NTNU
Norwegian University of
Science and Technology

TDT4900 Computer Science, Master's Thesis

Optimization of Seed Selection for Information Diffusion with High Level Synthesis

Julian Lam

Fall 2016

Department of Computer and Information Science
Norwegian University of Science and Technology

Supervisor 1: Donn Alexander Morrison
Supervisor 2: Yaman Umuroglu

0.1 Assignment

Information diffusion is a field of network research where a message, starting at a set of seed nodes, is propagated through the edges in a graph according to a simple model. Simulations are used to measure the coverage and speed of the diffusion and are useful in modelling a variety of phenomena such as the spread of disease, memes on the Internet, viral marketing and emergency messages in disaster scenarios.

The effectiveness of a given spreading model is dependent on the initially infected nodes, or seeds. Seed selection for an optimal spread is an NP hard problem and is normally approximated by selecting high-degree nodes or using heuristic methods such as discount-degree or choosing nodes at different levels of the k-core.

High-level synthesis (HLS) is becoming an important tool in the optimization/acceleration of algorithms in hardware. Starting with an algorithm written in a high-level language such as C or C++, HLS aids with hardware design by providing a methodology and tools that guide the developer through the design process.

This project should employ HLS as a design methodology for hardware accelerated seed selection in large graphs. The student will study seed selection for a given diffusion model, write a high-level model, and use HLS to implement a hardware design that exploits parallelism in the seed selection algorithm in order to improve performance over a GPCPU implementation. –

Abstract

Information Diffusion are often used for different simulations in network research because it simulates how information propagetes thorough a network, from memes on the Internett, spreading of disease in populations, to viral marketing. Measuring spread and speed, we can find influential targets in the network, such targets are optimal targets to pass message during disaster scenario, vaccinate to prevent spreading of a disease, or even targets for viral marketing.

High Level Synthesis have in recent years matured greatly. With HLS, designing custom architectures is no longer a

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Information diffusion is a field of network research where a message, or data, is propagated through a *network* or a *graph*. The message originates from a chosen set of nodes, known as *seed nodes*. These seed nodes passes the message to it's neighbour through the edges and thus propagate the message over theentire network. There are different models used in Information diffusion, *Independent Cascade Model*, and *Linear Threshold Model*. Information Diffusion can be used to model different phenomena such as the spread of disease, viral marketing, or even spread of viral videos and *"memes"*[2] [3]. The effectiveness of the simulation is measured in the spread and the speed of propagation. The effectiveness of the simulation is dependent on the chosen seed nodes. By finding the most optimal set of seed node, we can potentially stop an epidemic by vaccinating influential nodes, we can find important target for viral marketing by giving free sample, and use this information to quickly spread message during disaster scenarios.

There are multiple studies done regarding information diffusion, [4], [2], [5], [6]. There are few that focus on optimizing the seed selection, especially in hardware. Finding the most optimal set of seed node is useful in multiple fields. We prevent the spread of a disease by vaccinating influential nodes in the network, we can pass critical message through a population in disastrous scenario, or even find optimal target for viral marketing. The current seed selection algorithm is an greedy solution[7][DOUBLE CHECK THIS SOURCE], where every set of node is tested and the set with best coverage and time is chosen. This is a time consuming process and highly parallelizable. This makes it a good candidate for *Field-programmable gate arrays*(FPGAs).

High Level Syntesis(HLS) synthesis high level behaviour and constrains to lower level design.[8]. It allows users to implement an algorithm in high level language, C or C++, and generate an optimal design in *verilog* or *VHDL*. Verilog and VHDL are hardware descriptive language designed to describe digital

systems [9]. In recent years, High Level Synthesis have gotten more attention and more support, the xilinx forums are anwsered quickly by the developers and highly populated with seasoned hardware designers and novices.

Unlike traditional hardware design, HLS allows programmer with limited knowledge to desgn an optimal custom *Intellectual property core*(IP-core). In HLS, programmers can test out multiple different optimization schemes in short period of time. Thus allowing the programmer to quickly test out different optimization schemes.

For our implementation, we focused mainly on the ICM. The ICM is a special case of the common graph traversal algorithm *Breadth First Search*(bfs). For our implementation, we chose to implement the ICM as a custom *sparse matrix vector multiplication*(spmv). By performing ICM as spmv, we can utilize the parallelism options that spmv uncovers.

1.2 Assignment Interpretation

From the assignment text, these task were chosen as the main focus of this thesis:

Task 1 ***(mandatory)*** Implement Information Diffusion as Sparse matrix vector multiplication, with high level language C.

Task 2 ***(mandatory)*** Tailor the implementation of Information Diffusion for synthesise with Vivado HLS.

Task 3 ***(optional)*** Implement said design on a Zynq FPGA board.

Task 4 ***(optional)*** Extend the system to be able to handle graph in the size of toy graphs(containing 2^{26}) nodes)

1.3 Report Structure

We have here the basic outline for this report and a short overview of the remainder of this report:

Chapter 2: Background contains the information regarding network, Information diffusion, matrix vector multiplication and High level synthesis. Most of the background information regarding this report can be found in this chapter.

Chapter 3: Related Work shows what the related works and state of the art regarding information diffusion.

Chapter 4: Architecture

Chapter 5:.

