8) *Infrasound:* They are audio frequencies below threshold of hearing of human ears (1 to .001Hz) radiated from many geophysical processes. These frequencies below 20Hz can define the range of infrasound and designated as acoustic and gravity waves. Sodar echograms can easily distinguish signatures of sinusoidal, conical (thermal convection) or other complex natural processes.

   A network of suitably placed 3 acoustic sounders in a earthquake prone area can determine the direction of wave propagation which will help in locating the epicentre.

9) *VLF and ULF Electromagnetic Emissions:* The Electromagnetic (EM) emissions by seismic events will help to know the earthquake events prior to their rupture for short term earthquake prediction.

   Measurement of electromagnetic phenomenon can be classified in 3 types: 1) The passive ground based observation for lithospheric emissions, 2) The ground based observation with use of transmitter signals for study of seismic atmospheric and ionospheric perturbations, 3) The satellite observations.

   A comparison between seismic and electromagnetic data will give understanding of fundamental of physics of the earthquake preparation process.

   A network of conventional magnetic field detectors spaced less than 100 kms apart would be required to detect ULF magnetic field fluctuations prior to the earthquakes with magnitude greater than 7.

   Under some conditions superconductivity magnetic field gradiometers could offer greater sensitivity and range. Employing bore holes and terrestrial antennas, VLF electric field perturbations can be monitored.

10. *Thermal Anomalies:* Earth's crust passes through an Earthquake preparatory phase before imminent earthquake. Accumulation of stresses and resultant pressure builds up loads to rise of inland surfaces temperature (LST). The enhanced TIR (Thermal infrared) emission from Earth's surface retrieved by satellites prior to earthquake is called 'Thermal Anomaly'.

    Statistical processing of data of GPS received from network, together with various other atmospheric parameters demonstrate the possibility of an early warning of an impending strong earthquake.

IV. REGION OF RESEARCH

The F region ionosphere or ionospheric parameters that offer the precursory signatures of major impending earthquakes. The F layer is the closest to sun and is always ionized irrespective of hours of the day as well as season of the year. This ionization density is affected largely by earth's magnetic field, winds, storm and ionospheric tides.

There is an NPL network of Digital Ionosonde systems for extensive measurements of f_0F2, hmF2, VHF and UHF to continuously monitor different ionospheric layer. This information can be used to figure out a pattern that would be of help to find earthquake precursors.

Atmospheric dynamics in lower planetary boundary layer (LPBL) or planetary boundary layer (PBL) that is one kilometre of atmospheric environment, adjacent to the earth's surface that transfer momentum and heat energy from ground to higher levels.

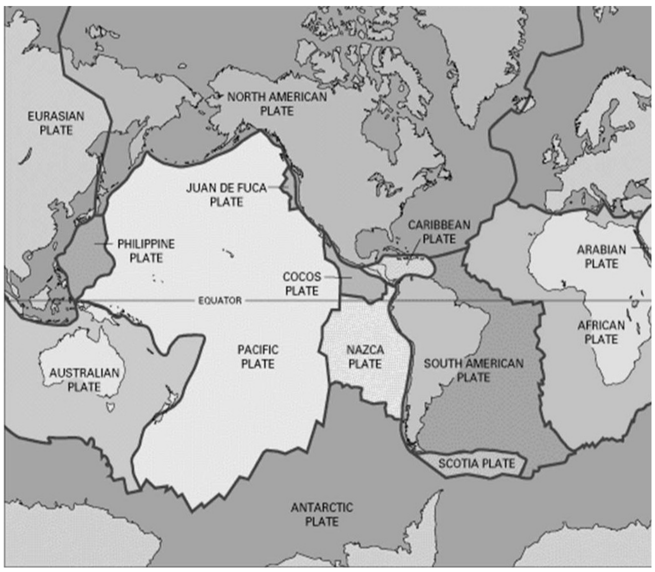

Figure 1. Tectonic plates of the Earth crust. (Source: http://pubs.usgs.gov/gip/dynamic/slabs.html[1])

