1-1-2011

A low level analysis of Cellular Automata and Random Boolean Networks as a computational architecture

Prateen Reddy Damera
Portland State University

A low level analysis of Cellular Automata and Random Boolean Networks as a computational architecture

by

Prateen Reddy Damera

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
Christof Teuscher, Chair
Mark Faust
Douglas Hall

Portland State University
2011

Abstract

With the transition from single-core to multi-core computing and CMOS technology reaching its physical limits, new computing architectures which are scalable, robust, and low-power are required. A promising alternative to conventional computing architectures are Cellular Automata (CA) networks and Random Boolean Networks (RBN), where simple computational nodes combine to form a network that is capable of performing a larger computational task.

It has previously been shown that RBNs can offer superior characteristics over mesh networks in terms of robustness, information processing capabilities, and manufacturing costs while the locally connected computing elements of a CA network provide better scalability and low average interconnect length. This study presents a low level hardware analysis of these architectures using a framework which generates the HDL code and netlist of these networks for various network parameters. The HDL code and netlists are then used to simulate these new computing architectures to estimate the latency, area and power consumed when implemented on silicon and performing a pre-determined computation.

We show that for RBNs, information processing is faster compared to a CA network, but CA networks are found to a have lower and better distribution of power dissipation than RBNs because of their regular structure. A well-established task to determine the latency of operation for these architectures is presented for a good understanding of the effect of non-local connections in a network. Programming the nodes for this purpose is done externally using a novel self-configuration algorithm requiring minimal hardware. Configuration for RBNs is done by sending in configuration packets through a randomly chosen node. Logic for identifying

the topology for the network is implemented for the nodes in the RBN network to enable compilers to analyze and generate the configuration bit stream for that network. On the other hand, the configuration of the CA network is done by passing in configuration data through the inputs on one of the sides of the cell array and shifting it into the network.

A study of the overhead of the network configuration and topology identification mechanisms are presented. An analysis of small-world networks in terms of interconnect power and information propagation capability has been presented. It has been shown that small-world networks, whose randomness lies between that of completely regular and completely irregular networks, are realistic while providing good information propagation capability. This study provides valuable information to help designers make decisions for various performance parameters for both RBN and CA networks, and thus to find the best design for the application under consideration.

Acknowledgements

I would like to thank my academic and thesis advisor Prof. Cristof Teuscher and Teuscher Lab for the opportunity and constant guidance throughout my course of study at Portland State University.

I would like to thank Prof. Mark Faust and Prof. Douglas Hall for their time and invaluable guidance.

I am grateful to the faculty and staff of the Electrical and Computer Engineering department at Portland State University for their support and resources. I would also like to thank my Teuscher lab mates for their help in various parts of my Thesis.

Finally, I would like to thank my family and friends for their support and motivation.

Contents

List of Figures

1

Introduction

1.1 Motivation

Integration levels for nano-scale electronic devices are predicted to be 10^{10} to 10^{11} devices per cm^2 according to [1], but physical limitations of CMOS architectures, such as high failure rates, extreme variations in the manufacturing process, and quantum and thermal errors arising due to the small dimensions limit the evolution of current architectures for general purpose computation. Due to these physical limitations of CMOS technology, new computing architectures, which are capable of general purpose computation while being easily scalable, robust, and low-power are desired. Architectures that are formed by self-assembly overcome these limitations by embracing the variations that occur during the manufacturing process. Architectures, which use small computing elements capable of performing simple logic, connected together in the form of a mesh to perform larger computations were shown to be promising because of their inherent scalability, robustness, and well-distributed power consumption [20]. A more general case of such networks of computing nodes leads us to random networks, where the computing cells of the network may be at any spatial location and the connections can be established without constraints.

Some challenges that need to be addressed with these types of networks are the configuration of the computing cells, the design of a compiler that is capable of identifying the network and establishing functions for each of the computing cells

used for the larger computation, and optimizing the networks to utilize minimum resources for computation.

Zhirnov et al. [20] present the analysis of cellular automata (CA) as a general purpose computing architecture because of its advantages in scalability, robustness, and lower and better power distribution across the chip due to its regular structure. Cellular automata have the advantage of short cell-to-cell interconnects, thus reducing the power consumption. Cellular automata are also found to have lower energy dissipation from the interconnects within the cell due to their low cell complexity. Their study, however, did not provide insight into how many cells are required in the cellular automata for comparison with current von Neumann architectures for general purpose computation and the design of the compiler that would be required to configure cellular automata for general computation. Their study also did not consider future non-CMOS technologies that can be used to implement cellular automata.

A more generalized analysis of architectures that utilize a large number of low complexity computing elements connected together in a network leads us to random Boolean networks (RBNs). The operation of these networks is similar to cellular automata, but the arrangement of the cells may not be regular as in cellular automata. A hardware level evaluation of this general class of computing architecture will provide insight into the advantages and issues with using these types of architectures for general purpose computing. The ability to determine the area occupied by these networks and the interconnect complexity within each cell will help in determining the advantages and disadvantages of manufacturing and utilizing these networks for computation. An estimate of the power consumption of

these networks at the micro and macro level will allow for comparison with cellular automata and current general purpose architectures.

1.2 Challenges

A major challenge in modeling these computing architectures is the lack of understanding of the variance of area and power consumption when implemented in CMOS. Although an assessment of the area and power consumption of cellular automata has been done in [20], it is based on a rough, high-level estimation of the number of elements that are required to perform the required computation. A more realistic understanding of the performance of these architectures can be achieved using hardware models. Generating hardware models and estimating the area and power consumption of these architectures can be done using well-established tools that have traditionally been used for CMOS devices.

Another major challenge is the ability to configure the networks using minimal hardware resources. Zhirnov et al. [5] present the implementation of a totalistic CA cell that uses simple elements, such as registers and multiplexers for configuration and computation. Because of the random topology of random networks, information about neighboring nodes cannot be determined without having a global view of the system. This provides a different challenge in terms of configuration of the network. For similar reasons, compilers will require information about the topology and the node functionalities of the network for generating configuration bit streams.

1.3 My contributions

The following presents my list of contributions that were made in modeling and analyzing these computing architectures.

- Design of the identification algorithm for the RBN network to allow the compiler to identify the topology of the network and generate configuration packets. The algorithm for the identification of the topology of the network is described in section 3.3.2.

- Design of the configuration algorithm for the RBN network using a packet-based mechanism. The algorithm for the configuration of RBN nodes is described in section 3.3.3.

- Implementation of the CA node. The CA node was modeled in HDL and synthesized using traditional CMOS technology. The design of the CA node is described in section 3.2.1.

- Design of the RBN node. The RBN node was modeled in HDL and synthesized for traditional CMOS technology. The RBN node consists of logic to support the identification algorithm in section 3.3.2, the configuration algorithm in section 3.3.3 and the normal RBN operation described in section 3.3.4

- Implementation of a toolbox that generates the HDL for random networks with specified size and connectivity. The description of the algorithm is presented in section 4.2.

- Implementation of a toolbox that generates small-world networks. A description of small-world networks is presented in section 2.3 and the algorithm for

the generation of the small-world networks is presented in section 4.5.

- Implementation of an evaluation framework to evaluate area and power consumption of the networks generated above. The tools and methodology by which the power and area consumption is estimated are presented in sections 4.3 and 4.4 respectively.

- Implementation of an evaluation framework to evaluate information processing capability of the networks using the density task as a benchmark is presented in section 2.4.

Chapter 5 presents the experiments conducted and the findings using the above toolboxes and evaluation frameworks for a hardware level analysis of CA and RBN architectures. Part of this thesis was presented at the Sigma Xi annual Student Research Symposium 2011 [4]. The power and area evaluation framework developed as part of this thesis was used for a study that evaluates the performance of random architectures as a reconfigurable fabric and has been published at the IEEE nano conference 2011 [2].

2

Background

2.1 Random boolean networks

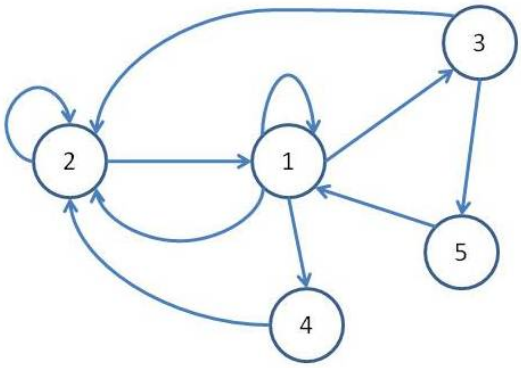
A random Boolean network [8] or RBN is the random arrangement of nodes or cells connected using randomly distributed links. Each cell consists of a binary look-up-table (LUT), which stores the output of the cell for every combination of its inputs. At every execution time step, the cell sends out the output corresponding to the combination of inputs at the previous time step to all of its connected nodes. The output of the cell is considered as the state of the cell at that particular time step.

RBNs were chosen as a representation of random networks for this study because of their simplicity. Also, the results obtained in this study can easily be extended to larger networks with more complex cells. It has been shown [7, 10, 12, 13] that RBNs have the computational capacity to perform complex tasks using simple local rules, which further motivates the study of RBNs as a computing architecture.

Network parameters: The following are the RBN design parameters considered for this study:

1. Number of cells in the network (N).

2. Average connectivity (k), which is the average number of inputs to a node, also called the fan-in.



(a) Example random Boolean network.

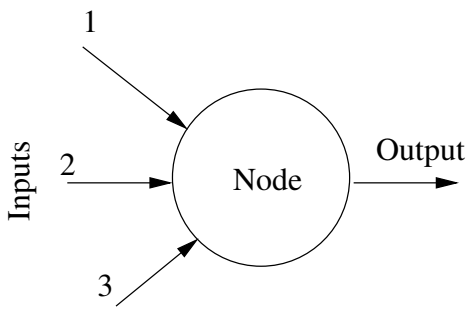
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Adjacency matrix for the example network.

Figure 2.1: Example random Boolean network with N=5 and k=2.

Network representation: A useful method of representing RBNs is through adjacency matrices, where a '1' represents the existence of a link between the nodes represented by the row and column of the matrix. Figure 2.1(a) shows an example RBN with $N = 5$ and $k = 2$ (average). The corresponding adjacency matrix is shown in 2.1(b). The transfer function of each node in the network is a look up table which consists of an output for each combination or a set of combinations of the inputs to the node. The output of the node at the next time step is the value in the look up table corresponding to the combination of inputs at the current time step. An example transfer function is shown in figure 2.2.

Since the cells of a random networks are distributed randomly in space, a cell will not know which other cells it is connected to unless explicit logic for the same is implemented. This complicates the ability to configure random networks. On the other hand, random networks are a result of the lack of control over the manufacturing process and hence predicted to be the future of such computing architectures [16].



Input combination Inputs 3,2,1	Output
0,0,0	0
0,0,1	1
0,1,0	1
0,1,1	1
1,0,0	0
1,0,1	0
1,1,0	1
1,1,1	0

Figure 2.2: Example transfer function for a random Boolean network.

2.2 Cellular automata

Cellular Automata (CA) [17, 19] networks are a special case of random networks where the cells are arranged in the form of a regular mesh of any dimension and the average connectivity k is the same for all cells of the network. The operation of a CA cell is the same as that of the RBN. Since, the number of inputs to a cell is constant for all cells, the cell complexity for all cells in the networks is identical.

Processing elements that are closely connected to each other in the form a lattice are predicted to be the future of parallel computation [3]. The configuration of CA networks is easier than for random networks because the spatial location of the cells are known. Cellular automata is a representative of the mesh architecture, where the processing element is a simple cell that outputs stored values for different input combinations. It has been shown that cellular automata networks enjoy the advantage of short interconnect length and distributed power dissipation [20]. With CMOS reaching its physical limitations, future technologies are hypothesized to

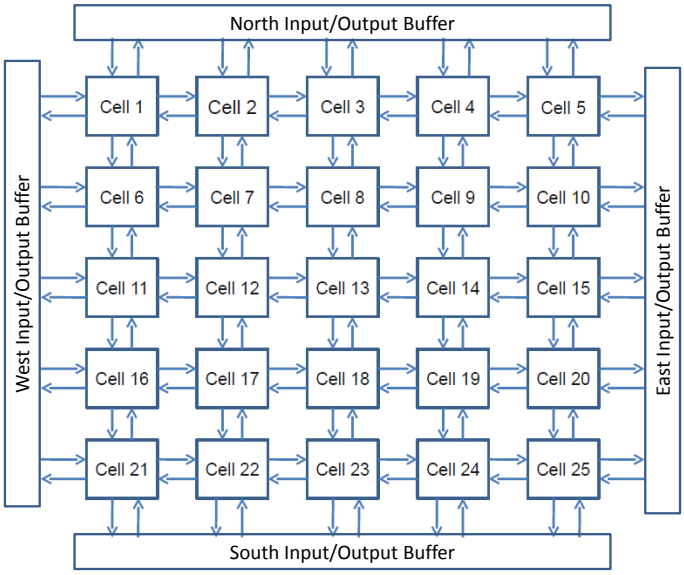


Figure 2.3: Example Cellular Automata network with $N = 25$ and $k = 4$.

have irregular structures because of the lack of precise control. Also, random networks are theoretically shown to exhibit higher information processing capability and better robustness [16] than CAs.

2.3 Small-world networks

Small-world networks [18] are formed when networks in which the nodes are placed regularly are rewired to introduce varying degrees of randomness. This makes the networks highly clustered, as in a regular network but, it can communicate faster due to the long range links, and hence has a shorter average path between the nodes. For example, consider nodes arranged in the form of a ring, as shown in figure 2.4, with N nodes and k links per node.