Chapter 6: Future Work

Chapter 7: Conclusion Find something

Chapter 2

Background

In this chapter, we will look at the fundamental concepts and background information for the different diffusion model, the seed selection problem and performing breadth-first search using matrix multiplication. We will also have a look at High Level synthesis and Information about the zedboard. This chapter will contain notations that we will use throughout the report. One aspect we will focus on, is how we can perform graph algorithm such as breadth first search as matrix multiplication. The motivation for transforming breadth first search as matrix-vector multiplication is that displaying the graph algorithm as a matrix multiplication can display the data access pattern for the algorithm and can be readily optimized [10]. We will look at the independent cascade model, which is a special case of breadth first search [11]. By looking at how to improve BFS, we can apply such optimization to ICM and the seed selection algorithm. Another major part of the report is to utilize High Level Synthesis to synthesise the algorithm.

2.1 Network teminalogy and glossary

The fundamental unit in a network is a *vertex* (pl. vertices), sometimes called a node. For this report, both vertex and node will be used. The "bridge" or the line connecting two vertices is called an *edge*, and is shown in Figure 2.1. Different networks have different types of edges, some are *directed*, while others are *undirected*. A directed edge is an edge that runs in only one direction (such as a one-way road between two points), and undirected runs in both directions. Directed edges can be thought of as sporting arrows indicating their orientation, while undirected edges have no such orientation. A graph is directed if all of its edges are directed.

Each vertex has a value that is called *degree*. Degree for a vertex v_1 is the amount of edges connected to v_1. Note that the degree is not necessarily equal to the number of vertices adjacent to a vertex, since there may be more than

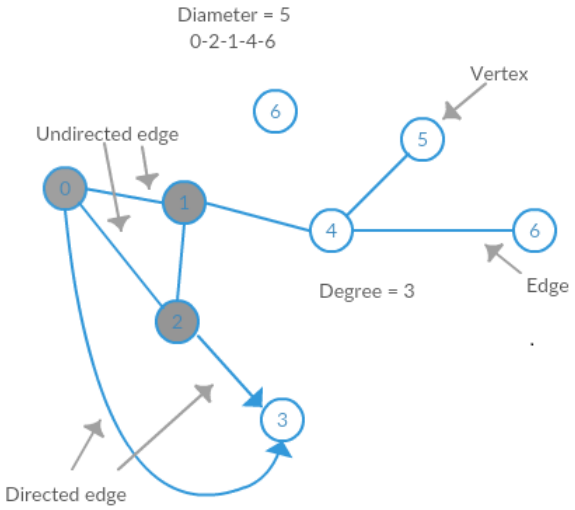


Figure 2.1: Simple network

one edge between any two vertices. A directed graph has both an in-degree and an out-degree for each vertex, which are the numbers of in-coming and out-going edges respectively. The *component* to which a vertex belongs is that set of vertices that can be reached from it by paths running along edges of the graph. In a directed graph a vertex has both an in-component and an out-component, which are the sets of vertices from which the vertex can be reached and which can be reached from it.

A *geodesic path* is the shortest path through the network from one vertex to another. Note that there may be and often is more than one geodesic path between two vertices. The *diameter* of a network, however, is the length (in number of edges) of the longest geodesic path between any two vertices. A few authors have also used this term to describe the average geodesic distance in a graph, although strictly the two quantities are quite distinct.

2.2 Network

A *network* is a collection of vertices and edges [12]. In the real world, multiple systems takes the form of networks around the world, examples are the internet, the World Wide Web, social media like Facebook, Twitter etc. There are different types of network. These include *social network* [12], *information networks* [12], *technological networks* [12] ,and *biological network* [12]. Different network have different properties and in this report we will look at those most relevant to social networks.

2.2.1 Social network

A social network is a set of individuals connected to each other via some form of contact or interactions [12]. The nodes are people while the edges are the connections between people. The social network display information regarding connection, interaction or location of a set of people.It forms patterns regarding friendships, business interactions between companies and families history. The social network is often used in social science [12]. Some notable experiments are the letter passing experiment [13], which we will discuss more in 2.3.1

2.3 Network properties

Network properties are different characteristic properties that networks displays. We will focus on those that are more relevant to social networks and how it is relevant to the ICM, data diffusion and seed selection.

2.3.1 The small world effect

The small world effect was first demonstrated by Stanley Milgram in the 1960s during his famous letter passing experiment [14]. The experiment involved passing letters from person to person to reach a designated target with only small steps. For the published case, the chain was around six[15], meaning there were only six passes necessary for the letter to reach its destination. This shows us that for most pairs of vertices in a network can reach each other with a short path. A more precise wording is that "Networks are said to show the small world effect if the value of l scales logarithmic or slower with the network size for a fixed mean degree." [12]. We have defined l to be the mean geodesic distance between vertex pairs in a network.

The small world problem can be summarized as: "what is the probability that any two people, selected arbitrarily from a large population, such as that of the United States, will know each other?" [15]. This in itself is not that interesting, the paper [15] asked even though person A and$Z$ does not know each other, do they have a set of individuals $\{b_1, b_2, ...b_n\}$ who are mutual friend or even a "chain" of such individual(A-B-C-...-Y-Z).

For the data diffusion problem, this kind of effect would result in that the diffusion through a network would need around 6 steps to have traveled through the entire network if the transition probability is high. Meaning that most node can reach each other through a relatively small step.