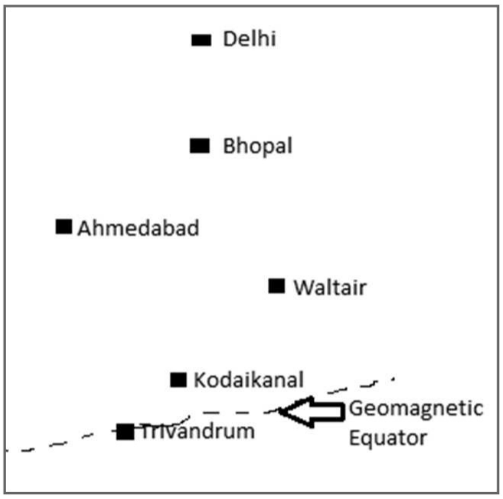

Figure 2. Network of digital ionosondes.

Sodar or acoustic radar model can be used to record basic types of single layer lower atmospheric structures like convective plumes, inversion with flattop, inversion with small spikes, inversion with tall spikes, rising inversion, rising inversion with convective plumes and many others depending upon type of observations and experts.

V. FUTURE SCOPE

Classification of patterns produced by sodar models with any pattern recognition technique so that the dependency on scientists to analyse the results for predicting the earthquake can be minimized.

VI. REFERENCES

[1]. "A Connectionist Approach to Sodar Pattern Classification", *IEEE Geoscience and Remote Sensing Letters,* Vol. 1, No. 2, April 2004.

[2]. "Unique Atmospheric Wave: Precursor to 26 January 2001 Bhuj, India earthquake", *International Journal of Remote Sensing.*

[3]. "Ionospheric Perturbations over Delhi Caused by the December 26, 2004 Sumatra Earthquake".

Mohini Preetam Singh received Bachelor's degree in Electronics and Instrumentation from UPTU, India and Master's degree in Microelectronics from Subharti University, India. She received Academic Excellence award for research work in "NEGF Approach in Silicon Nanowire Transistors" in her Master's. She is pursuing PhD (Part time) in Electronics and Communication from Amity University, Noida. She is working as an Assistant Professor in Vidya College of Engineering since 2010. Her current work focus is on "Pattern Recognition of thermal anomalies".

News Aggregation in Python Using Hierarchical Clustering

Rahul S Verma, Satyam Gupta and **Shivangi**
CSE Department, IMS Engineering College, NH-24, Near Dasna, Adhyatmik Nagar, Ghaziabad, 201009 UP India
rahul.1a94@gmail.com, satyam905@gmail.com, bitts.beans@gmail.com

Abstract: **In this paper we illustrate a way to cluster similar news articles based on their term frequency. We use python and nltk to recognize keywords and subsequently use hierarchical clustering algorithm. This method can be used to build news aggregation backends. Aggregation means clustering like documents from different sources. There is fast moving data and heterogeneity of sources in news aggregation scenarios. We need to remove the duplicates arising due to heterogeneous sources.**

Keywords: Python, nltk, *Feedparser, News Aggregation. Hierarchical Clustering, Algorithms, Aggregation, News, Text Mining*

I. INTRODUCTION

NEWS Aggregators can be considered a multilateral platform of interconnection [1]. In principle, news aggregators can be a substitute or a complement to the news outlets who invest in the creation of news stories. A policy debate centers around the decrease in the incentives for news creation that results if readers choose to consume their news through aggregators without clicking through to the news websites or generating any revenue for the outlets [2].

Getting these two ideas in perspective our idea is to get a small script in python which anyone can run on their own systems, decide their own set of RSS feeds and get the relevant news articles from past hours. The goal explicitly is: given multiple sources of documents, RSS news feeds, cluster together similar documents that cover the same material. For instance, both *NYT* and *Washington Post* might cover Hillary Clinton's Primary win in New York. In the google news we can see that USA Today, Boston Herald and 10 other news organizations are publishing the same story. We want to recognize that they are the same underlying story and cluster or aggregate them.

II. RELATED WORK

There are lots of free and premium web based applications for feed aggregation as well as news aggregation, though they serve the larger purpose of providing streamlined news, our project focuses on using open source tools and libraries to create a news aggregation backend which is lightweight and use it to power a news aggregation engine that can be personalized on per user basis. Some example of news aggregator are – techmeme.com, feedly.com, flipboard.com etc [3][4][5].