In the Watts-Strogatz model [18], each link is rewired to a randomly selected node with a probability of p such that at $p = 0$, a network of regularly connected nodes is

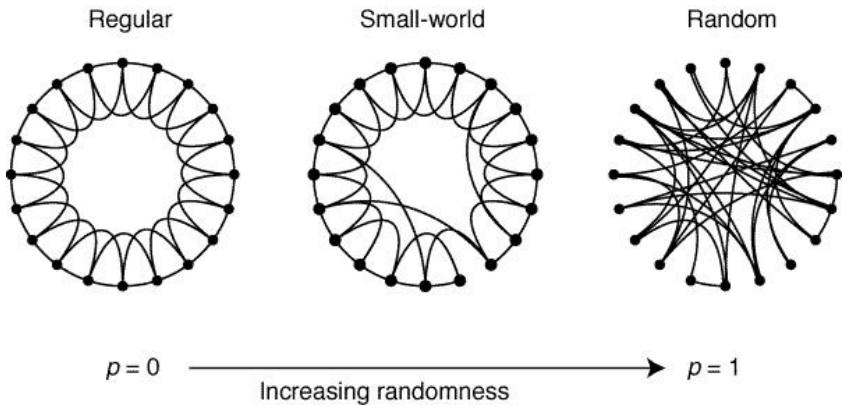
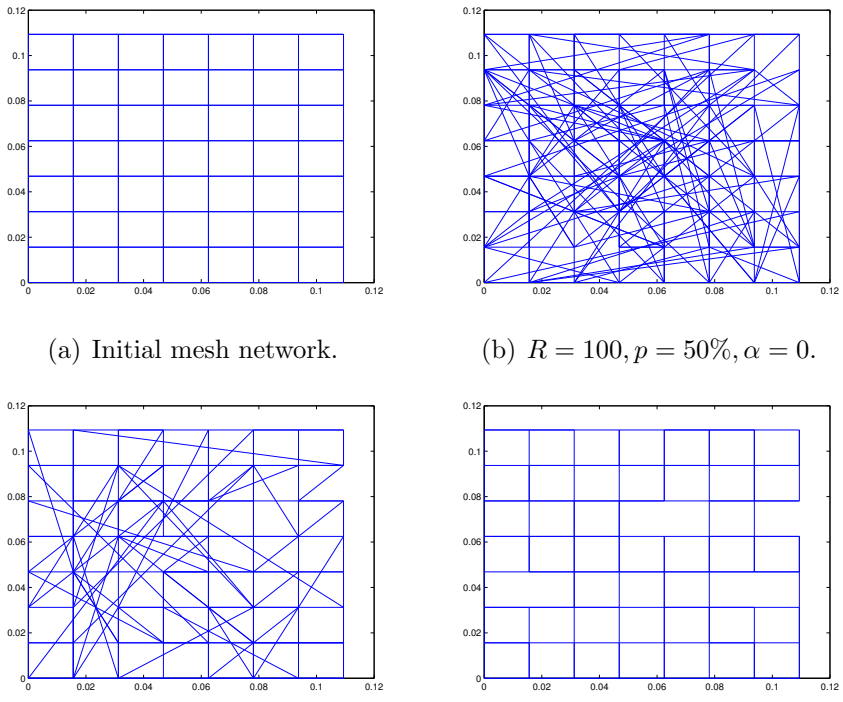


Figure 2.4: Example small-world network obtained by rewiring links from a regular ring network with a probability p. Source: [18].

formed and when $p = 1$, a randomly connected network is formed. This introduces a group of networks whose degree of randomness of the connections are between that of a regular network and a random network. This can be applied to a cellular automata network where the nodes are arranged in the form of a grid and the short node-to-node connections can be rewired with a probability p to form new links with the same or longer length. The longer connections contribute to a shorter average path between the nodes, and hence to a better information processing capability.

Kleinberg [9] and Petermann et al. [11] present a set of small-world networks that are characterized by their clustering exponent α. An initial mesh network is considered and long range links are added between the nodes of the network as a function of the clustering coefficient. It has been found that there exists a unique value for α ($\alpha = 2$) at which there is a very short path between two nodes. The links added to the mesh networks follow the power law distribution for their length given by $q(l) \sim l^{-\alpha}$.

The present thesis uses a combination of these types of small-world networks. An initial mesh network is considered with links only to its neighboring nodes. R extra local links are added the network to ensure that the network remains connected after the rewiring procedure. Each link in the network is then rewired with a probability p and rewired to a randomly selected node in the network with a length distribution determined by the power law function $l^{-\alpha}$. This generates a comprehensive set of small-world networks which can be controlled by the three parameters, R, p, and α. Figure 2.5. The set of networks consist of completely local networks for large values of α and random networks for $\alpha = 0$. A study of these networks will contribute to a more comprehensive understanding of using CAs or RBNs for computation.



(a) Initial mesh network.

(b) $R = 100, p = 50\%, \alpha = 0$.

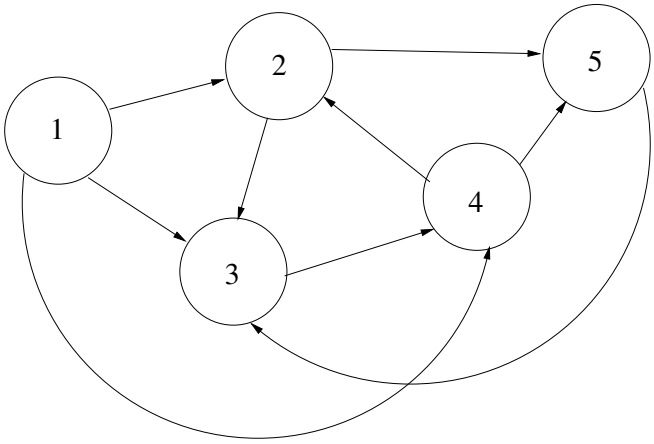
(c) $R = 100, p = 50\%, \alpha = 1$.

(d) $R = 100, p = 50\%, \alpha = 3$.

Figure 2.5: Example small-world networks from the set of networks used for this study.

2.4 Density task as a performance evaluation benchmark

The density classification task [10] is a common application used in cellular automata. The goal of the task is to determine using local rules only if a randomly initialized automaton consists of more than 50% nodes that were initialized to state 1. The state of the node is the output of the node at the time of initialization. The network is said to solve the task successfully, if all of the nodes of the network eventually reach state 0 or state 1 when the majority of the nodes during initialization have a state 0 or state 1, respectively. This can be achieved using local rules for the nodes. The rule for the current state of each node was found to be the majority function of the inputs that the node receives. We use this task in this study as a benchmark to compare the information processing capabilities of cellular automata and random Boolean networks. Figure 2.6 shows an example network and a time state diagram showing the states of the nodes when it attempts to solve the density task for an initial set of randomly assigned states.



Time State Diagram

	Nodes				
Time	1	2	3	4	5
0	1	1	1	1	0
1	0	1	1	1	1
2	0	0	1	0	1
3	0	0	0	0	0

Figure 2.6: Example network and its time state diagram with random initial states assigned to the nodes at time step 0. The expected result is that all the nodes have the state 1, but it can be seen that all the nodes end up with state 0 and hence the network is considered to fail the density task for this set of initial states.

2.5 Previous work

A study of the information processing capabilities of cellular automata architectures is presented in [20]. This study argues that the future of multi-core computing lies in regular arrays of locally connected computational elements. It has been theoretically shown that cellular automata cells require a minimum cell complexity of a few hundred binary switches to provide information processing capabilities comparable to a general purpose processor. A true CA implementation was found to require around 2,300 elements for a 256 rule CA, which is the same as the Intel 4004 microprocessor. To reduce the size of the CA, a smaller set of rules were suggested. For example, using the sum of the inputs to determine the output of the cell [5] will significantly reduce the number of look up table entries required in the cell. This implementation is called totalistic cellular automaton and is shown in figure 2.7.

Since the internal interconnect distribution of a CA node is similar to a general purpose processor, the interconnect distribution density of the cells is expected to be similar to that of a general purpose processor. But CAs have the advantage of short cell-to-cell interconnects, thus reducing power dissipation due to interconnects. Any additional global interconnects for configuration or control will reduce the benefit of using CA architectures. The study also discusses that CAs require a large average connectivity to be significantly fault tolerant. Cellular automata also enjoy well-distributed power dissipation at the macro level and reducing thermal hot spots on the chip.

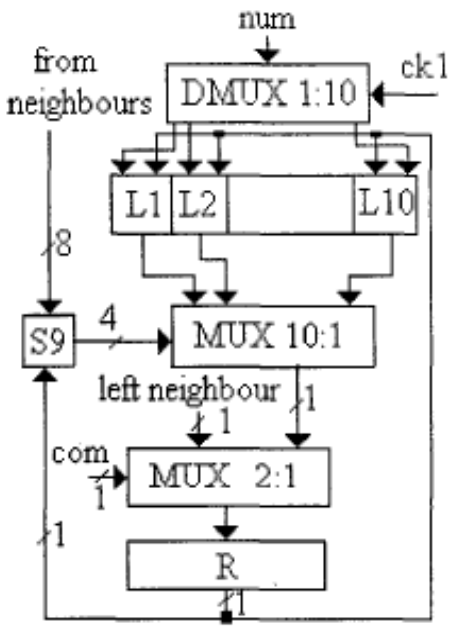


Figure 2.7: Implementation of the totalistic cellular automaton adapted from [5].

A contrasting study [16] argues that random networks are a more promising candidate for future computing architectures because of their resistance to errors in systems with low reliability, task solving efficiency, and manufacturing issues. This study shows that there exists an average connectivity for random networks at which damage spreading due to the perturbation of a node in the network remains constant and is independent of the number of nodes in the network, whereas, CAs do not possess such a characteristic average connectivity, therefore making them more susceptible to errors. The study also presents manufacturing issues, which may prevent CAs from being economically manufactured at the nano-scale. Self-assembly is considered to be an easy and inexpensive method to form random arrays of wires with randomly attached switches. A hybrid assembly, where the cells are constructed using traditional silicon and the interconnects are self-assembled

nanowires, is considered to be a good first step with currently available technologies. Random networks are also capable solving global density classification and synchronization tasks more efficiently compared to CAs [10]. These tasks are well-known benchmarks for the performance of CAs. This was attributed to the slow information propagation capability of CA networks.

3

Design Methodology

3.1 Network characteristics

Cellular automata and random Boolean networks consist of the following elements:

- Nodes
- Interconnect
- Network topology

Nodes are the computing elements of the network responsible for analyzing and generating data. The interconnect is the medium by which the nodes communicate with each other and with the outside world. The network topology is the arrangement of nodes and interconnect in space. It has been shown [6] that an economic first step towards nanoscale electronics is to have a hybrid structure consisting of nanowires for the interconnect and traditional CMOS implementation for the nodes of the network.

For this study, the nodes are implemented using CMOS technology and their area and power consumption were estimated using well-established tools and methods such as Mentor Graphics ModelSim and Synopsys Design Compiler [15].

3.2 CA implementation

3.2.1 CA node implementation

Cellular automata are a network of interconnected nodes where the nodes are arranged in a form of a grid and each node is connected to its four immediate neighbors to its north, south, east and west directions thus forming a mesh network. Each node consists of inputs, outputs, a look-up-table, and logic to place data into the look-up-table. The output of the node at the current time step is the corresponding value from its look-up-table for the combinations of inputs at the previous time step. The block diagram for the above logic is as shown in figure 3.1. Each node in the network is eventually configured to have the same data in their look-up-table through systematic configuration. This type of CA is called a uniform CA.

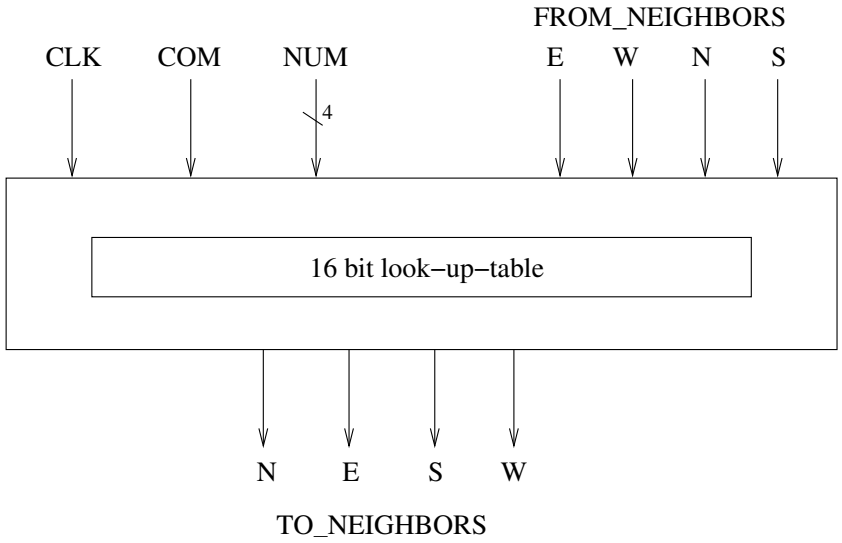


Figure 3.1: Block diagram of the cellular automata node used for this study.

Signal descriptions:

- CLK: An external common clock signal is provided to all the nodes of the network to maintain synchronization.

- COM: This signal when high, indicates configuration mode and the algorithm described in section 3.2.2 is used to treat data from neighbors as data to be placed in the look up table of the node.

- NUM: 4-bit signal indicating the location at which the data from neighbors has to be placed in the look-up-table during configuration. This signal is ignored during the normal operation of the node.

- FROM_NEIGHBORS: 4-bit input from the four neighbors of the node, which is the 1-bit output from each of the neighbors.

- TO_NEIGHBORS: 4-bit output to the four neighbors of the node which is the 1-bit input to each of the neighbors.

As seen from the block diagram for the cellular automata node, the construction for the node is simple, consisting of a few simple logic elements only. This node is a generalized implementation of the totalistic cellular automata node presented in [5] where all of the input combinations have an entry in the look-up-table as opposed to having an entry for all possible sums of all inputs. This implementation was chosen to represent a general class of cellular automata networks, which have all the capabilities of a theoretical CA network. The nodes are arranged as shown in figure 3.2. The nodes at the edges of the grid have an input and output each that is connected to an external interface for configuration, sending inputs and capturing outputs.

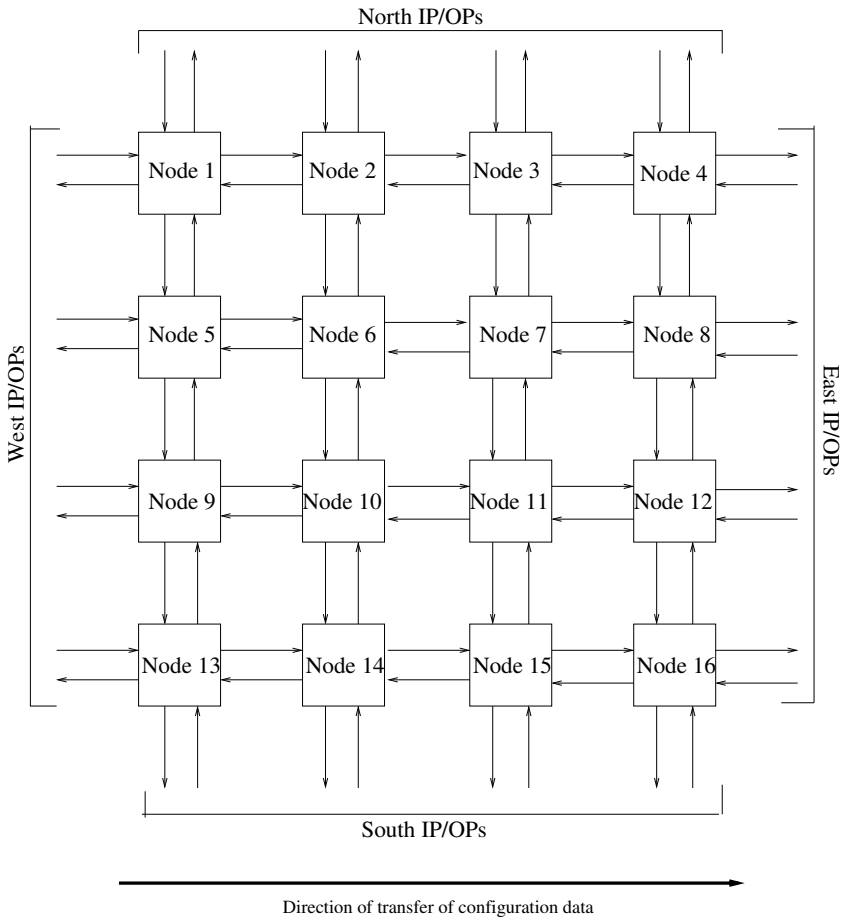


Figure 3.2: Block diagram of a cellular automata network with 16 nodes arranged as a $4 * 4$ grid. The direction of transfer of configuration data is also shown.