2.3.2 Transitivity/Clustering

In graphs and networks, there would often have a special connection pattern called *triangles*. Triangles is where three Vertices : v_a, v_b, v_c is all connected to each other as shown in Figure:2.2. We can look at such a connection as person A is friends with B and C. There are a chance that B and C are friends with

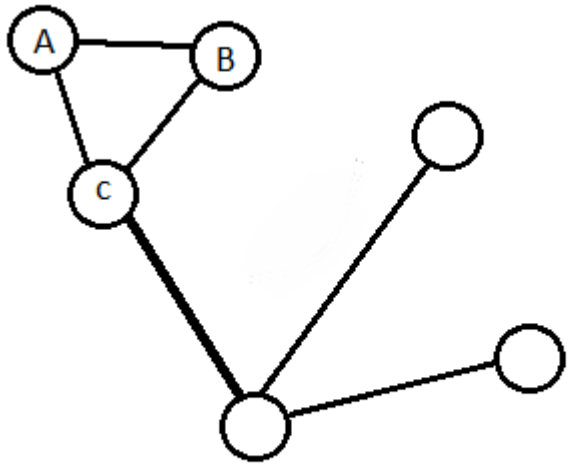


Figure 2.2: Examples of triangles in networks

each other too. Transitivity is used to determent how many such components is present in the graph.

For data diffusion and seed selection, this would mean that picking nodes that are neighbor to each other, would have a smaller spread then picking two not neighboring vertices. By picking two nodes that are connected to each other, they would likely share common neighbors and thus having a smaller reach.

2.3.3 Degree distribution

As mentioned earlier, each node have a degree. The degree distribution shows how the different degrees in the network is distributed. From the degree distribution we can see how many nodes have specific degree. Different networks have different shaped degree distribution.

For random graphs, the distribution would most likely be a Poiison distribution or binomial. For social graph, the degree distribution is often in the shape of exponential distribution. For the information diffusion, this distribution shows us how many high degree nodes there are in the network and those node would likely be high prioritized node. One of the algorithm uses the distribution to select the seed nodes. We will discuss this in later sections.

2.3.4 Degree correlations

As mentioned in section 2.3.3, the network have a degree distribution with few high degree nodes and many low degree nodes. One interesting properties is the degree correlations. Degree correlations is how the high degree and low degree nodes connects to each other. One question is wether high degree nodes tends to connect to other high degree nodes, or if they prefer to connect to low degree nodes. It turns out that both incidents are found in networks [12]. Most social networks are assortative, meaning vertices have a selective linking, where high degree vertex tends to connects to other high degree vertex, while technological network and biological network are most likely disassortative [16].

For data diffusion, this kind of behavior would result in wasting a seed by picking a majority of high degree nodes for starting seed, since most of them would be connected to each other. One solution is by mixing the selection, choose some percentage to be high degree, and some with lower degree.

2.3.5 Network resilience

For most network model, there is the need to remove nodes from the network. Removal of a node can have no effect on the network, or it can be devastating. Network resilience looks at how the network can resist to such a removal. There are two different removal schemes, the random removal where nodes are randomly picked and removed, or the targeted removal where specific nodes are removed depending on the criteria.

The experiment mentioned in [12] by Albert et al showed that for a subset of network representing the internet and the world wide web, targeted removal had a larger impact then random removal. The targeted removal removed the highest degree nodes from the network, and the random removal removed nodes randomly. The random removal had a minimal effect on the network, while the targeted removal had a much larger impact, in which mean vertex-vertex distance increased. They proposed that the internet was highly resilient to random removal, while much more vulnerable to targeted removal.

There are other studies that proposed a different interpretation regarding the data found.

In an example of information diffusion, a targeted removal would result in massive change to the diffusion path. By removing high degree node it would result in remove influential nodes and limit the spread of the data. Removing important nodes connectiong two different communities would result in isolation and no path towards other communities. For our case, we assume that the network is a static, unchanging network.

2.3.6 Community structure

One properties that is often observed in a social network, is the community structure. The community structure is where a group of vertices having high density of edges with each other, while having low density of edges to other

"community". We can see an example of the community structure clearly displayed from [17]. Where we can see the playground was divided into different groups.

This type of community structure would have a large impact on how the algorithm would select a seed for information diffusion. If all the seed would be selected in one community, the probability of spreading over to other communities would be smaller. This is of course dependent on how many intercommunity connections there are in the network.

2.4 Breadth First search

Breadth first search is an tree traversal algorithm. BFS start at the root node v_r. The algorithm then stores all v_rs children node in an *queue*. The algorithm then takes the first node from the queue, v_1 and stores all the children node to v_1 in the back of the queue, this process continues until the queue is empty and all the nodes have been iterated over.

Breadth first search is a common graph iteration algorithm. The breadth first search is often limited by the irregular memory access where the algorithm have to find the data stored in different spaces in the memory. As mentioned earlier, independent cascade model is one special case of the Breadth first search. Breadth first search is the independent cascade model with a 100% chanche to activate the neighbores.

Algorithm 1 Breadth First Search

```
1:  dist[∀v ∈ V] = −1; currentQ, nextQ = ∅
2:  step = 0; dist[root] = step
3:  ENQUEUE(nextQ,root)
4:  while nextQ ≠ ∅ do
5:      currentQ = newxtQ; nextQ = ∅
6:      step = step + 1
7:      while currentQ ≠ ∅ do
8:          u = DEQUEUE(currentQ)
9:          for v ∈ Adj[u] do
10:             if dist[v] == −1 then
11:                 dist[v] = step
12:                 ENQUEUE(nextQ, v)
    return dist
```

2.5 Data diffusion

Data diffusion is looking at how information is propagated through a network or a graph. Some example would be how a new Internet meme, a new product or how a new disease is spread through a community. The process consist of a set

of starter nodes, which we will call seed nodes, that are "infected". During each time-step, there are a percentage p_g that the "infected" nodes would "infect" its neighbors. Seed nodes is a set of k nodes that in the initial time-step are infected. They will pass on the information/infection during each time-step and the information/infection will propagate through the network.