III. METHODOLOGY

We are using following key steps to do our aggregation as defined in next steps. We chose technology feeds for our experiment and aggregated the most relevant technology news articles.

Defining a Set of RSS feeds: We start with a predefined set of RSS feeds, in our case we are starting with technology news category so we include following feeds in our set. The data we want to work on is very diversified and heterogeneous. So we remove this issue by predefining a set of most rated feeds. By doing this we can easily filter a lot of content because most of the news publications use the same sources to research their content. This is one of our key factors to tackle fast moving data and recognize only the most relevant articles during the given time period. We worked in technology news category and used following website's RSS feeds for our first clustering attempt: TechCrunch, SFGate, Computer Weekly, Cnet, ZDnet, TheNextWeb, TechGIG, CodeNinza etc.

Parse RSS feeds: Using universal feed parser [6] we can parse our feeds. Feedparser is an excellent feed parsing library for python, It provides functions to capture the semantics of web news articles such as authors information, publication date, title, body of text article, images and any other that publisher provides. We parsed our feeds and stored the titles along with text body of article in our corpus. Once the filtering is done by these sources, we can carry out rest of operations.

We want to parse our sources. We used the document title and description as a short summary. We used the natural language processing library NLTK [7]. For each document, We concatenate the title and description, convert to lowercase, and remove one character words. We stored these final documents in a corpus.

We also stored the titles to make the final results recognizable. The end result looked like:
0 TITLE Microsoft stops X-Box 360 manufacturing

1 TITLE Intel layoff 12,000 employees in 2016
2 TITLE New VR devices ready to change the gaming
3 TITLE Facebookenablesarticle view for publishers
4 TITLE New video shows non public facebook app

Extraction of Keywords using TF-IDF: TF-IDF composed by two terms: first computes the normalized Term Frequency, that is the number of times a word appears in a document upon total number of words in that document. The second term is Inverse Document Frequency, which computes the logarithm of the number of the documents in the corpus divided by the number of documents where the term appears. The TF-IDF shows how important a word is in a document in a collection, it takes in consideration the isolated term and the term within the document collection. The intuition is that a term that occurs frequently in many documents is not a good discriminator[8]. So it will scale down the frequent terms while scaling up the rare terms; for instance, a term that occurs 10 times more than another isn't 10 times more important. To compute the TF-IDF weights for each document in the corpus, we operate on the corpus in following order of steps:

1. Tokenize the corpus in words present in each document.
2. Model the Vector Space for corpus decide features.
3. Compute the TF-IDF for each document in the corpus again.

We only want to extract top keywords from each document and store the set of these keywords, through all documents, in key word list that we are maintaining.
0 KEYWORDS Bill 360 stops Xbox.
1 KEYWORDS layoff 12000 2016 intel...
99 KEYWORDS facebook red whatspp encryption.

1.1 Union these to a set of features
We have the superset of keywords, so we go through each document again and compute TF-IDF. Thus, all the entries are 0 this will likely be a sparse vector.With a vector of TF-IDF for each document in corpus, we can now got to our next step which is to create a feature vector for each document in corpus and compute the similarity.

Matrix of Cosine Similarity: With TF-IDF of each document, we can finally use cosine similarity [9] to compare the documents with each other. Cosine similarity measures the similarity between two vectors of an inner product space which measures the cosine ratio of the angle between them.

Clustering of Documents: Now we have a unique similarity number for every pair of documents. Our next step is to cluster. Our first choice was to use k-means++ but it get stuck at local optima and one has to choose k for it which is problem in news aggregation because in case of a special event almost all channels will be covering the same news hence we'll have fewer number of large clusters. On the other hand on a normal day, there may be a large number of smaller clusters. This poses the problem with a constant k. Instead if we use agglomerative or hierarchical clustering, we'll be growing clusters by fusing neighboring clusters to get a tree. Although the structure of tree will vary on daily basis but we can choose similarity threshold for pruning the tree to a final cluster set. Hence using hcluster python library, we obtain the following dendrogram. Finally to chose a threshold to prune -- how similar documents are similar enough. 0.75 worked good for our data. Then we just need to extract the clustered documents from the dendrogram and then print out the IDs with titles of the final clusters.