3.2.2 Operation of the CA network

Configuration of the CA network is done similar to the method presented in [5], i.e., by shifting in the data that is needed to be placed into the look-up-table from one of the edges of the grid through the entire network. The node differentiates between configuration and operation modes by monitoring the COM signal which is common to all the nodes. The algorithm for the configuration and operation of the network is described in algorithm 1:

Algorithm 1 Algorithm for the configuration and operation of a CA node.

```
if Rising edge of clock then
  if COM is asserted then
    TO_NEIGHBORS(ALL) = FROM_NEIGHBORS(W)
    LUT address = NUM
    data at LUT address = FROM_NEIGHBORS(W)
  end if
  if COM is not asserted then
    LUT address = FROM_NEIGHBORS(ALL)
    TO_NEIGHBORS(ALL) = data at LUT address
  end if
end if
```

For the algorithm implemented above, the nodes are configured from the west edge to the east edge of the grid by shifting in the data a column at a time at every time step. The NUM signal indicates the location in the LUT where the configuration data received from the west neighbor is to be placed. The arrow in figure 3.2 shows the direction of configuration of nodes using the above algorithm.

An important advantage with cellular automata networks is that the topology of the nodes is known before configuration. This reduces the logic required for the identification of the topology of the network. RBNs on the other hand require additional logic to support the identification of the topology of the network as described in section 3.3.1, which significantly increases the area and power consumption of each node in the network.

3.3 RBN implementation

3.3.1 RBN node implementation

The RBN node is similar to the CA node described in section 3.2.1, except that it cannot identify which other nodes of the network it is connected to. During

the configuration phase of the cellular automata network, the CA node identifies that the data it receives from its west neighbor is the configuration data that needs to be placed in its look-up-table, whereas the RBN nodes cannot determine their location with respect to the other nodes, and thus require additional logic to identify and configure the network. The logic required for that purpose can be categorized into 3 parts based on the operating modes of the node.

- Logic required for identification of the topology of the network;
- Logic required for configuration of the node; and
- Logic required for normal operation of the node.

To support the identification and configuration, a packet-based approach is considered. For this purpose, a set of input buffers and an output buffer are used to store incoming and outgoing packets. The logic for normal operation of the RBN does not use packets and hence receives inputs from its neighboring nodes directly for processing. Similarly, the output buffer is not required for normal operation of the RBN node. The clock (CLK) and reset (RST) signals are common to the entire node. The IDN signal indicates the identification mode and enables the identification logic, which treats all incoming packets as identification packets. The CNF signal indicates the configuration mode and treats all incoming packets as configuration packets. When neither IDN nor CNF is asserted, the node performs normal RBN operations.

Figure 3.3 shows the block diagram of the RBN node with n nodes from which it receives inputs, p nodes to which its output is connected to and input buffers of depth m. It can be seen that each of the parts mentioned above contribute

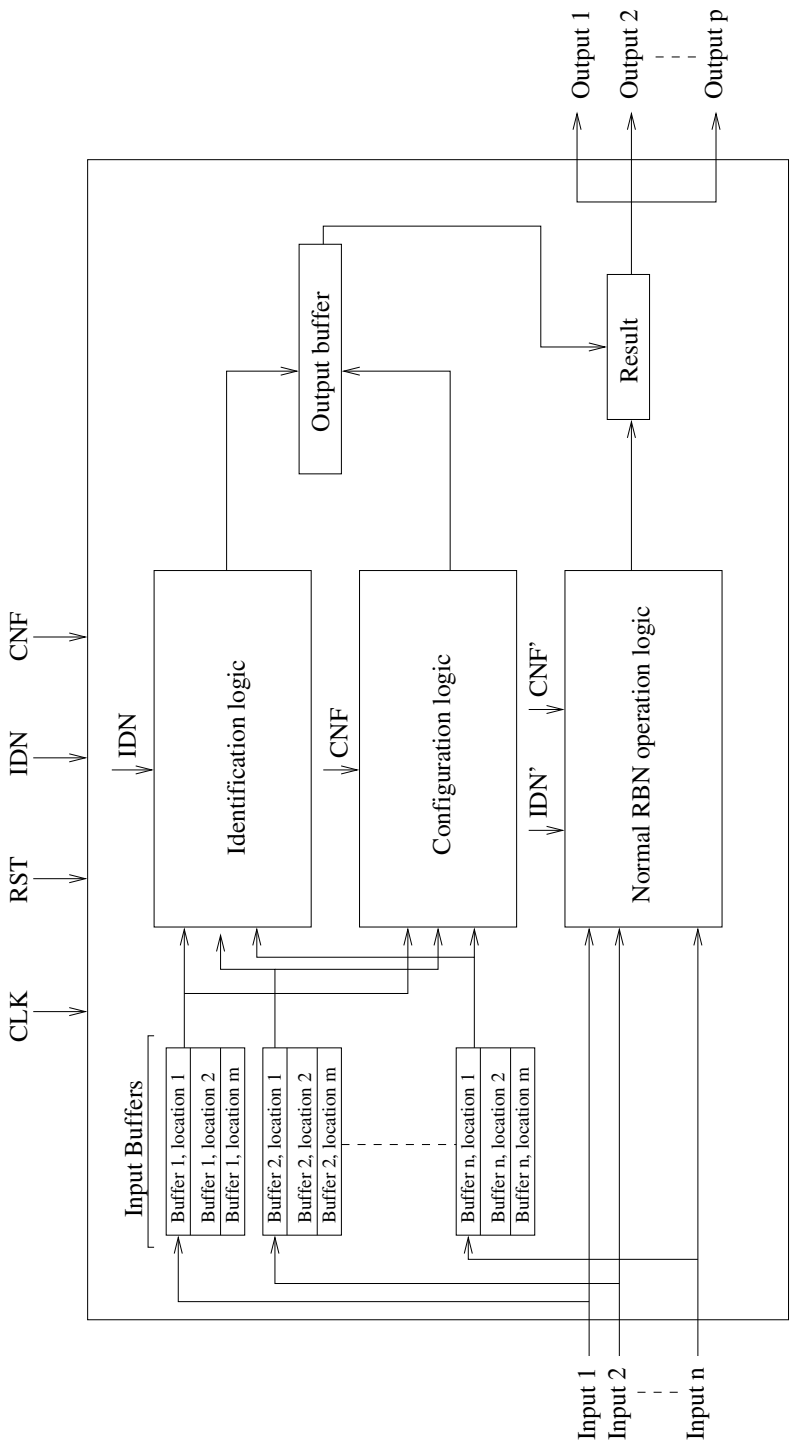


Figure 3.3: Block diagram of a RBN node with support for identification, configuration, and normal operation.

to largely independent logic. Each of these modes of operation of the node are presented in the following sections.

3.3.2 RBN node identification algorithm

Both the identification and configuration of the node are done using special packets sent by the compiler. These packets consist of the identification or configuration data along with the necessary information needed for the node to understand and process the data. Another way that RBNs differ from CAs is that RBNs have minimal external access, i.e., a single input and single output for sending data into the network and bringing data out of the network. An external identification and configuration device is assumed to connect to this device that generates the identification and configuration packets and analyzes the output packets that appear at the output of the network. The design of the compiler has not been studied as part of this thesis and is assumed to have the capability to identify the network using the algorithm mentioned below and generate configuration packets with the appropriate data and in optimal order for fast configuration times. During the manufacturing process of the network, each node is assumed to be assigned an unique address. This address information is used to identify the topology of the network.

To reconstruct the topology of the network, information regarding the connections between the nodes is required. This can be achieved if the address of the nodes at the ends of the link are known to the compiler. When the nodes see the external identification signal, they generate packets with the fields shown in figure 3.4. The description of the fields are as follows:

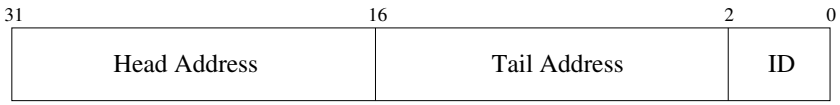


Figure 3.4: Identification packet contents. The packet and the individual field sizes shown are for this study and can be modified to suit the topology of the network.

- ID: 2-bit Packet ID. When this field consists of a 11, it indicates that the packet has already represents a link and need not be processed. It is thus forwarded through the output of the node. When the ID field consists of a 01, it indicates that the packet was generated by its neighbor and needs to be completed by adding the current nodes address to the packet and changing the ID field to 11 indicating that the packet is complete.

- Tail address: This field consists of the address of the node that generated the packet, which is also the node at the tail of the link.

- Header address: This field consists of the address of the node that is at the head of the link on which the packet is first placed.

Thus, a packet for each link in the network is generated and consists of information about the nodes that are at the head and tail of the link. The compiler can easily reconstruct the network using this information and generate the appropriate configuration bit stream. The algorithm for the node during the identification phase is shown in algorithm 2.

The packets representing the links closest to the output node of the network are expected to reach the output first and the packets representing the links farthest from the output node are expected at the end, indicating the end of the identification phase. An example of the identification algorithm and the expected sequence

Algorithm 2 Algorithm for the RBN node during the identification phase.

```
if Rising edge of clock then
  if IDN is asserted then
    Shift in data from each input into the respective input buffers
    Shift out data from output buffer to common output register
    packet bit count = packet bit count + 1
    if packet bit count = packet size then
      packet bit count = 0
      for each valid packet in input buffer do
        if IdentificationID = 01 then
          Place address of node in header address field of packet
          Place new packet in output buffer
        end if
        if IdentificationID = 11 then
          Mark packet to be forwarded to output nodes
          Increment respective buffer counter
        end if
      end for
      Place first packet marked for forwarding in output buffer
    end if
  end if
end if
if Falling edge of clock then
  Send data in output register to all outputs
end if
```

of packets are shown in figure 3.5.

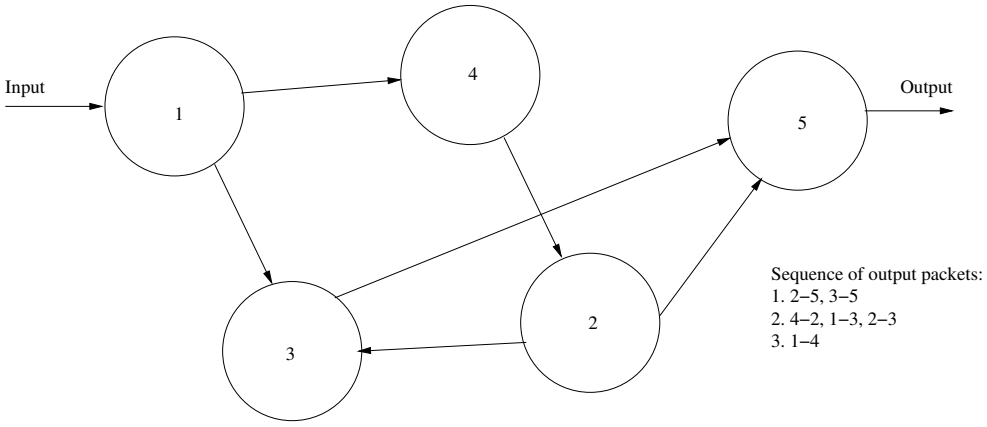


Figure 3.5: Example network showing the expected sequence of identification packets expected at the output of the network when in identification phase.

3.3.3 RBN node configuration algorithm

Node configuration takes place after the identification of the network topology is done. The compiler is assumed to analyze the network based on the topology and generates configuration data for each node, splits them into packets, and injects them into the network. Since the topology of the network is known, the compiler can estimate the time for the packet to reach its destination and accordingly inject new packets into the network. Each packet has the fields specified in figure 3.6. In the following, a description of each of the fields in the configuration packet is given.

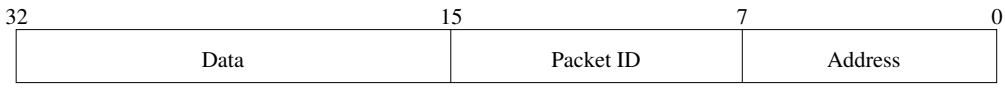


Figure 3.6: Configuration packet contents. The packet and the individual field sizes shown are for this study and can be modified to suit the topology of the network.

- Address: 8-bit field containing the address of the node that the packet is targeted to.
- Packet ID: 8-bit field representing the location in the look-up-table of the node where the data in the packet needs to be placed.
- Data: 16-bit field containing the data that needs to be placed in the look-up-table.

The address of the look-up-table at which the data is to be placed is calculated as follows: $(Address = (Packet\ ID * Data\ field\ size) - 1)$. Figure 3.7 shows an example of where the data would be placed for the given Packet ID and Data fields.

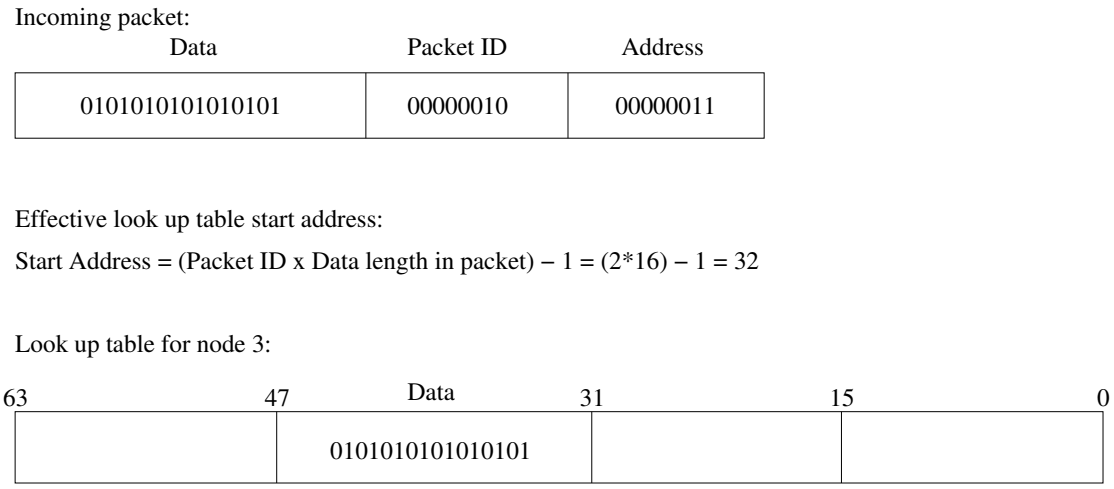


Figure 3.7: Calculation of the effective address at which data from the incoming packet is placed in the look-up-table.