2.6 Basic Diffusion Models

When we talk about data diffusion, we can look at how diseases or technological innovations would spread through a social network. We can simulate those kind of behaviors with different diffusion models. There are two basic diffusion models used to simulate the propagation of information through a network [18], the *linear threshold model*(LTM) and the *independent cascade model*(ICM) [18].

This process is similar to how a new product is promoted via social media. Each node is a person that can either buy the new product(activated), or ignore it(inactive). Each person will then see their friends promote the new product and potentially buy the promoted product. There are several different criteria for each person to buy the product(activates). They can either have a percentage chance to be affected by the advertisement(ICM), or they will only be interested if a percentage of his or hers friends have promoted it(LTM). Some people might have a larger circle of friends then others(high degree), while others have larger impact on a person(large p_x). Some might be harder to promote too(weighted edges), while some users have no friends(singletons). The different models we will focus on, are the independent cascade model and the linear threshold model.

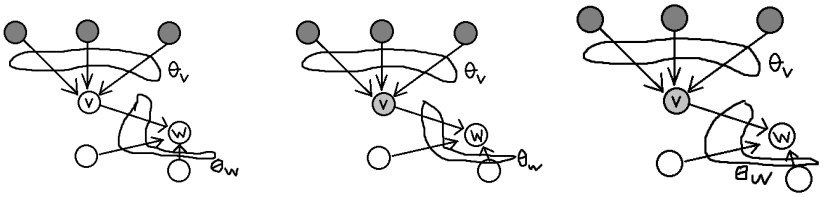
2.7 Linear threshold model

The linear threshold model uses a threshold θ_v between the interval [0,1], which represent the fraction of $v's$ neighbors that need to be active to activate node v, this is known as the weight of v, b_v. In this case, let's assume that the weight of all the nodes in this model are 0.5, meaning over half of its neighbors must be activated for the node to be activated. As we can see in Figure 2.3a, current node v will get activated when $b_v <= \theta_v$. In figure **??** we can see that v is now activated. The next time-step, Figure 2.3c, node w is checking if it will too, be activated. We can see here that $b_w! <= \theta_v$, so node w will not be activated.

We can look at the linear threshold model as a cosmetic company trying to promote their new product via social media. Each users of the product would display the new cosmetic product through social media. Each users would then be exposed to the product through their friends update. Each user would adopt the new product after seeing a percentage of their friends using the product.

2.7.1 Independent cascade model

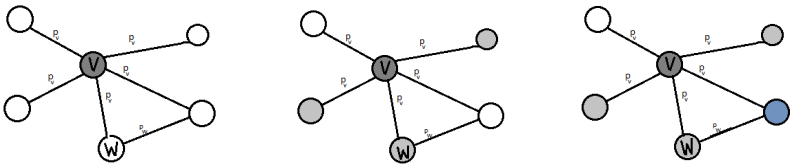
The independent cascade model(ICM) have a local or global probability for determining of activation, p_v. In figure 2.4, we have private probability. The



(a) Step 1 (b) step 2 (c) step 3

Figure 2.3: Linear Threshold mode

probability $p_v! = p_w$. In 2.4a, the node v is activated, during the next time-step, node v have activated three neighbors. In the next time-step 2.4c node w was able to activate the blue node. For a ICM with global probability, each node would then have the same probability to activate its neighbor. As we can see from Figure 2.4, each neighbor to v is activated individually. As in 2.4b, only three nodes were activated. In ICM, each node can only try to activate the neighbor once. In figure 2.4b, the node that was not activated by v, got activated in 2.4c by the node w.

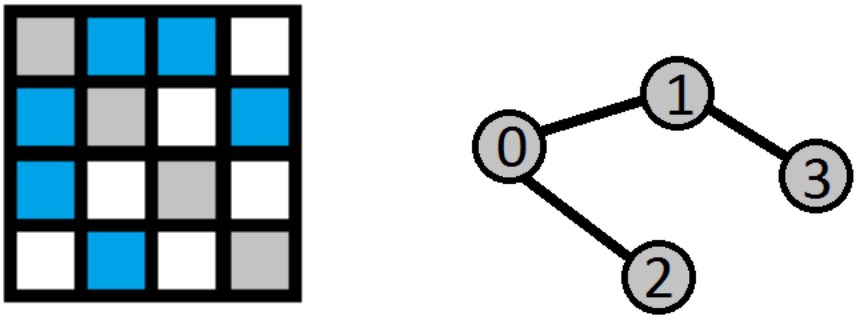


(a) step 1 (b) step 2 (c) step 3

Figure 2.4: Independent cascade model

We can compare the ICM with the same example we have been using, the cosmetic company promoting a product. Each new user have a chance(pp_v) to promote the product to a new user. The new user would then continue promoting the new product to his/hers friends.

As mentioned earlier, the ICM is a special case of the breadth first search. If we set the transmisison probability to 1.0, meaning there are 100% chance to activate the neighbores, the ICM is equal to BFS. BFS iterates through the graph by appending the child node not examined into the queue. Then examines a new node from the queue and repeating the process until all nodes have been examined. The main difference between the ICM and BFS, is that in ICM, there is a chance that the child node is not added into the queue. During each step, a random "coin toss" is tested to determent if the child node would be added into the queue. Other then that, both algorithms iterates through the network via edges.