IV. FINAL AGGREGATED RESULTS

Here are the final results of clustered news in aggregated form. The results are impressive given the lightweight code and fast moving nature of news.

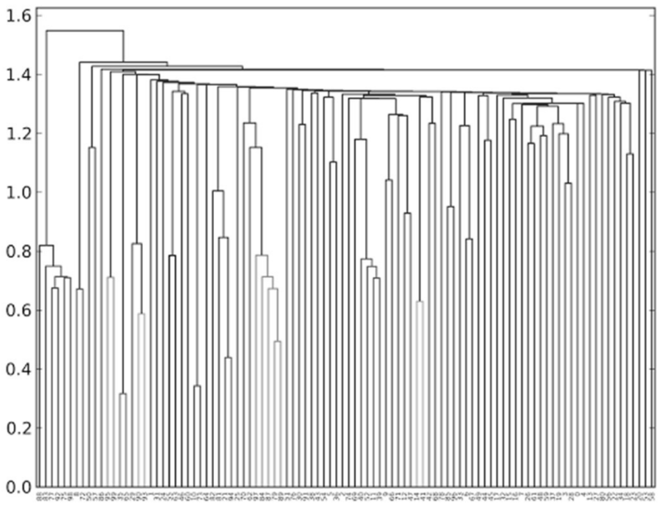

Figure 1. Dendrogram of final clusters.

```
==========================================
35 Facebook mobile app with slideshow and other cool things
65Unseen new features in FB application slideshow and more
==========================================
10 India whatsapp largest userbase, says German study 2016
73Indians are one third of whatsapp users: German Study
==========================================
21 Nest Dotcom To Host Mega's Launch Event in May 2016
94 Nest Dotcom next week mega launch event in Ukraine
==========================================
90 Data suggests Facebook's ads work better than most TVCs
93 Facebook's ads better than TVCs, reach more people fast
==========================================
```

V. FUTURE WORK

The aggregation can be improved using weighted TF-IDF, where words title are given preference over words in the body. The number of keywords can be varied to see if the results improve right now the best result is obtained at n_keyword

= 4. But the most important scope of this paper is to use this method to build a functional and scalable news aggregation web application.

VI. ACKNOWLEDGMENTS

Our thanks to experts who contributed towards through their research in field of semantic net and content aggregation, Sir Tim Berners Lee, Aaron Swartz for creating RSS, Kenneth Reitz for requests library, PSF for keeping python open source and alive.

VII. REFERENCES

[1]. Jeon, Doh-Shin and Esfahanizan, Nikrooz Nasr. 2012. *News Aggregators and Competition Among Newspapers in the Internet.* Northwestern University research facility.

[2]. Athey, Susan and Mobius, Markus.2012 *The Impact of News Aggregators on Internet News Consumption: The case of Localization.* Harvard University, Iowa University and Microsoft Research.

[3]. Strange, Adario. What makes techmeme tick? Inventor Gabe Rivera Explains. *Wired Magazine.* 2007.

[4]. Khodabakchian, Edwin. 2012. Feedly it is.

[5]. MacManus, Richard. 2010. How Flipboard was created & its plan beyond iPad. ReadWriteWeb.

[6]. McKee, Curt. 2010. Universal feedparser for python. PSF and PyPI.

[7]. Bird, Steven; Klein, Ewan; Baldrige, Jason and Loper, Edward. 2008. Multidisciplinary instruction with Natural Language Toolkit. *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics, ACL.*

[8]. Caraciolo, Marcel. 2011. Machine Learning with Python: Meeting TF-IDF for Text Mining. Artificial Intelligence in motion.

Satyam Gupta completed his under graduation in Computer Science and Engineering from IMS Engineering College. Currently, he is working with ‘Teach For India’ as a Fellow in Pune, teaching 120 kids Science. His areas of interest include Education Technology and aligned fields of Artificial Intelligence.

