)**
8	64	$5.5*10^5$	$1,5*10^9$
	32	$2.7*10^5$	
	16	$1.3*10^5$	
	8	$6.6*10^4$	

.

7. CONCLUSION

We have performed a design space exploration of the feasibility of a digital implementation of the human brain cortex using basic FPGA technology for implementing the neuron calculations. We can conclude that it should be possible to build it using a memory stack of DRAMs for weight storage, using TSVs for connecting the DRAM to the largest available FPGA chips of today.

We can also conclude that the design will be area limited for the FPGA solution. With maximum TSV density, we should be able to deliver about 3.66 times more data than we can ever hope to consume in the FPGA.

We have shown that a modern FPGA is sufficient to implement the functionality, if it is complemented with external DRAMs connected through TSVs, to be able to access the input-weight pairs in parallel as fast as possible. The resulting brain-size FPGA design will require ~66,000 chips, where each chip consists of a 1,000,000 LUT FPGA with a DRAM memory stack of 1.5 GByte.

8. FUTURE WORK

The current ANN is mainly based on the integrate and fire model of the neuron but in the future we will work on more types of neuron models. The final goal is to have an automated tool which will accept any type of high level neural model, synthesize it after design-space exploration, and then map it onto a multi-board FPGA platform. We also aim to build a specialized NOC for communication in ANN model to speed-up the emulation of such ANN models. Ultimately a small scale ANN model will be built.

9. REFERENCES

[1] W.S. McCulloch, W. Pitts, ''A logical calculus of the ideas immanent in nervous activity,'' BullMath. Biophy. 5 (1943) 15–133.

[2] Dan W. Patterson. Artificial Neural Networks: Theory and Applications (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[3] National Academy of Engineering(nae.edu), Grand Challenges for Engineering, www.engineeringchallenges.org, 2010

[4] D. Modha, et al, "Cognitive Computing: Unite neuroscience, supercomputing, and nanotechnology to discover, demonstrate, and deliver the brain's core algorithms," Communications of the ACM 54, 8, pp 62-71, August 2011.

[5] J. Schemmel et al, "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neuron Modeling", in Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, 2010.

[6] 1st Heterogeneous 3d ICs based on SSI Technology: http://www.eetimes.com/document.asp?doc_id=1319270

[7] Saban, Kirk. "Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency." *Xilinx White paper: Vertex-7 FPGAs*, 2010.

[8] IA. Parker et al, "Towards a Nanoscale Artificial Cortex", International Conference on Computing in Nanotechnology,, Las Vegas, USA, June 26, 2006.

[9] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, DC: Spartan Press, 1961.

[10] J. Schemmel et al, "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neuron Modeling", in Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, 2010..

[11] Rumelhart , D.E .and McClelland, J.L ,1986, Parallel distributed processing: *Exploration in the Microstructure of Cognition* (Vol1 &2), Cambridge, MA:MIT Press.

[12] Hopfield,J.J., 1982, Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of National Academy of Science*, USA,79, 2554-2558 .

[13] Kohonen, T,1984, *Selforganization and Association Memory*, Springer-Verlag.

[14] "The Spinnaker Project Website," http://apt.cs.man.ac.uk/projects/SpiNNaker/, 2007.

[15] H. Markram, "The Blue Brain Project", in Nature Reviews, Volume 7, February 2006, pp 153-160.

[16] P. Merolla, J. Arthur, F. Akopyan, I. Nabil, R. Manohar, D. Modha: "A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm", IEEE Custom Integrated Circuits Conference (CICC), San Jose, 2011

[17] "The Neurogrid Website," http://www.stanford.edu/group/ brainsinsilicon/neurogrid.html, 2009.

[18] R. Jacob Vogelstein et al, "A Multichip Neuromorphic System for Spike-Based Visual Information Processing", In *Neural computation*, Volume 19, 2007, pp 2281-2300.

[19] J. Park, T. Yu, C. Maier, S. Joshi, G. Cauwenberghs, "Hierarchical Address-Event Routing Architecture for Reconfigurable Large Scale Neuromorphic Systems," *IEEE International Symposium on Circuits and Systems*, Rio de Janiero, May 2011.

[20] T. Yu and G. Cauwenberghs, "Analog VLSI Biophysical Neurons and Synapses with Programmable Membrane Channel Kinetics", *IEEE Transactions on Biomedical circuits and Systems*, 4, 3, p 139-148,June 2010.

[21] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, and S. Cawley, "A Reconfigurable and Biologically Inspired Paradigm for Computation Using Network-on-Chip and Spiking Neural Networks," Int'l J. Reconfigurable Computing, vol. 2009, pp. 1-13, 2009

[22] Carrillo, S.; Harkin, J.; McDaid, L.J.; Morgan, F.; Pande, S.; Cawley, S.; McGinley, B. "Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations", *Parallel and Distributed Systems, IEEE Transactions on*, On page(s): 2451 - 2461 Volume: 24, Issue: 12, Dec. 2013.

[23] Wulf, Wm.A and McKee, S.A. Hitting the Memory Wall: Implications of the Obvious.ACM Computer Architecture News. Vol.23, No.1 March 1995.

[24] Wilkes, M.V., The Memory Wall and the CMOS End-Point, ACM Computer Architecture News. Vol. 23, No. 4 September 1995.

[25] G. Loh.3D-Stacked Memory Architectures for Multi-core Processors. *In Proceedings of the International Symposium on Computer Architecture*,2008.

[26] International Technology Roadmap for Semiconductors http://www.itrs.net/Links/2012ITRS.

[27] J. Öberg, F. Robino, "A NoC Generator for the Sea-of-Cores Era". In proc. of FPGA World 2011. ACM Digital Libraries.

[28] S. Furber, S. Temple, and A. Brown, "On-chip and inter-chip networks for modelling large-scale neural systems," in Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece, May 2006.

[29] S. Gao, A. G. Schmidt, and R. Sass, "Hardware implementation of MPI Barrier on an FPGA cluster," in Proc. of International Conference on Field Programmable Logic and Applications, 2009, pp. 12–17

.