The size of each of the fields in the packet can be resized to fit the requirements of the network size and connectivity. For example, a network with a large number of nodes and low connectivity (smaller look-up-tables in each node) will require more

bits for the address field and less for the packet id or the data fields. The size of each field and the size of the packet are fixed before the manufacturing process and the sizes are made known to the nodes.

Figures 3.8 to 3.13 describe an example for the configuration of 4 nodes using configuration packets sent into the network through the input. Common signals such as CLK, CNF, and RST are not shown for convenience, and it is assumed for this example that the network is in configuration mode (CNF is asserted). Each node has a look up table smaller than 16 bits (size of data field in packet) and hence requires only one packet for configuration. Algorithm 3 describes the operation of the node during this mode.

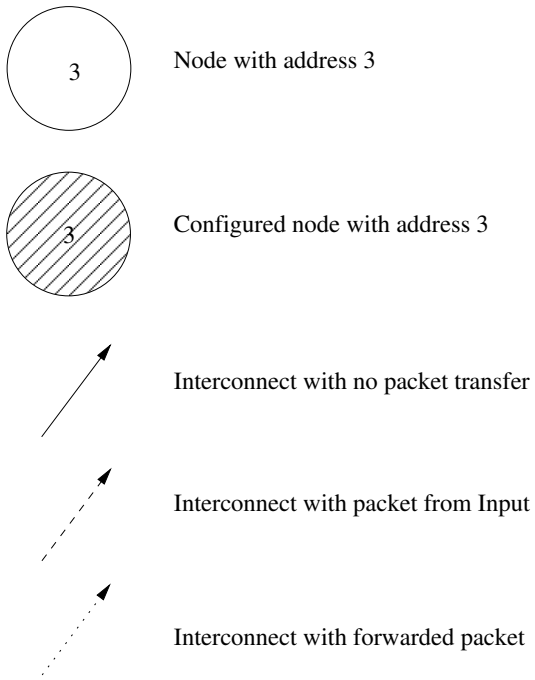


Figure 3.8: Notations for the RBN configuration example.

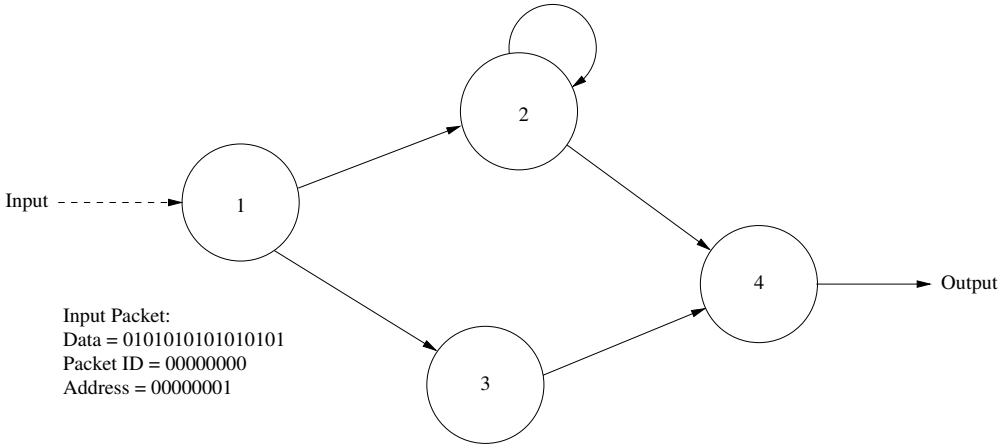


Figure 3.9: Step 1: A packet addressed to node 1 with the appropriate data is sent into the network through the input.

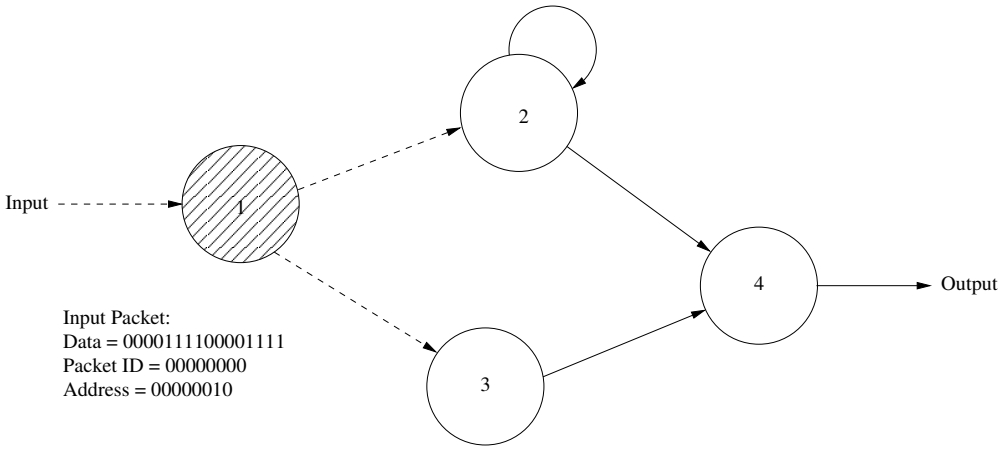


Figure 3.10: Step 2: Node 1 absorbs the packet and places the data in its look-up-table. Another packet addressed to node 2 is sent into the network. Node 1 forwards this packet onto both its outputs since there is no address match with node 1.

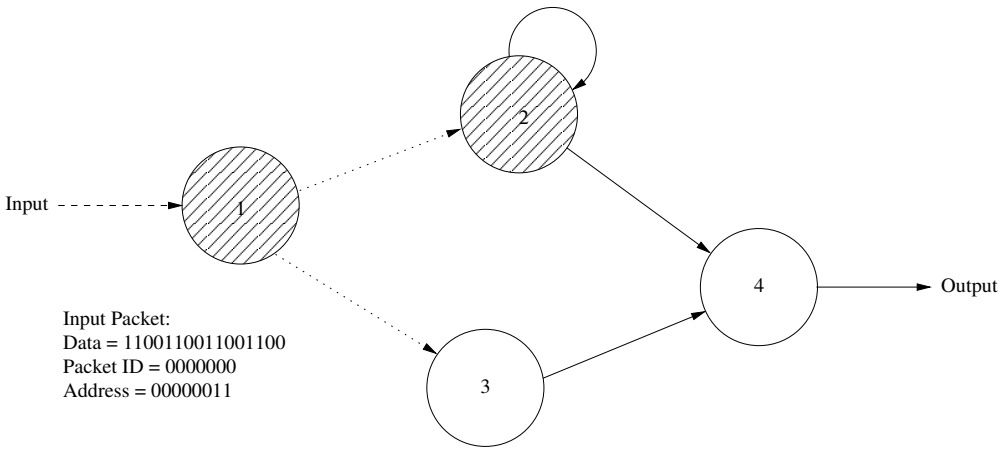


Figure 3.11: Step 3: Node 2 absorbs the packet forwarded from node 1 and places the data in the packet in its look-up-table. Node 3 forwards the packet from node 1 since there is no address match. Another packet addressed to node 3 is sent into the network.

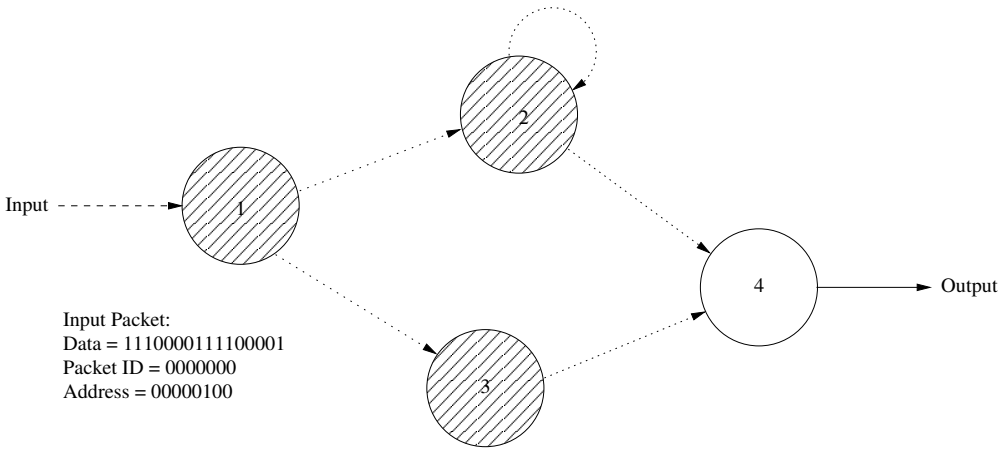


Figure 3.12: Step 4: Node 3 absorbs the packet forwarded from node 1 and places the data in the packet in its look up table. Nodes 1,2 and 4 forward the packet onto all of its outputs. Another packet addressed to node 4 is sent into the network. This packet is forwarded by nodes 1,2 and 3 to all of its outputs.

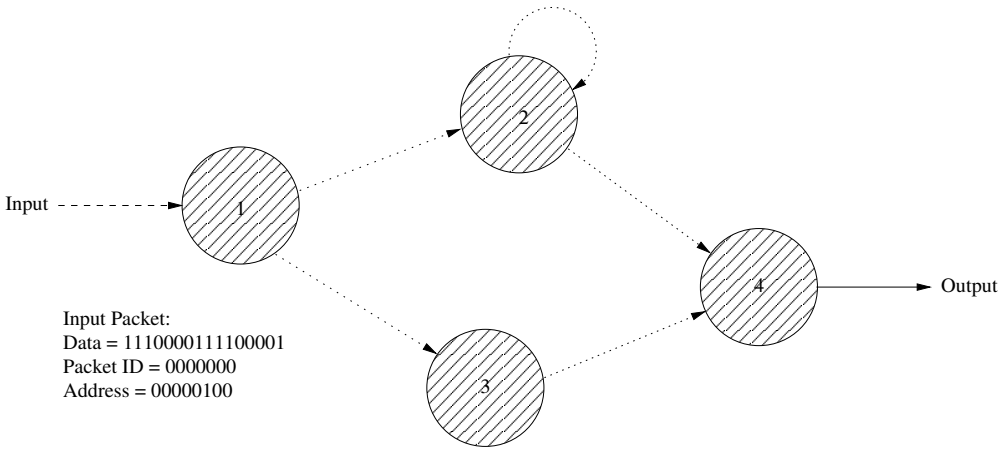


Figure 3.13: Step 5: Node 4 absorbs the packet addressed to it and places the data in the packet in its look up table. All nodes are now configured and the network is ready for normal operation.

3.3.4 RBN node normal operation algorithm

Normal operation of the RBN node is as described in section 2.1. The algorithm for the node is as shown in algorithm 4. It is similar to the operation of the CA node described in algorithm 1.

The function that determines the look-up-table address for this study is a simple binary to decimal conversion. For example, if there are 5 inputs and the input combination is $(Input1, Input2, Input3) = 010$, the look-up-table address is the decimal value of the combination of inputs 2. This mapping between the look-up-table address and the inputs can be chosen to be any function of the inputs, such as the sum or products of the inputs.

Algorithm 3 Algorithm for the configuration of a RBN node.

```
if Rising edge of clock then
  if CNF is asserted then
    Shift in data from each input into the respective input buffers
    Shift out data from output buffer to common output register
    packet bit count = packet bit count + 1
    if packet bit count = packet size then
      packet bit count = 0
      for each valid packet in input buffer do
        if address in packet = address of node then
          LUT address = (packet ID * data field size) − 1
          LUT[LUT address] = data in packet
        else
          Mark packet to be forwarded to output nodes
          Increment respective buffer counter
        end if
      end for
      Place first packet marked for forwarding in output buffer
    end if
  end if
end if
if Falling edge of clock then
  Send data in output register to all outputs
end if
```

Algorithm 4 Algorithm for RBN node normal operation.

```
if Rising edge of clock then
  if CNF and IDN are not asserted then
    LUT address = function(Input 1, Input 2, ..., Input n)
    Result = data at LUT address
  end if
end if
if Falling edge of clock then
  (Output 1, Output 2, ..., Output p) = Result
end if
```

The operation of the node during the positive and negative clock cycles are as shown in figure 3.14. For this study, all the nodes in the network run on a common clock to maintain synchronization, which may not be an ideal situation considering clock distribution constraints. Asynchronous communication techniques are a promising area to eliminate this problem and provide scope for further study.

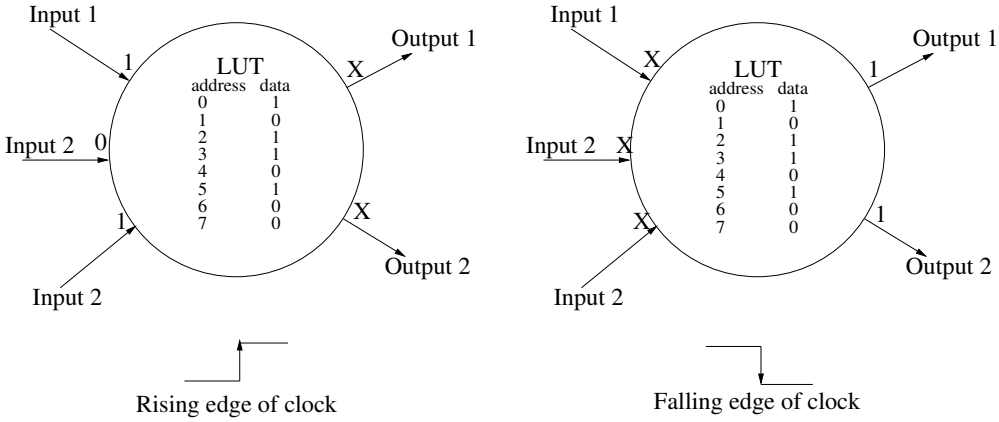


Figure 3.14: Operation of the RBN node during the rising and falling edges of the clock as described by the algorithm 4. During the rising edge of the clock, the node checks the combination of inputs and selects the corresponding value from its look-up-table. At the falling edge of the clock, the value from the look-up-table is sent to its outputs.

3.4 Nanowire model

Snider and Williams [14] describe a nanowire model that can be used for the interconnect between CMOS layers. The CMOS layers implement the difficult functions necessary for logic while the nanowires are used for wired-OR logic and signal routing. This study uses this nanowire model for evaluating the interconnect between the nodes of the network.

The dynamic power consumption due to the interconnect can be determined using the following formula:

$$Dynamic\ power = \frac{1}{2} * A * N * C * V_{dd}^2 * f \tag{3.1}$$

Where, A is the switching activity of the signal through the wire. We approximate the switching activity as 0.5 for this study. N is the length of the wire, which can be determined using the adjacency matrix for the network and the random locations for each node of the network. The capacitance per unit length for the wire C is found to be $2.0pFcm^{-}1$ for the nanowire model. The supply voltage V_{dd} is considered to be $1V$ and the frequency of operation used for the study is $100MHz$.