Crossbar Switch Using I-Slip Scheduling Algorithm for NoC

Abhimanyu Singh
Department of Electronics and Communication Engineering, IEC Group of Institutions, 4, Knowledge Park – I, Surajpur Kasna Road, Greater Noida 201306 UP India
abhimanyus88@hotmail.com

Abstract : **As fabrication technology continues to improve, smaller feature sizes allow increasingly more integration of system components onto a single die. Communication between these components can become the limiting factor for performance unless careful attention is given to designing high-performance switches. It provides fast communication and full N-to-N routing capabilities. This implementation is done in VERILOG, using Synopsys Design Vision software. Implemented switch is a modular design and consists of: buffered input port modules, output port modules and crossbar scheduler embodying i-SLIP scheduling algorithm. The module comprises of 8 input blocks that receive and queue requests from input devices, 8 output blocks that send packets to the output devices, and a crossbar scheduler that implements the iSLIP scheduling protocol.**

Keywords: Crossbar Switch, N-to-N Routing, Algorithm for i-SLIP, Synopsys Design Vision Software

I. INTRODUCTION

THE goal of this design is to provide a fast, efficient System on Chip (SoC) switch between 8 on-chip devices. The eight devices are connected to one another through a single instance of the routing switch to be designed. Each device has three output ports, and three input ports. i-SLIP scheduling algorithm used to design the scheduler features following advantage over Round-Robin scheduling algorithm :

- If an output receives any requests, it grants one that appears next in a fixed round robin schedule starting from the highest priority queue. However, the round robin at the output is not incremented (module *N*), unless the grant is accepted by the input in the Accept step. In other words, the priority round robin at the output side is incremented (provided that the grant was accepted) after the Accept step is passed.

- Those inputs and outputs not matched at the end of one iteration are eligible for matching in the next. This small change to the RRM algorithm makes iSLIP capable of handling heavy loads of traffic and eliminates starvation of any connections. The algorithm converges in an average of *O(log N)* and a maximum of *N* iterations. iSLIP can fit in a single chip and is readily implemented in hardware.

II. PROPOSED ALGORITHM

The algorithm for i-SLIP is as follows:
Step 1: *Request.* Each unmatched input sends a request to every output for which it has a queued cell.

Step 2: *Grant.* If an unmatched output receives any requests, it chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer *g_i* to the highest priority element of the round-robin schedule is incremented (modulo *N*) to one location beyond the granted input iff the grant is accepted in Step 3 of the first iteration.

Step 3: *Accept.* If an unmatched input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The pointer *a_i* to the highest priority element of the round-robin schedule is incremented (modulo *N*) to one location beyond the accepted output only if this input was matched in the first iteration.
Constraints:

- This implementation of the i-SLIP protocol will use a fixed number of input and output ports. It begins with an 8x8 design in synthesized static CMOS. Since the most complex portion of the design appears to be the programmable priority encoders, most of the design phase will be focused on optimizing their implementation for speed and area.

- The i-SLIP switch will be programmable to support a variable number of iterations from 1 thru *N*.

- Each input port will contain one virtual queue per output port. These virtual queues will share a buffer pool of up to 32-packets per input.

Changes to i-SLIP
Due to the flow control mechanisms, i-SLIP must be modified so that it doesn't connect an input port to an output that does not have enough credit remaining to accept a data transfer. To handle this case, outputs with 0 credit will not participate in the arbitration iteration. If credit arrives between iterations

of the same scheduling cycle, the port may become active again and participate in the remaining arbitration iterations.

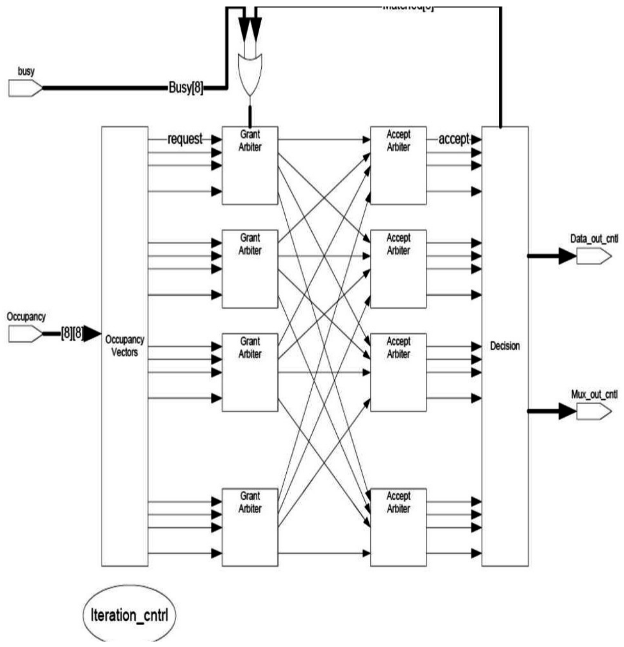

Figure 1. Scheduler Block Diagram.