(b) The graph corresponding to the adjacency matrix

(a) The adjacency matrix

Figure 2.5: Sparse matrix to graph

2.8 Matrix notations

Nodes and edges are not the only way to present a graph, graphs and networks can be represented as sparse adjacency matrices [10]. We can see that such an idea have been proposed in earlier literature [19]. By representing graphs as a sparse matrix, we can often discover different ways to optimize the algorithm, we can have a different structure to store data. The adjacency matrix in particular, is a interesting way to represent the graph. A graph $G = (V, E)$, G have N vertices and M edges, this correspond to a N×N adjacency matrix called A. If A(i,j)=1, then there is an edge from v_i to v_j. Otherwise its 0. In Figure:**??**, we can see how a undirected graph can be represented as an adjacency matrix. To generate a undirected graph as a adjacency matrix, the matrix must be mirrored diagonally, meaning if A(i,j)=1, then A(j,i)=1, if this is not true, then the matrix would be representing a directed graph.

2.8.1 Sparse Matrix

A sparse matrix is a matrix containing few nonzero. Social graph with few edges would often be represented as a sparse matrix. Since sparse matricies only have few non-zero elements, by storing only the non-zero elements, we can have savings in memory.

2.8.2 Breadth First Search as a matrix multiplication.

From [10], we can see that BFS can be recast as algebraic operations. BFS can be performed by applying matrix-vector multiplication over Boolean semirings [11]. The graph is represented as a adjacency matrix A, then for the root node,

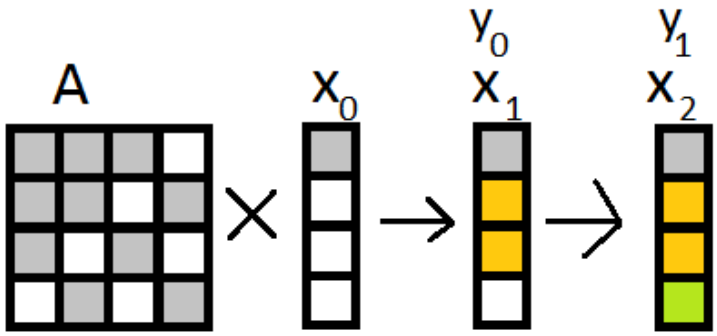


Figure 2.6: BFS on Boolean semiring

a vector x(root)=1 is multiplied with the matrix A. $A \times x_0 = y_0$. y_0 is the result after the first matrix-vector multiplication and in the next iteration, $x_1 = y_0$. We can see from the Figure2.6

2.8.3 Semiring

A *semiring* is a set of elements with two binary operations. The two operations are often known as "addition"(+) and "multiplication"(×). As we shown in previous section, the algorithm perform matrix multiplication uses the two operations, multiplications and addition. In [11], the AND and OR operator was chosen instead of the normal addition and multiplication.

2.8.4 BFS to data diffusion

As mentioned before, ICM is a special case of the breadth first search. By modifying the algorithm proposed earlier, we can in theory perform ICM with matrix-vector multiplication.

2.9 Seed selection algorithm

The seed selection algorithm, is the algorithm used to select the initial k seed nodes to be chosen at the start of the information diffusion. Each selected nodes is in the initial timestep activated. During each timestep, the seed nodes will propagate the activation along the network depend on what diffusion model is used. We can compare it to a new gadget or a cosmetic company trying to

promote a new product. By selecting a few influential persons to give a free sample, the new trend would most likely spread through *viral marketing* [20]. The seed selection algorithm would be the algorithm to select the few influential individuals to receive this free sample. There are multiple different scheme to choose from, in this section, we will focus on four different algorithms, greedy algorithm, degree algorithm, random algorithm and the independent greedy algorithm.

2.9.1 The greedy algorithm

The greedy algorithm [7] proposed by Kempe et al, is known to be the best algorithm according to the result from [7]. The greedy algotihm starts by itterate over the entire network and finds one node that have the largest coverage and stores that node in the set A. The algorith then activates that node from A, and computes the coverage of each node in the network with the previous choosen node. After the algorithm finds the second node, that node is stored in A with the previous node and we have now choosen two seed node. The algorithm continues until we have choosen k seed nodes, where each have been tested to have maximum coverage in relation with the previous choosen nodes. To compute the maximum coverage, the algorithm have to test every combination of nodes together, this results in heavy computation and the algorithm would therefore not scale well.

We can look at the greedy algorithm as a special case of BFS with a transistion probability. Each node in the network will be choosen as the seed node for the

Algorithm 2 Greedy Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to approximate $\sigma(A \cup x)$ to within $(1 \pm \varepsilon)$ with probability 1δ
4: Add the node with largest estimate for $\sigma(A \cup x)$ to A.
5: Output the set A of nodes.

2.9.2 The degree algorithm

Another popular algorithm is the degree algorithm [7]. Unlike the greedy algorithm, does not compute the coverage of node, the algorithm picks the top k nodes according to the degree distribution instead. The node chooses the top k nodes with the highest degree and stores them as the seed nodes. This approach benefits over the greedy algorithm by not having as much computation time as the greedy algorithm since only one itteration is needed to compute the degree to node. The disadvantage is that this algorithm does not take the degree correlation into acount. As mentioned in section 2.3.4, high degree nodes would

often have common node as neighbor. This would result in multiple overlapping activated node choosen.