The dynamic power consumption using the above formula is found to be $N*5*10^{-9}$. The interconnect power is added to the estimated operation power due to operation of the node obtained using the Synopsys Design Compiler tool to get the total power consumed by the node.

4

Evaluation Methodology

4.1 Overview

An estimation of the power and area consumption of the networks under study at the hardware level will provide parameter ranges that give and economical implementation of the networks. Since there exist well-established tools for simulating devices implemented on silicon and since the nodes in the network are identical in operation, the nodes in the networks are constructed using HDL. For the connections between the nodes, a Verilog top module for the network was generated for each network by instantiating each node configured to operate for a number of inputs and outputs specified by the adjacency matrix.

The following tools were used in the estimation of power and area of CAs, RBNs and the small-world networks.

- MATLAB - Used for the generation of adjacency matrix of the networks, generation of the verilog top module, generation of small-world networks, and evaluation of the density task.
- ModelSim - Used for the functional simulation of the networks and generating switching activity for power estimation.
- Synopsys Design Compiler [15] - Used for synthesis and generation of area and power estimation.

4.2 Network generation

Network generation is done in MATLAB using an existing network toolbox that is capable of generating networks with specified network parameters, such as the number of nodes N and the average connectivity k. The toolbox provides the adjacency matrix of the network as the output. The adjacency matrix is analyzed and the information is used to determine the input and output connections to each node. The size of the look-up-table for each node is computed and the HDL is generated by instantiating each node with the parameters obtained from the adjacency matrix. The algorithm for generating the top module for the HDL of the network is shown in algorithm 5

Algorithm 5 Algorithm for the generation of the HDL top module for the network.

```
Define module and its inputs and outputs
Declare connections between nodes as follows
for each element in adjacency matrix do
  if value of element = 1 then
    Declare connection with name C_⟨elementrow⟩_⟨elementcolumn⟩
  end if
end for
for each row in adjacency matrix r do
  Parameter number of inputs = sum of elements in row r
  Parameter number of outputs = sum of elements in column r
  Parameter size of LUT = 2^numberofinputs
  Instantiate node with clock, reset, configure signal and the above parameters
  for each element in row r do
    if element = 1 then
      Declare C_⟨elementcolumn⟩_⟨elementrow⟩ as an input to the node
    end if
  end for
  for each element in column r do
    if element = 1 then
      Declare C_⟨elementrow⟩_⟨elementcolumn⟩ as an output to the node
    end if
  end for
end for
```

The toolbox is also capable of generating small-world power-law networks with a rewiring distance probability determined by the parameter α. These networks as described in section 2.3, allow for a study of the effect of the randomness of the network interconnects on the power and area consumption of the networks.

The top module generated using the above algorithm interfaces to the test bench through a single input and output from the network and also receives the clock and reset signal. Figure 4.1 shows the block diagram for the system framework used for the study.

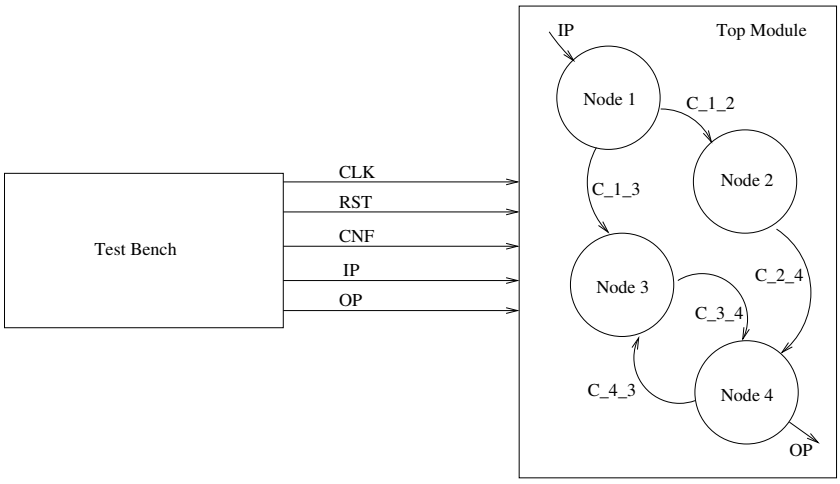


Figure 4.1: Block diagram for the simulation framework used. The HDL for the top module is generated by a MATLAB script. The HDL for the node is written by hand and instantiated with the required parameters in the top module.

The test bench is responsible for generating the clock and providing inputs to the network. An array of input values is defined and shifted into the network through the input port during every clock cycle. The simulation stops after a specified number of clock cycles.

4.3 Power estimation

The power consumption of the network is estimated using the HDL with the models for the assumed nanowire interconnects in section 3.4. Power is estimated for the design by first simulating the HDL of the *complete* network in ModelSim using a test bench that provides inputs to the network. The switching activity of each signal in the design is saved in a value change dump (VCD) file using special constructs in the test bench. The HDL for the network is then synthesized using the Synopsys Design Compiler [15] tool and the switching activity obtained from the simulation is mapped onto the synthesized design and the total power the network consumes because of the switching activity is estimated using the technology library specified (180nm technology was used for the study). Since the Synopsys Design Compiler does not support reading a VCD file, it is required to convert the VCD file generated by ModelSim to the switching activity interchange format (SAIF) that can be read directly by Design Compiler. The total power consumed by the networks are extracted from the power reports generated by Design Compiler using a PERL script and are exported to MATLAB for analysis. The design flow for the estimation of power consumption is shown in figure 4.2.

The total power consumption of the network is then estimated by adding the power of to the following parts of the network:

- Identification: Power consumption for the identification of the topology of the network. CAs do not require this step because the topology of the network is fixed for all CAs.
- Configuration: Power consumption for placing data into the look-up-tables of the nodes.

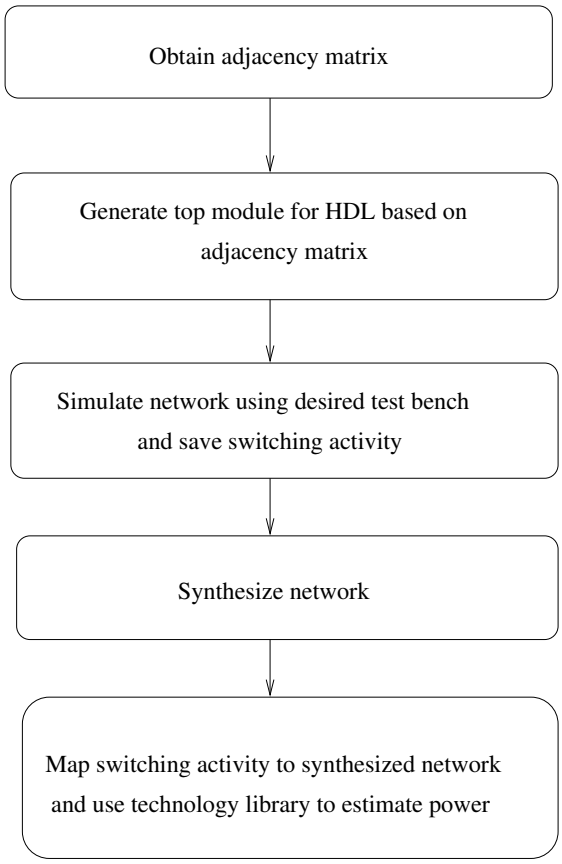


Figure 4.2: Design flow for estimation of power consumption using Synopsys Design Compiler.

- Operation: Power consumption of to the normal operation of the random Boolean network.
- Interconnect: Power of the nanowire interconnect based on the model that has been assumed for this study.

4.3.1 Power estimation for identification and configuration stages

Measuring the power consumption of the network with the identification and configuration logic in the node using the above method is not feasible directly using the above method because of the large synthesis and simulation times. For this

purpose, an approximation of the power consumption is obtained by creating a database with the power consumption values for identifying and configuring nodes with various number of inputs (various connectivities) and use the database to estimate the total power consumed by the network of nodes by adding the power consumption of individual nodes in the network. This is repeated over multiple networks with the same network parameters and the power due to identification and configuration is averaged over these networks. The average power for configuring the nodes is added to the average power for normal operation of the network and the power of the interconnect as shown in figure 4.2.

4.3.2 Power estimation for operation stage

Since the logic for the RBN and CA nodes is simple, the entire network is synthesized together and simulated for 2,000 time-steps, toggling the input every 100 time-steps. The switching activity obtained from this simulation is used by the power compiler tool to estimate the power consumed by the network. This is repeated over multiple networks with the same network parameters and the power consumed is averaged over these networks and added to the average power obtained due to the identification and configuration logic.

4.3.3 Power estimation for nanowire interconnect

To estimate the power consumed by the wire, we use the nanowire model presented in [14]. The power consumed by the wire per unit length of the nanowire model is calculated based on the capacitance values provided in the paper. This is explained further in section 3.4. The nodes of the network were placed in a two dimensional plane at random locations or in a mesh, and the distance between the

nodes was calculated to obtain the length of the wire if a link was present between the nodes. The minimum spacing between the nodes was considered to be $1\mu m$ and the nodes were placed in a square whose side was $\frac{N}{2}$ μm. The total length of the nanowire for all links is multiplied with the power per unit length of the wire to estimate the total power consumed by the interconnect of the network. As done for the previous stages, the power is averaged over multiple networks with the same network parameters and added to the power due to the previous stages.

4.4 Area estimation

The area consumption is estimated with the Synopsys Design Compiler based on the HDL code for the network. Area estimation can be categorized into two parts.

- Area consumption of the logic for identification, configuration, and operation stages of each node of the network generation.
- Area consumption of the nanowire interconnect.

The area consumption of the logic for identification, configuration, and operation stages of each node is computed individually using the Synopsys Design Compiler tool for nodes with different connectivities, and is saved in a database. Using the adjacency matrix of the network, the connectivity of each node is computed by adding the columns of the matrix. The total area of the network is then easily obtained by adding the individual areas of each node from the database created previously. The area estimations are completely dependent of the technology library used for synthesis, which in this case is the OSU 180nm technology library.

4.5 Generation of small-world networks

As described in section 2.3, small-world networks are a set of networks that are generated by the controlled rewiring of an initial locally connected network. The rewiring is done as a function of the rewiring probability p and the length of the rewired wire follows the power law distribution $l^{-\alpha}$. To ensure that the network generated remains connected, we add R extra local links to the network before the rewiring procedure. The algorithm for the generation of these networks is shown in algorithm 6. Initially the nodes are arranged in a grid where each node is separated from its closest neighbors by 1 unit of distance. The following algorithm was implemented using a MATLAB script, which returns the adjacency matrix of the small-world network. The average interconnect length distribution for small-world networks is shown in figure 5.5 and example networks that were generated using the algorithm are shown in figure 2.5.

Algorithm 6 Algorithm for the generation of the adjacency matrix for a small-world network.

Generation of the initial mesh network.
for Each $node_i$ **do**
  for Each $node_j$ **do**
    if distance between $node_i$ and $node_j$ is 1 unit **then**
      create link from $node_i$ to $node_j$
    end if
  end for
end for

Addition of R extra local links.
for each extra link to be added **do**
  while link not created **do**
    Select random node $node_1$
    Select random node $node_2$
    if Distance between $node_1$ *and* $node_2$ $=$ 1 *unit* **then**
      Create link from $node_1$ *to* $node_2$
      Signal link created
    end if
  end while
end for

Rewiring with probability p and length distribution α.
for Each link in network **do**
  Generate random number between 1 and 100
  if *number generated* $\leq p$ **then**
    rewire link with power law distribution
  end if
end for
for each element in adjacency matrix **do**
  if *element value* > 1 **then**
    element value $= 1$
  end if
end for

5

Results

5.1 Variation in power consumption with number of nodes in CA and RBN

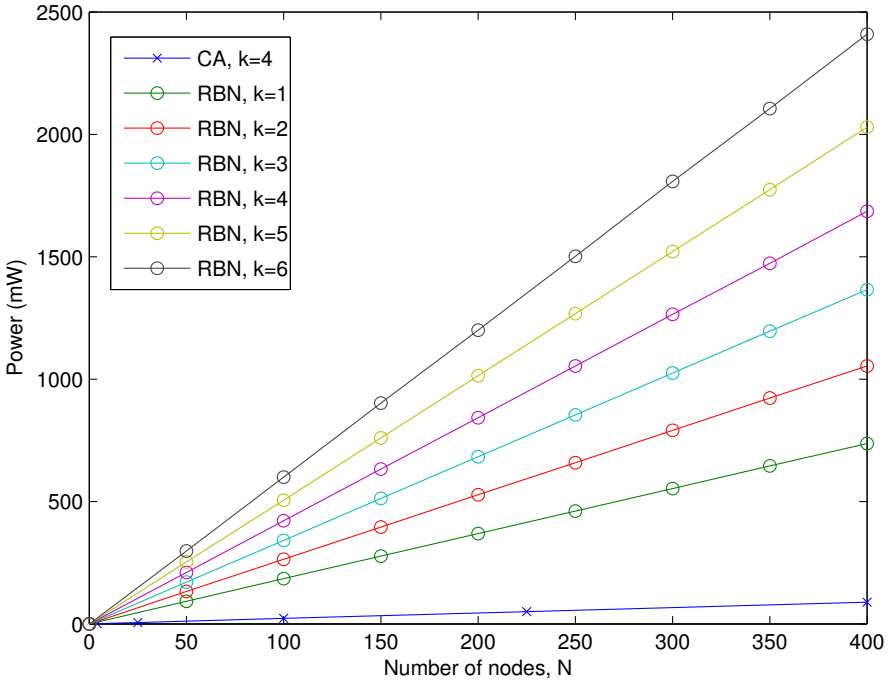


Figure 5.1: Power consumption in CAs and RBNs increases linearly with and increasing number of nodes in the network. The power consumption for RBNs with different connectivities are shown. The power consumption for the RBNs is much higher compared to CAs because of the large amount of logic involved in the identification and configuration of the networks.

The power consumption in RBNs and CAs was evaluated using the framework described in section 4.3 on networks generated using the algorithm specified in

section 4.2. For the RBNs, the results were averaged over 50 networks for each network size and average connectivity. As seen from figure 5.1, the power for both types of networks increases linearly with network size. The power consumption for RBNs is much higher, even for low average connectivities, because of the large amount of overhead required for the identification and configuration logic described in section 3.3.1.

5.2 Variation in area consumption with number of nodes in CA and RBN

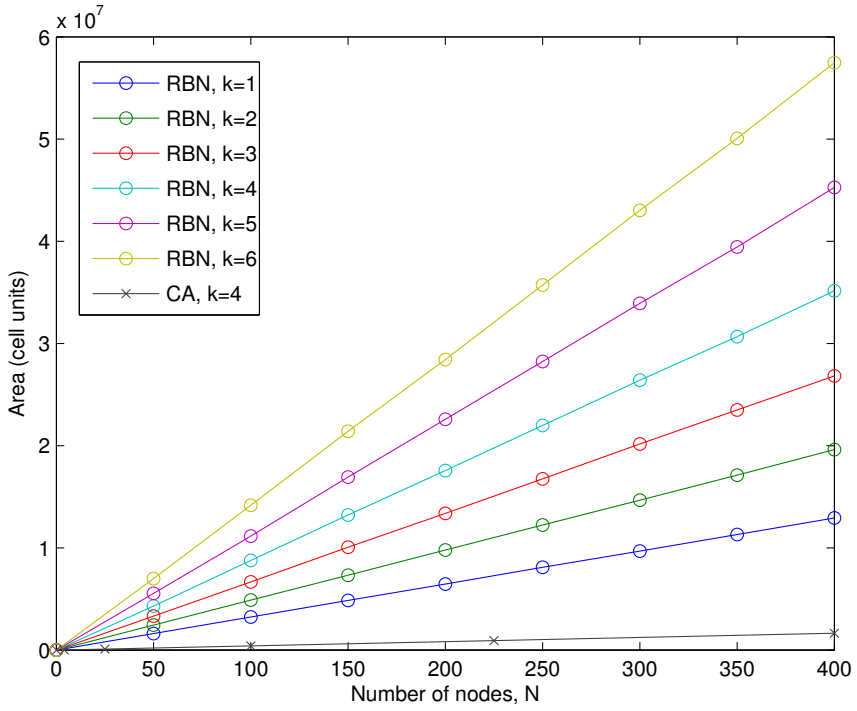


Figure 5.2: Area consumption in CAs and RBNs increases linearly with an increasing number of nodes in the network. The area consumption for RBNs with different connectivities are shown. The area consumption for the RBNs is much higher compared to CAs because of the large amount of logic involved in the identification and configuration of the networks.

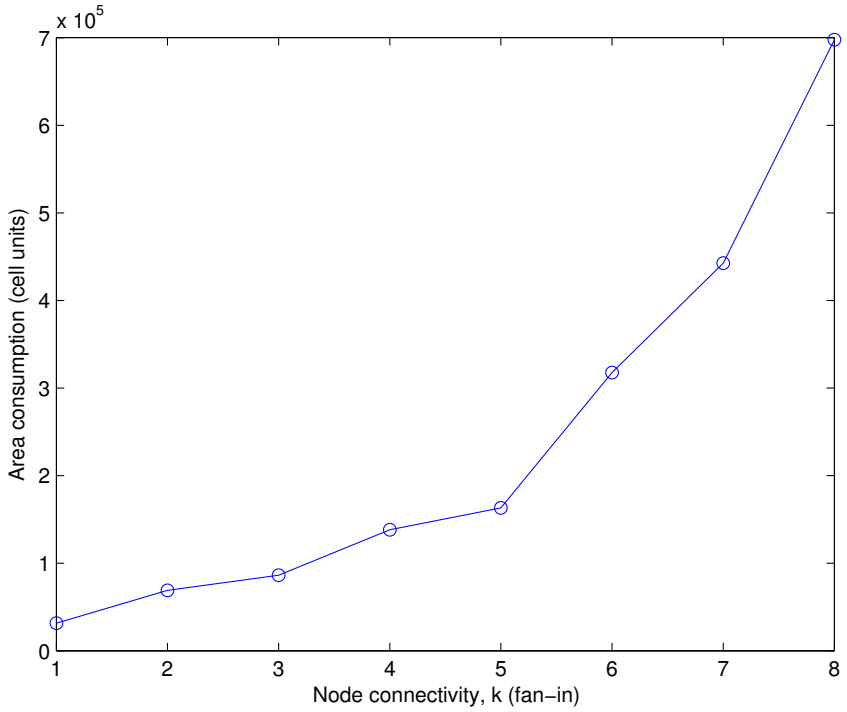


Figure 5.3: Area consumption of RBN nodes with different fan-in. As mentioned in figure 5.2, the area of the node increases with the increase in connectivity (fan-in) because of the increase in the size of the look-up-table and the increase in number of input buffers.

The area consumption is evaluated using the framework described in section 4.4. Similar to the power consumption evaluation, the RBN values were averaged over 50 random networks. Both the CA and RBN implementations include the logic for configuration of the networks and the identification logic for the RBN implementation for a fair comparison of the area consumption values. The results obtained are similar to the power evaluation results and the high area consumption of RBNs can be attributed to the large amount of configuration logic overhead such as input buffers. Also the look-up-table in the RBN node increases with the fan-in of the node to support all input combinations. This is highlighted by the plot in figure 5.3 which shows the area consumption for an individual RBN node. It can be seen

that the are increases exponentially with an increase in connectivity.

5.3 Power consumption in RBNs due to identification, configuration, and normal operation logic

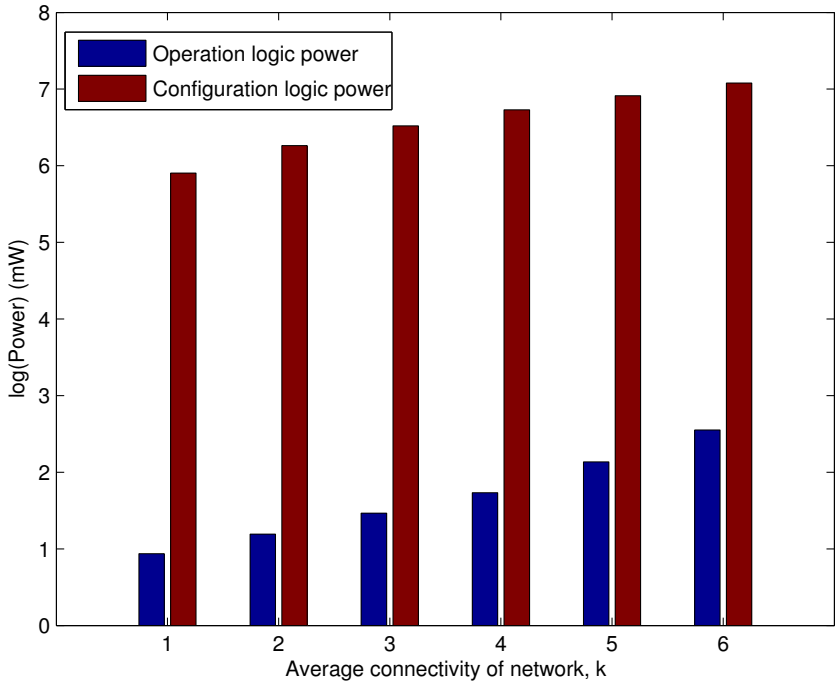


Figure 5.4: Increase in power consumption due to the addition of identification and configuration logic to the RBN node. The increase in power due to the identification and configuration logic scales with the connectivity of the network since the number of input buffers and number of packets required for configuration increase. The data shown in the plot is for networks with 200 nodes.

The RBN node power consumption can be analyzed further by separately evaluating the power for normal operation and configuration and identification of the node. Figure 5.4 shows the power consumption when the node is being configured and when the node is performing the RBN functions. Power due to configuration

dominates the node power, but if the configuration logic switched off during the normal operation phase and used only during the configuration phase, the high power consumption will be applicable only during the configuration phase and the node will consume much less power during normal operation.

5.4 Power consumption in small-world networks

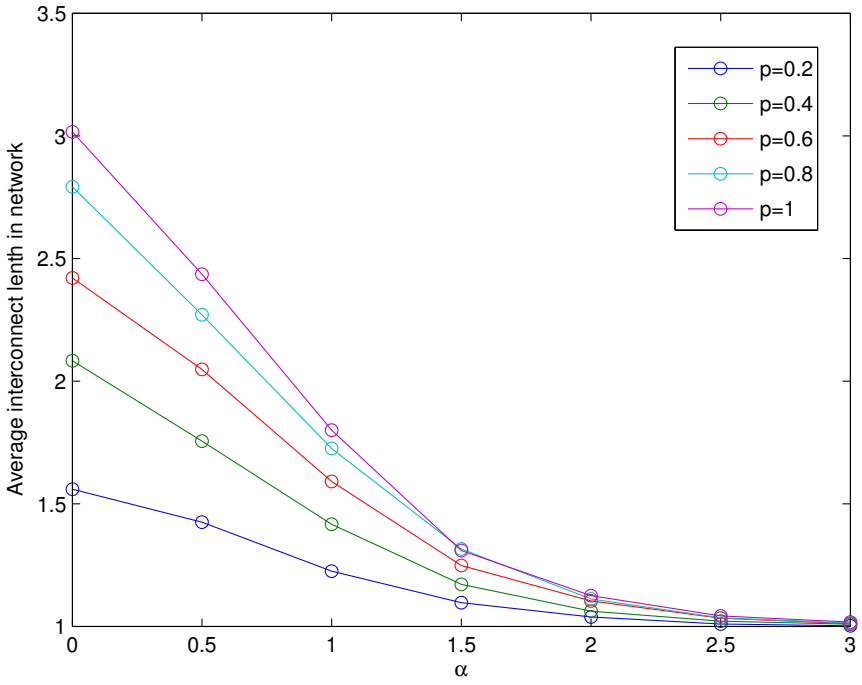


Figure 5.5: Interconnect distribution for small-world networks measured as the average interconnect length in the network for rewiring parameters p and α. The average interconnect values were averaged over 25 networks for each rewiring parameter and the number of nodes in the network is 64.

Using the small-world network generation algorithm described in section 4.5, a set of networks that allowed us to evaluate and compare random networks as a function of the amount of randomness. The average length of a link in the network

has a distribution shown in figure 5.5. A higher value of α results in a more locally connected network and a lower value of α increases the length of the link based on the power law distribution. For $\alpha = 0$, we obtain a network with a uniform high length distribution.

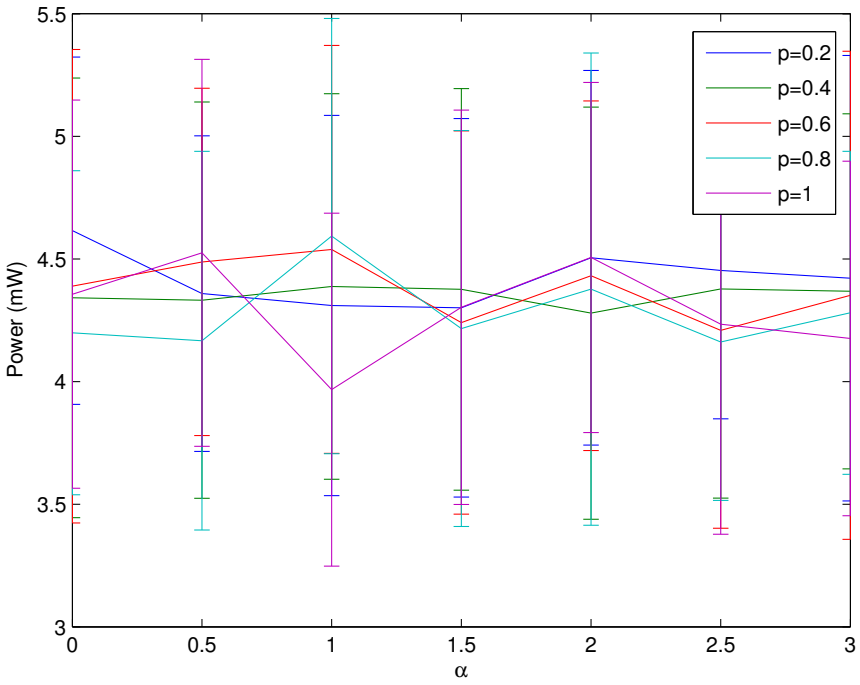


Figure 5.6: Total power consumption of small-world networks as a function of the rewiring parameters p and α. The power values for each parameter were averaged over 25 networks and the number of nodes in the network is 64. Since the power due to the configuration logic dominates the total power consumption of each node. Since the wires are only rewired and the average connectivity of the node remains approximately the same at $k = 4$, the total power consumption remains the same with increase in α and p. The power consumption of the initial mesh network is shown for comparision.

The power consumption for small-world networks is shown in figure 5.6. The power remains the same in the network because the power due to the node dominates the

interconnect power and since the average connectivity of the network remains approximately the same ($k = 4$) for the small-world networks, the power consumption remains mostly constant. On the other hand, the interconnect power estimated using the nanowire model described in section 3.4 and shown in figure 5.7 decreases as α increases and longer links are less probable. But, as p decreases, the probability that a link is rewired reduces, and, as a consequence, less long range links are present in the network. The interconnect power thus reduces for smaller values of α. The variation in interconnect power with respect to α is smaller for low rewiring probabilities p and higher for larger values of p. Also for larger values of p, as α increases, links are rewired to connect neighboring nodes and hence multiple links to neighboring nodes are created. These links are eliminated later, when generating the HDL to emulate realistic hardware and reduce the total interconnect length of the network as a result.

5.5 Density task evaluation of small-world networks

The density task evaluation (section 2.4) provides a benchmark for evaluating small-world networks in terms of information processing. Figure 5.8 shows the number of iterations to solve the task successfully by the network. Random networks have the advantage of a shorter average path between two nodes in the network, and hence the output of a node is transmitted to a node situated at the far end of the network in a short span of time. Hence the shorter time to solve the density task.

Mesot et al. [10] present the performance in solving the density task as a function of the number of nodes and connectivity of random networks. Their study has

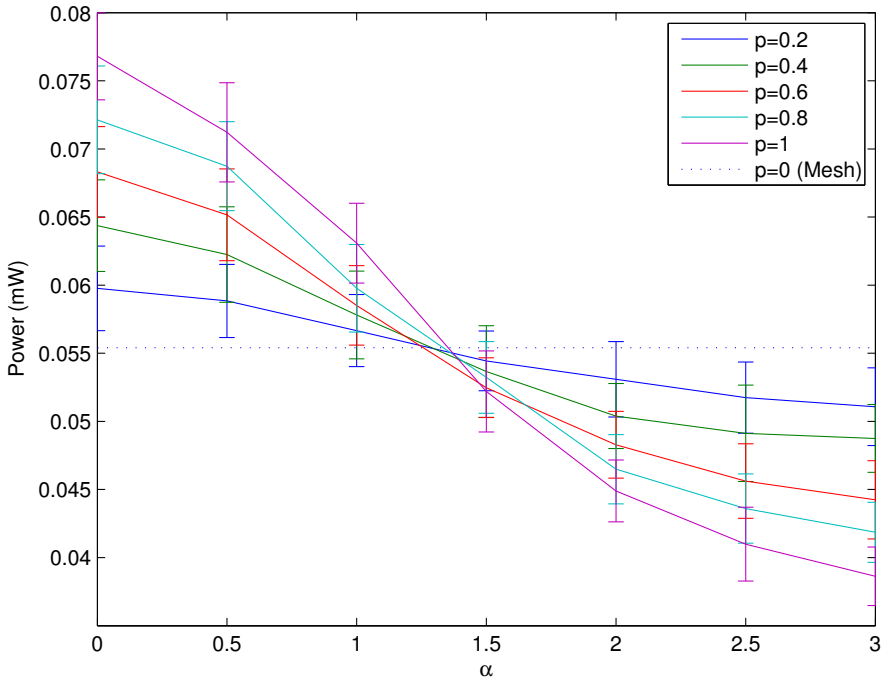


Figure 5.7: Interconnect power consumption of small-world networks as a function of the rewiring parameters p and α. The power values for each parameter were averaged over 25 networks and the number of nodes in the network is 64. For larger rewiring probabilities, there is a larger variation of total interconnect length because more links are rewired as a function of α. Similarly, for low rewiring probabilities, the variation in total interconnect length is small. Since the interconnect power varies linearly with the total interconnect length, the variation in the interconnect power with respect to alpha is different for various p.

show that networks with a large number of nodes require more time to solve the task but, since there exists long distance links in random networks that allow for faster propagation of information, the performance degradation is not linear with the number of nodes and follows a logarithmic-like law. The findings in this study show that the performance of random networks degrades with increase in alpha (decrease in long range links) which are consistent with the results presented by Mesot et al.