Algorithm 3 Degree Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to compute DegreeMax(x).
4: Add the node with largest degree to A.
5: Output the set A of nodes.

2.9.3 Independent algorithm

Another algorithm is the independent greedy algorithm. The algorithm iterates through the network, computing the spread of each node. The algorithm then chooses the vertex with the largest coverage independent of the other previous chosen nodes. This algorithm is a special case of the greedy algorithm mentioned above.

Algorithm 4 Independent Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to approximate $\sigma(A \cup x)$ to within $(1 \pm \varepsilon)$ with probability 1δ
4: Add the node with largest estimate for $\sigma(x)$ to A.
5: Output the set A of nodes.

2.9.4 Random algorithm

The last one is the random algorithm. The random algorithm just picks a random seed node. This approach is the simplest to implement and easiest. The downside is that this is random and there are no strategic choosing of seed node.

2.10 RMat

One problem during graph analyzation and calculation is finding suitable graphs to analyses. Generate graphs with desired properties is not easy to do. One solution proposed by Chakrabartiy et al is to use the "recursive matrix" or R-mat model. The R-mat model generates graph with only a few parameters, the generated graph will naturally have the small world properties and follows the laws of normal graphs, and have a quick generation speed [1]. The R-mat models goal is to generate graphs that matches the degree distribution, exhibits

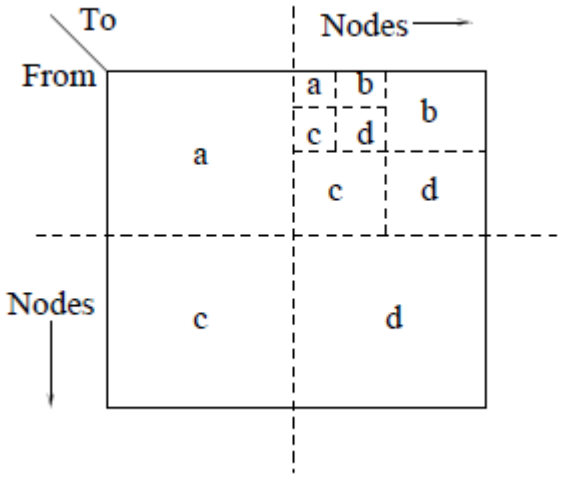


Figure 2.7: The R-mat model [1]

a " community " structure and have a small diameter and matches other criteria. [1]. The algorithm to generate such a recursive matrix is as follows: The idea is to partition the adjacency matrix into four equally sized part branded A,B,C,D, as shown in Figure2.8. The adjacency matrix starts by having all element set to 0. Each new edge is "dropped" onto the adjacency matrix. Which section the edge would be placed in, is chosen randomly. Each section have a probability of a, b, c, d, and $a + b + c + d = 1$. After a section is chosen, the partition that was chosen is partitioned again. This continues until the chosen section is a 1x1 square and the edge is dropped there. From the algorithm, we can see that the R-mat generator are capable to generate graphs with total numbers of node $V = 2^x$. Since the algorithm partitioned the matrix into four part. This is approach would only generate a directed graph. To generate undirected graph, $b = c$ and the adjacency matrix must make a "copy flip" on the diagonal elements, like Figure 2.8.

1 what is Information diffusion

what is the solution(bfs)

1 what is ICM

1 how to implement bfs as SpMV

1 how to apply information diffusion to bfs and problem

1

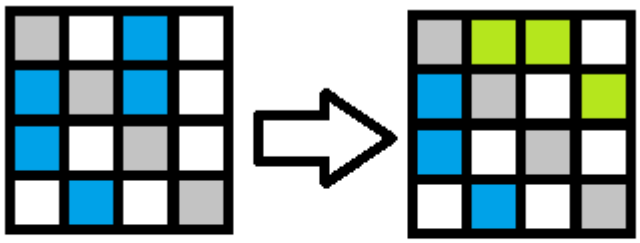


Figure 2.8: How the adjacency matrix is flipped on the diagonal

2.11 High Level Synthesis

[21] High Level Synthesis converst algorithms implemented on higher level down to *Register Transfer Level*. It converst C and C++ codes to Verilog and VHDL. The concept of HLS was first proposed in the mid-1990 to 2000[21]. The poor Quality of result, and lacing of tools,

2.12 Memory Mapped axi interface

Something something mapped to memory.

2.13 Cache coherency

Cache coherency, which is where multiple cache reference the same area.

2.14 code

balkdf

2.15 ZedBoard

We choose to implement on the zedobard

Chapter 3

Related Work

Here, we will give you a short overview of the current state regarding Information Diffusion, High Level Synthesis and different optimization options.

- Yamans paper, where there are some works that shows the solution i use
- parallalization of the algorithm
- maybe some examples of HLS to show that HLS is used.
- showe that there are not many HLS implementation, recently matured.
- show that there are not many hardware implementation for information diffusion.
- need to look through Yamans paper and get some refrences from there.
- might be good to look at how this type of sparce matrix multiplication can be used
- show other implementation of SPmv
- Show some examples where image processing is done through vivado HLS.
- [21] A good paper showing the state of the art for HLS.

This chapter, we will look at the state of research regarding High Level Synthesis, network research regarding Information Diffusion, and Optimization of Independent cascade model and Breadth first search.