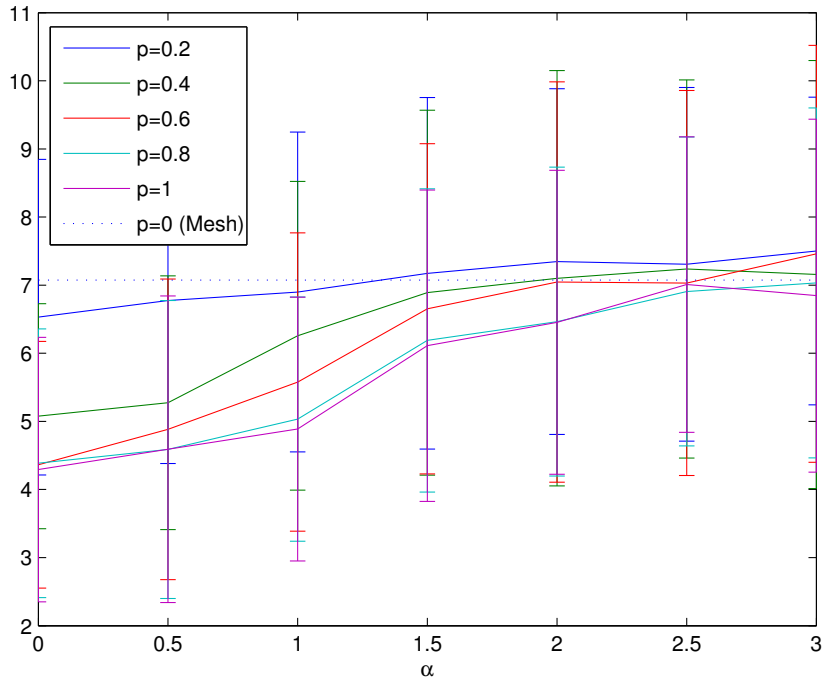


Figure 5.8: Plot showing the average number of iterations at which the density task is successful for small-world networks. Locally connected networks take longer to solve the task while networks with more global links are have a shorter average path length between nodes and can communicate faster. The results were averaged over 100 networks and $R = 100$ was considered to maintain connectivity. The performance of the initial mesh network is shown for comparison. Networks with 64 nodes were used for this study.

5.6 Aggregate objective function for the interconnect power consumption and density task performance

Using the aggregate objective function for the interconnect power consumption and density task performance, small-world network parameters which provide the best performance can be determined. The performance areas, interconnect power and number of iterations for solving the density task successfully, can be weighted based on the application that the network is targeted to. The aggregate objective

function used for the results in this section is given by:

$$(a)(Interconnectpower) + (1 - a)(Numberofiterations) \tag{5.1}$$

Where a is the weight given to the interconnect power. We analyze the performance of the networks as a function of small-world parameters α and p while keeping R constant at 100. Since most of the redundant links are eliminated by the small-world network generation toolbox, the variation in performance with respect to R was not found to be significant.

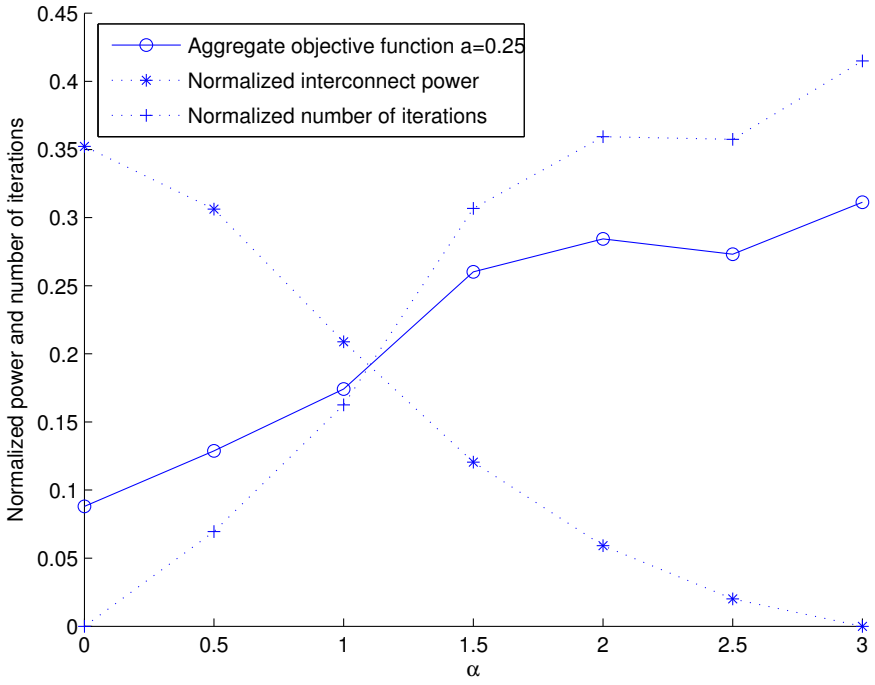


Figure 5.9: Aggregate objective function between interconnect power consumption and density task performance as a function of α. A weight of 25% was given to the interconnect power. The function is minimum at lower values of α indicating networks with long range connections are faster at information propagation. Networks with $N = 64$, $p = 0.6$ and $R = 100$ were used for this study.

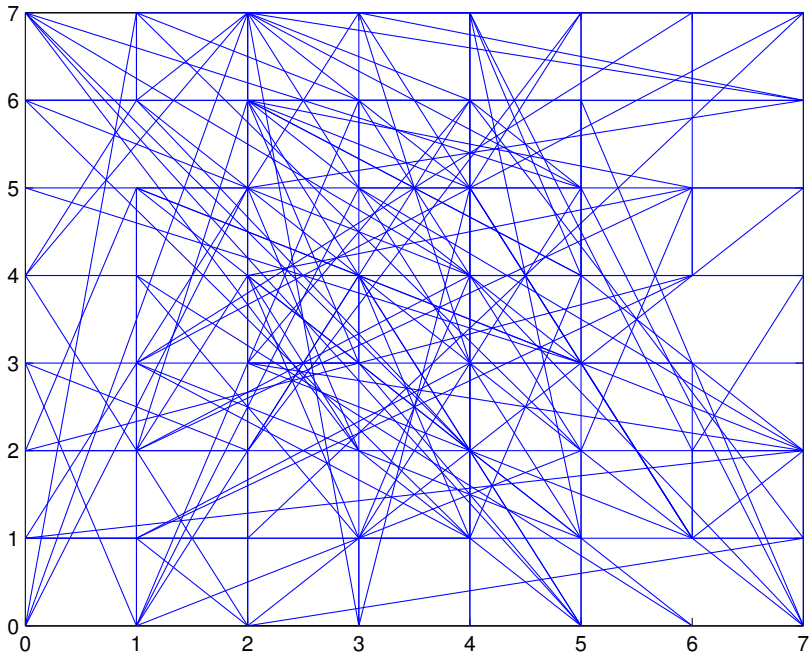


Figure 5.10: Example network for $\alpha = 0$, $p = 0.6$, and $R = 100$.

Figures 5.9, 5.11, and 5.12 show the aggregate objective functions as a function of α and a constant p of 0.6 for weights of 25%, 50%, and 75% of the interconnect power. It can be seen that for applications that require low interconnect power (more weight towards interconnect power), the aggregate objective function is minimum towards higher values of α indicating locally connected networks consume lower interconnect power. For applications that require high information propagation capability and willing to compromise on interconnect power, it can be seen that networks with long range links are more suitable. Figures 5.10 and 5.13 show example networks for some of the determined network parameters.

Figures 5.14, 5.16, and 5.17 show the aggregate objective functions as a function of p and a constant α of 1 for for weights of 25%. 50%, and 75% of the interconnect

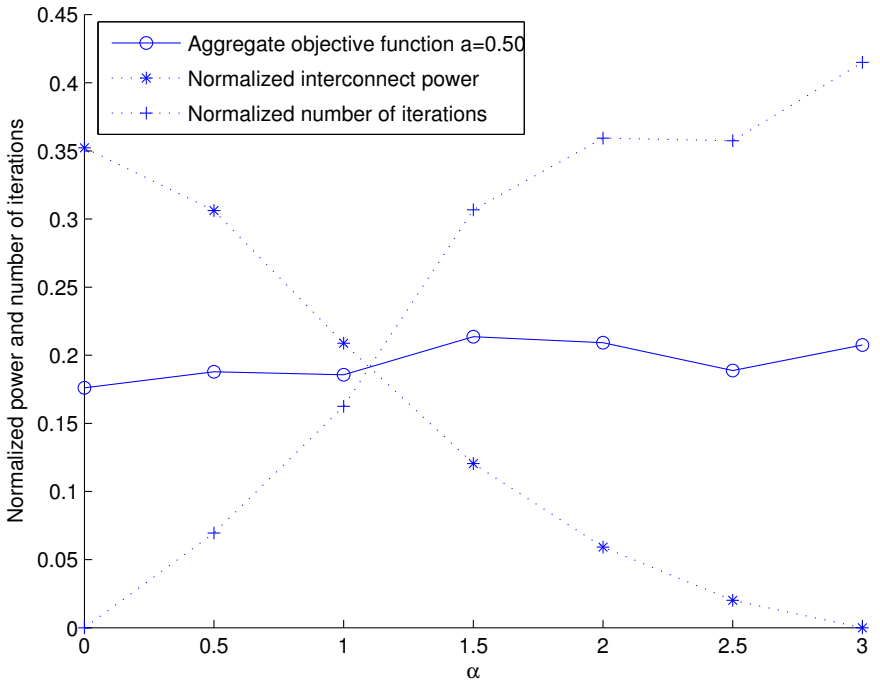


Figure 5.11: Aggregate objective function between interconnect power consumption and density task performance as a function of α. Equal weights were given to the density task and power performance. It can be seen that the interconnect power and density task performance compensate each other. Networks with $N = 64$ and $R = 100$ were used for this study.

power. It can be seen that for applications that require low interconnect power, the aggregate objective function is minimum at lower values of p. This is because of the lesser number of rewired links in the network. For applications that require high information propagation capability, it can be seen that networks with a higher number of re-wired links are desired. And for applications that require both less interconnect power and high information propagation capability, a rewiring probability of around 0.6 is desired. Figures 5.15 and 5.18 show example networks with the determined parameters.

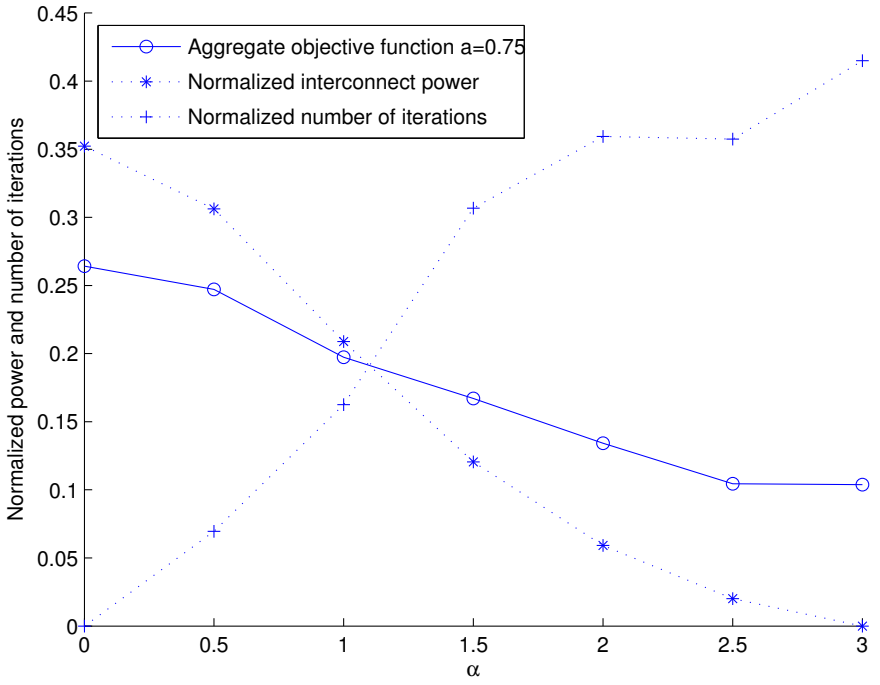


Figure 5.12: Aggregate objective function between interconnect power consumption and density task performance as a function of α. A weight of 75% was given to the density task performance. The function is minimum at lower values of α indicating networks with long range links propagate information faster. Networks with $N = 64$ and $R = 100$ were used for this study.

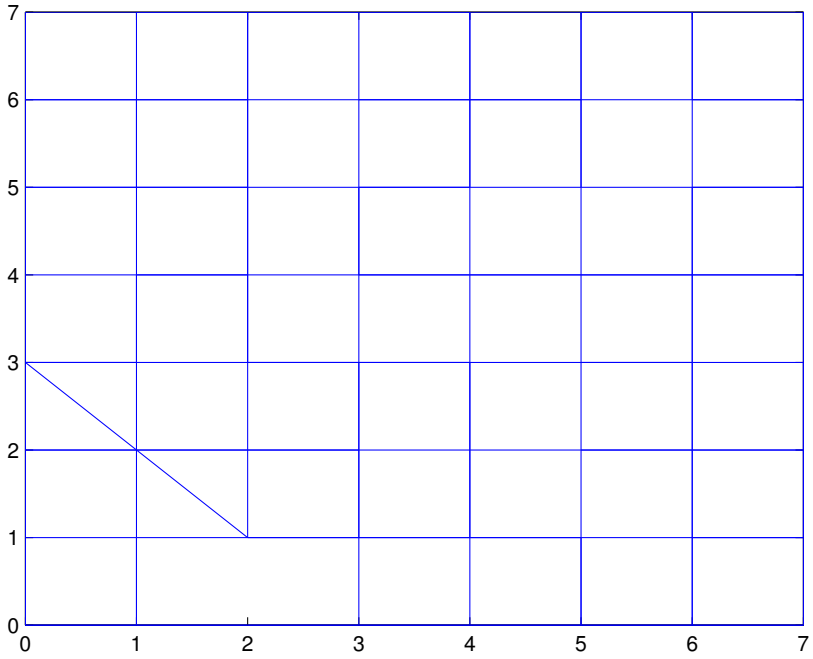


Figure 5.13: Example network for $\alpha = 3$, $p = 0.6$, and $R = 100$.

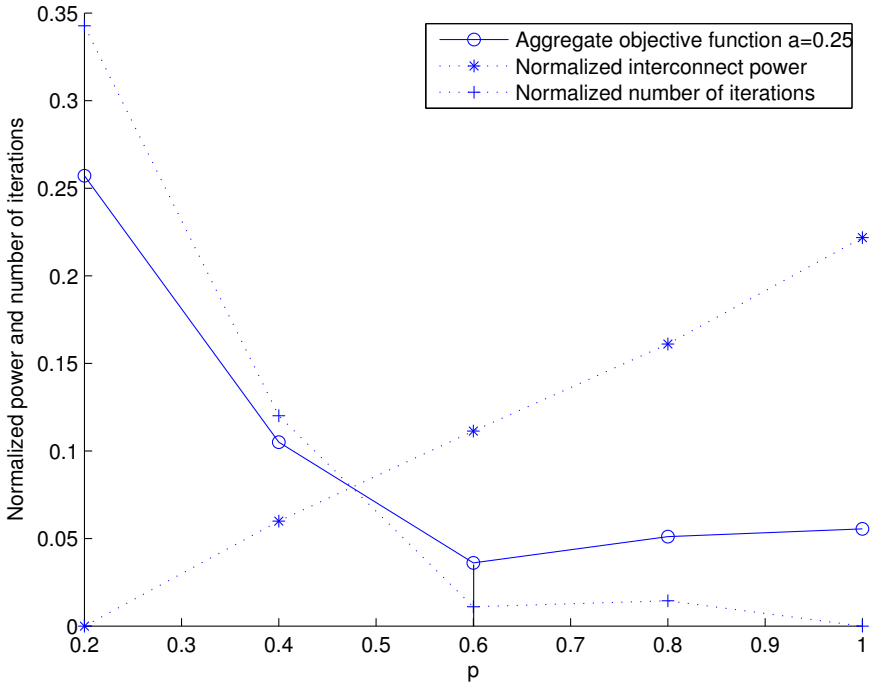


Figure 5.14: Aggregate objective function between interconnect power consumption and density task performance as a function of p. A weight of 25% was given to the density task performance. The function is minimum at lower values of p indicating networks with a lower number of rewired links consume lower power. Networks with $N = 64$ and $R = 100$ were used for this study.

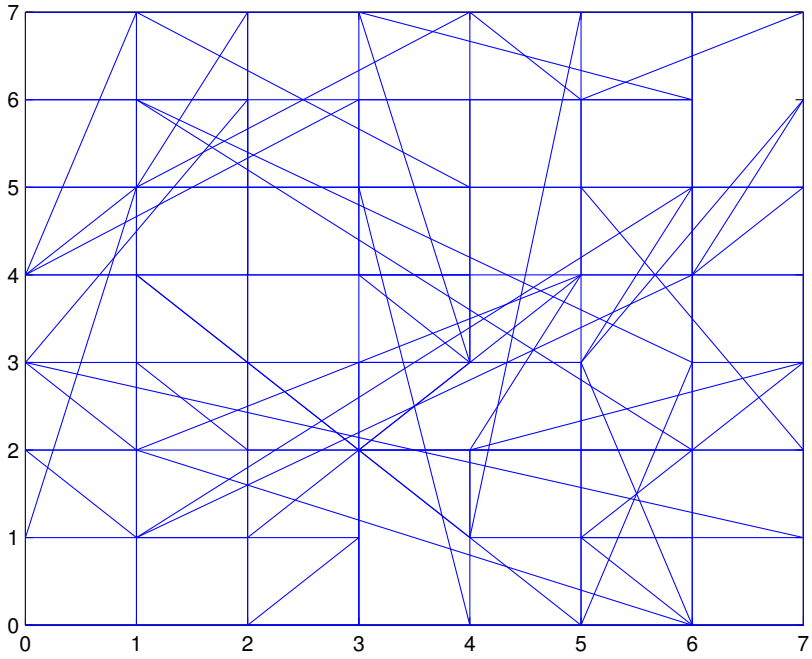


Figure 5.15: Example network for $\alpha = 1$, $p = 0.6$, and $R = 100$.

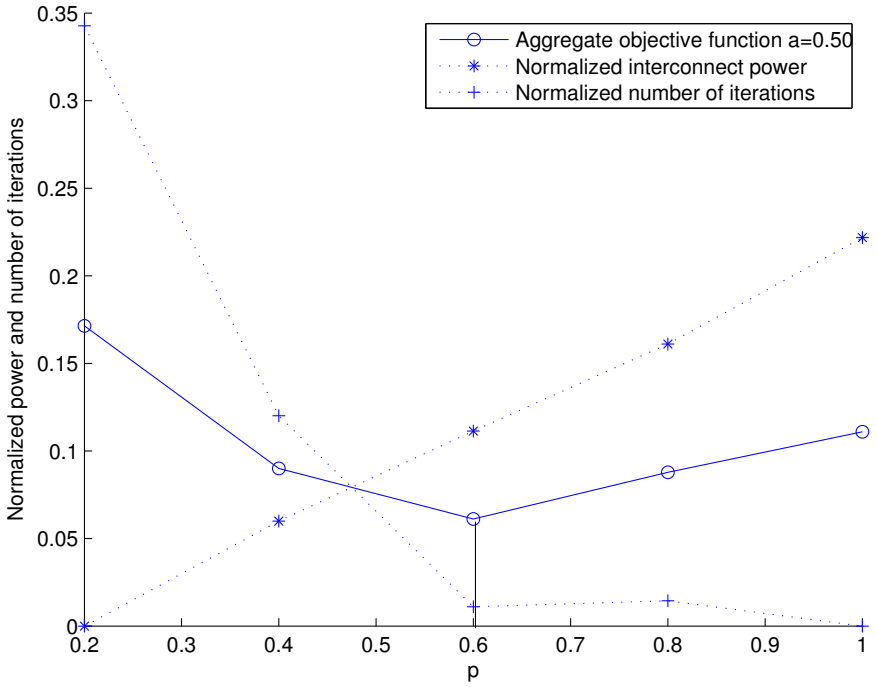


Figure 5.16: Aggregate objective function between interconnect power consumption and density task performance as a function of p. Equal weights were given to the density task and power performance. It can be seen that the function is minimum when the rewiring probability is around 0.6. Networks with $N = 64$ and $R = 100$ were used for this study.

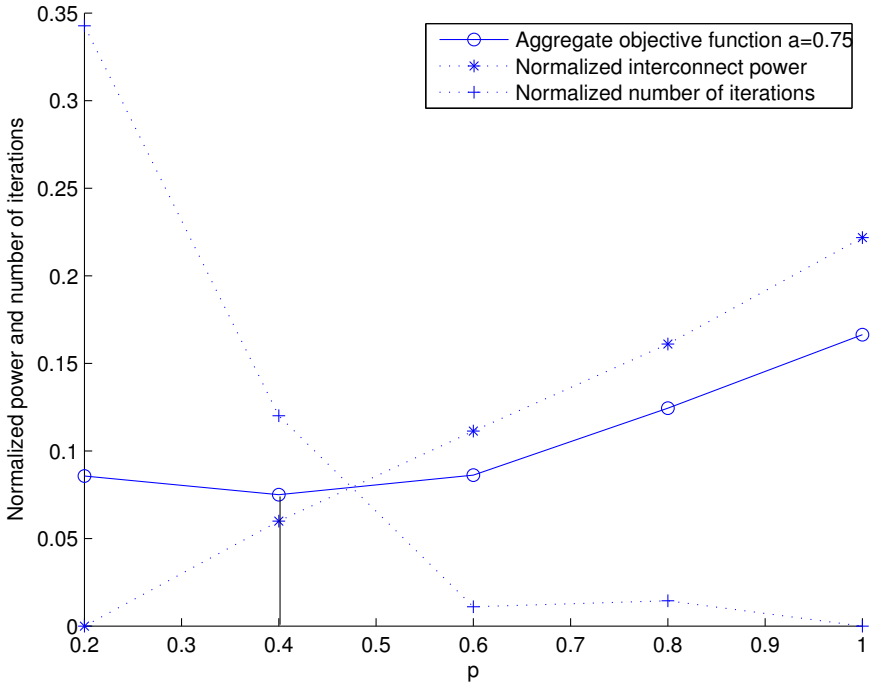


Figure 5.17: Aggregate objective function between interconnect power consumption and density task performance as a function of p. A weight of 75% was given to the density task performance. The function is minimum at lower values of p indicating networks with a more number of rewired links propagate information faster. Networks with $N = 64$ and $R = 100$ were used for this study.

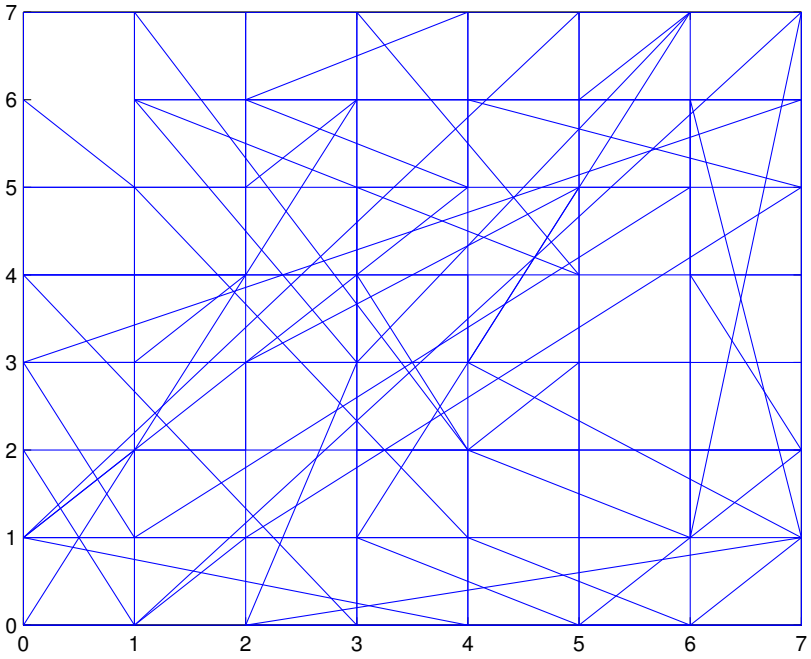


Figure 5.18: Example network for $\alpha = 1$, $p = 0.4$, and $R = 100$.

6

Conclusion

An hardware-level analysis of random and mesh architectures is presented to support the hypothesis that future nanoscale devices will have random architectures because of the expected lack of control in the manufacturing process of these devices.

A software framework was developed to generate networks of any size and connectivity. The HDL code for the networks is generated using the framework and simulated and synthesized using tools established for current CMOS technologies. The framework is independent of the CMOS technology used and the libraries used by the tool can be replaced to support more recent technologies. An evaluation framework was developed to evaluate the area and power consumption of these networks and help designers make performance related decisions.

The framework was modified to support the generation of small-world networks which provide a larger database of networks, which can be fine-tuned to induce any amount of randomness to enable a deeper understanding of the networks and allow designers to make architectural decisions. The performance of these small-world networks was addressed using a benchmark that was used previously to evaluate celluar automata networks.

By studying the power and information processing capability of small-world networks, it was found a completely random network or a mesh network were unrealistic for use as a computing architecture. Power consumption in random networks is very high and scales up linearly with increase in the network size and exponentially with the average connectivity of the network. On the other hand, power consumption in mesh networks is much lower compared to random networks. In terms of information propagation or the average shortest path between any two nodes of the network, which was evaluated using the density task, it was found that random networks are faster compared to mesh networks in terms of information propagation. Hence, a small-world network which lies in between these extreme cases will be capable of providing the required performance for a desired application. The aggregate objective function for the performance of small-world networks in terms of information propagation and power allows designers to determine the amount of randomness that is required for the application.

References

[1] International technology roadmap for semiconductors (ITRS). Technical report, Semiconductor Industry Association, 2011 update.

[2] A. Amarnath, P. Damera, A. Goudarzi, and C. Teuscher. Latency and power consumption in unstructured nanoscale boolean networks. *Proceedings of the* 11^{th} *International Conference on Nanotechnology (IEEE Nano 2011)*, 2011. In press.

[3] G. Bilardi and F. P. Preparata. Horizons of parallel computation. *Journal of Parallel and Distributed Computing*, 27(2):172–182, June 1995.

[4] P. Damera. Power consumption in mesh and random networks. Sigma Xi Student Research Symposium, Portland State University, Apr 11 2011.

[5] M. Dascalu and E. Franti. Implementation of totalistic cellular automata. *Semiconductor Conference, CAS 2000 Proceedings. International*, 1:273–276 vol.1, 2000.

[6] M. Haselman and S. Hauck. The future of integrated circuits: A survey of nanoelectronics. *Proceedings of the IEEE*, 98(1):11–38, January 2010.

[7] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, March 1969.

[8] S. A. Kauffman. *The origins of order: self organization and selection in evolution*. Oxford University Press US, 1993.

[9] J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, 2000.

[10] B. Mesot and C. Teuscher. Deducing local rules for solving global tasks with random boolean networks. *Physica D: Nonlinear Phenomena*, 211(1-2):88–106, November 2005.

[11] T. Petermann and P. De Los Rios. Physical realizability of small-world networks. *Physical Review E*, 73(2):026114, February 2006.

[12] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

[13] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, February 1995.

[14] G. S. Snider and R. S. Williams. Nano/CMOS architectures using a field-programmable nanowire interconnect. *Nanotechnology*, 18(3):035204, January 2007.

[15] Synopsys. Design compiler user guide, June 2010.

[16] C. Teuscher, N. Gulbahce, and T. Rohlf. Assessing random dynamical network architectures for nanoelectronics. *IEEE/ACM Symposium on Nanoscale Architectures, NANOARCH 2008*, pages 16–23, June 2008.

[17] J. von Neumann. Theory of Self-Reproducing automata. *University of Illinois Press, Urbana, Illinois*, 1966.

[18] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.

[19] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311:419–424, October 1984.

[20] V. Zhirnov, R. Cavin, G. Leeming, and K. Galatsis. An assessment of integrated digital cellular automata architectures. *Computer*, 41(1):38–44, 2008.