3.1 Information Diffusion

There are multiple studies done regarding Information Diffusion. One studies shows how information diffusion can be applied during an disease outbreak[2], viral marketing[20], coordinat during crisis situation[22].

Models of influence have been done on blogs[23][24], and twitter[25]. We can see that in an age of social media, the studies of information diffusion is more relevant then ever.

while other[6] have argued that the emerging of social network and media, have changed the traditional model. The activation is no longer only relying on neighbour nodes, but also an external influence. They found that large amount of information volume in Twitter is the result of network diffusion, while a small amount is due to external events and factors outside the network[6]. Another studies shows during the 2011 Egyptian Uprising, how larg amount of such a movement were "tweeted"[22].

As we mentioned in chapter 2, we mainly focus on 2 common information diffusion models, ICM and LTM. But there are different models too. One report[5] proposed several different problems with traditional models where each node is either *activated*(infected, influenced, '1') or *inactive*(healthy, not reached, '0'), and passes the *contagion*(information, data, infection, influence) to a neighbouring nodes through the edges. The report mentioned different assumptions that such models take. Among them is that a complete graph is provided, the spread of contagion is from a known source, and that the structure in the network is sufficient to explain the the behaviour[5]. The report propose an alternative model, *Linear Influence Model*(LIM), where the focus is on the global influence a infected node has on the rate of diffusion through the implicit network. This model takes the assumptions, that newly activated nodes is dependent on previous activated nodes. The LIM does not need explicit knowledge of the entire network, instead the model takes the newly activated nodes and model them as a *influence function*, which is used to find the global influence.

3.2 High Level Synthesis

High Level Synthesis as a concept have been around since the mid-1980s and early-1990s. Early tools, known as Carnegie-Mellon University design automation (CMU-DA)[26][27] was a pioneering early version of HLS tool. The tool gathered quickly considerable interest. A number of HLS tools were built in later year mostly for prototyping and research[28][29][30]. Some of these early tools was able to produce real chips, but the reason for lack of further development and adaptation, was that RTL synthesis was not a widely accepted and immature field. This often lead to suboptimal solutions.

In the 2000, new HLS tools was developed in academia and in the industry. These tools, used hihg level language, C and C++. Vivado HLS, designed by Xilinx [31], is one such HLS tool. The Vivado HLS became free during their 2015.4 update[32]. This resulted in an revived interest in HLS. The community around HLS is also evolving, on the Xilinx-forum, there are multiple anwsers and active members. We can see that the solution designed by HLS tools is close to traditional hand-crafted designs[33].

3.2.1 Different HLS implementations

in [34], HLS was used to design an accelerator for database analytic and SQL operation. The design was implemented on a Virtex-7 xc7vx690t-g1761-2 FPGA with focus on accelerating operations as join, data filter, sort, merge and string matching. The accelerator was implemented in C++ in Vivado HLS and optimized with UNROLL directive, PIPELINE directive and ARRAY_PARTITION. The UNROLL directive unroll all of the specified loops, while the PIPELINE directive allows multiple accelerator to process data at each clock cycle. The ARRAY_PARTITION directive partition data into registers. The accelerator showed promises, giving a 15-140× speedup compared to Postgres software DBMS running selected TPCH queries.

[35] explored the advanteage and disadvanteges of HLS implementation of image processing. Here, they argued that, most often, custom algorithm on FPGA platform will result in an improvement, but simply porting an existing algotrithm might not be a improvement. Here, the author conducted different case studies to show both the strength and the weakness with HLS. The report goes through image filtering, connected component analysis and two dimensional FFT. One example that the authore brings up, is during image filtering. The HLS was not able to identify the standard accessing pattern during special cases and resulted in that the HLS built additional hardware to counter such a exception. The report conclude with that while HLS can significantly reduce development time and improve utilization of design space, it is still important to focus on careful design. By recompiling the algorithm in HLS would result in an suboptimal solution, the designers would need to customize the algorithm for the HLS to optimize for FPGA implementation. The report concludes that HLS can offer many benefits, and is an improvement over conventional RTL-designs, but is not an replacement for hardware designers or clever designs.

[36] implemented an fast Fourier transform (FFT) algorithm for different digital system processing application in HLS. There teh authors used Simulink for vcerification of design, and implemented it in HLS.

[37] discuss improvements to the current HLS tools with polyhedral transformation. Here they discuss a problem with HLS, which is that unless the code is inherently compatible, HLS can't apply most of the optimizations. Zuo et al. proposed the polyhedral model, the model takes data/dependent multi-block program as input and performes 3 steps: Classification of array access pattern, performance Metric, and implementation. During the classification of array access pattern, a set of data access pattern is defined and classified, then the appropiate loop transformation is applied. The next step, the performance of each loop transformation with data-dependency is estimated, and the best improvement is chosen. The last step, the chosen solution, loop transformation and inserting HLS directive is applied. Then a interface block for the data-dependent blocks is generated. The generated communicatiion block is then optimized depending on how it behaves. The paper concludes with that the poluhedral model can model can find important loop transformation, thus enable optimization such as pipeline and parallelization.

3.3 Different optimization scheme

Chapter 4

Result

as we can see, the algorithm was able to finish a

Chapter 5

Conclusion

Bibliography

[1] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. 4:442–446, 2004.

[2] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 491–501, New York, NY, USA, 2004. ACM.

[3] Daniel M. Romero, Brendan Meeder, and Jon Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 695–704, New York, NY, USA, 2011. ACM.

[4] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and P Krishna Gummadi. Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10(10-17):30, 2010.

[5] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *2010 IEEE International Conference on Data Mining*, pages 599–608, Dec 2010.

[6] Seth A. Myers, Chenguang Zhu, and Jure Leskovec. Information diffusion and external influence in networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 33–41, New York, NY, USA, 2012. ACM.

[7] David Kempe, Jon Kleinberg, and va Tardos. Influential nodes in a diffusion model for social networks. 3580:1127–1138, 2005.

[8] M. C. McFarland, A. C. Parker, and R. Camposano. The high-level synthesis of digital systems. *Proceedings of the IEEE*, 78(2):301–318, Feb 1990.

[9] Donald Thomas and Philip Moorby. *The Verilog® Hardware Description Language*. Springer Science & Business Media, 2008.

[10] Jeremy Kepner and John Gilbert. *Graph algorithms in the language of linear algebra*, volume 22. SIAM, 2011.

[11] Y. Umuroglu, D. Morrison, and M. Jahre. Hybrid breadth-first search on a single-chip fpga-cpu heterogeneous platform. pages 1–8, Sept 2015.

[12] M. E. J. Newman. *The Structure and Function of Complex Networks*, volume 45. 2003.

[13] Stanley Milgram Jeffrey Travers. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[14] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[15] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, pages 425–443, 1969.

[16] Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[17] James Moody. Race, school integration, and friendship segregation in america1. *American journal of Sociology*, 107(3):679–716, 2001.

[18] Éva Tardos David Kampe, Jon Klein. Maximizing the spread of influence through a social network. pages 137–146, 2003.

[19] MH McAndrew. On the product of directed graphs. *Proceedings of the American Mathematical Society*, 14(4):600–606, 1963.

[20] Pedro Domingos and Matt Richardson. Mining the network value of customers. pages 57–66, 2001.

[21] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang. High-level synthesis for fpgas: From prototyping to deployment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(4):473–491, April 2011.

[22] Kate Starbird and Leysia Palen. (how) will the revolution be retweeted?: Information diffusion and the 2011 egyptian uprising. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 7–16, New York, NY, USA, 2012. ACM.

[23] Eytan Adar and Lada A. Adamic. Tracking information epidemics in blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, pages 207–214, Washington, DC, USA, 2005. IEEE Computer Society.

[24] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1019–1028, New York, NY, USA, 2010. ACM.

[25] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone's an influencer: Quantifying influence on twitter. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 65–74, New York, NY, USA, 2011. ACM.

[26] S. Director, A. Parker, D. Siewiorek, and D. Thomas. A design methodology and computer aids for digital vlsi systems. *IEEE Transactions on Circuits and Systems*, 28(7):634–645, Jul 1981.

[27] A. Parker, D. Thomas, D. Siewiorek, M. Barbacci, L. Hafer, G. Leive, and J. Kim. The cmu design automation system: An example of automated data path design. In *Proceedings of the 16th Design Automation Conference*, DAC '79, pages 73–80, Piscataway, NJ, USA, 1979. IEEE Press.

[28] John Granacki, David Knapp, and Alice Parker. The adam advanced design automation system: Overview, planner and natural language interface. In *Proceedings of the 22Nd ACM/IEEE Design Automation Conference*, DAC '85, pages 727–730, Piscataway, NJ, USA, 1985. IEEE Press.

[29] P. G. Paulin, J. P. Knight, and E. F. Girczyc. Hal: A multi-paradigm approach to automatic data path synthesis. In *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, DAC '86, pages 263–270, Piscataway, NJ, USA, 1986. IEEE Press.

[30] H. D. Man, J. Rabaey, P. Six, and L. Claesen. Cathedral-ii: A silicon compiler for digital signal processing. *IEEE Design Test of Computers*, 3(6):13–25, Dec 1986.

[31] D. Navarro, Luca, L. A. Barragn, I. Urriza, and Jimnez. High-level synthesis for accelerating the fpga implementation of computationally demanding control algorithms for power converters. *IEEE Transactions on Industrial Informatics*, 9(3):1371–1379, Aug 2013.

[32] XIlinx. Vivado Design Suite User Guide release notes, installation, and licensing,ug973 (v2015.4), 2015.

[33] F. Winterstein, S. Bayliss, and G. A. Constantinides. High-level synthesis of dynamic data structures: A case study using vivado hls. In *Field-Programmable Technology (FPT), 2013 International Conference on*, pages 362–365, Dec 2013.

[34] Gorker Alp Malazgirt, Nehir Sonmez, Arda Yurdakul, Adrian Cristal, and Osman Unsal. High level synthesis based hardware accelerator design for processing sql queries. In *Proceedings of the 12th FPGAworld Conference 2015*, FPGAworld '15, pages 27–32, New York, NY, USA, 2015. ACM.

[35] Donald G. Bailey. The advantages and limitations of high level synthesis for fpga based image processing. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, ICDSC '15, pages 134–139, New York, NY, USA, 2015. ACM.

[36] Shahzad Ahmad Butt, Mehdi Roozmeh, and Luciano Lavagno. Designing parameterizable hardware ips in a model-based design environment for high-level synthesis. *ACM Trans. Embed. Comput. Syst.*, 15(2):32:1–32:28, February 2016.

[37] Wei Zuo, Yun Liang, Peng Li, Kyle Rupnow, Deming Chen, and Jason Cong. Improving high level synthesis optimization opportunity through polyhedral transformations. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '13, pages 9–18, New York, NY, USA, 2013. ACM.