III. PERFORMANCE

- Despite its simplicity, the performance of *i*SLIP is surprisingly good. A detailed study of its performance and implementation can be found. In brief, its main properties are:

- **Property 1.** *High Throughput* — For uniform, and uncorrelated arrivals, the algorithm enables 100% of the switch capacity to be used.

- **Property 2.** *Starvation Free* — No connection is starved. Because pointers are not updated after the first iteration, an output will continue to grant to the highest priority requesting input until it is successful. Furthermore, *i* SLIP is in some sense fair: with one iteration and under heavy load, all queues with a common output have identical throughput.

- **Property 3.** *Fast* — The algorithm is guaranteed to complete in at most *N* iterations. However, in practice the algorithm almost always completes in fewer iterations. *i.e.* for a switch with 16 ports, four iterations will suffice.

- **Property 4.** *Simple to implement* — An *i*SLIP scheduler consists of 2*N* programmable priority encoders. A scheduler for a 16-port switch is readily implemented on a single chip.

- The performance of *i*SLIP can be seen in Figure 2. It shows that throughput of RRM is increaced from 63% to 100% in *i*SLIP.

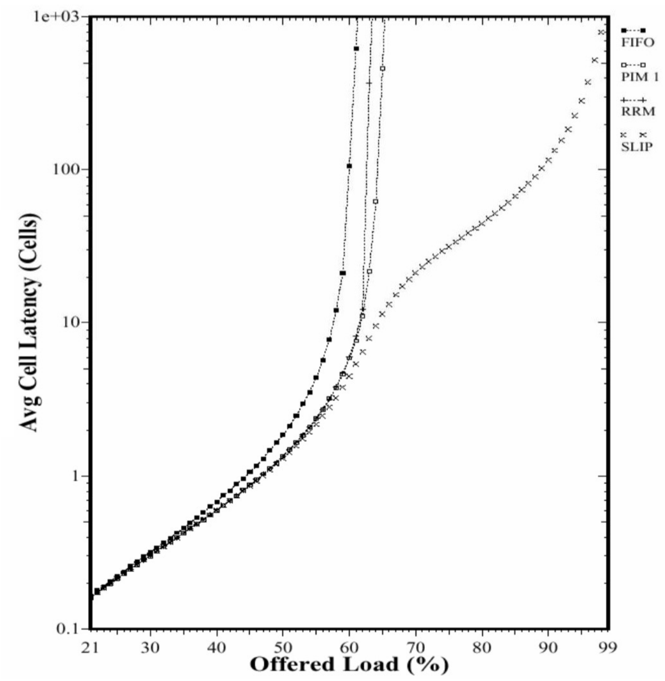

Figure 2. Performance Graph.

IV. RESULTS AND DISCUSSION

TESTING

- The switch design underwent a large testing effort to verify functional correctness. The testing involved both unit and system-level testing using structured and random stimulus and assertions. The testing occurred in two phases: unit-level verification and system-level verification.

RESULT

- The split unit-level and system-level testing strategy worked well for the 8x8 switch verification.

- By starting with the unit-level testing, low-level design bugs were flushed out without having to deal with the additional complexities of the other complex design components. Bugs in the unit-level verification were easy to diagnose and fix. Unit-level testing uncovered problems in the complex arbiter wiring of the scheduler, a bit-order problem in the priority encoders, and several bugs in the input block linked list update logic. In addition, unit-level testing uncovered a microarchitectural limitation in the input block when an incoming packet arrives in the same clock as a scheduler send request. The microarchitecture was changed to allow this case.

- Once the unit-level bugs were worked out, the system-level verification tested the interaction between the three main design units as well as the wiring of the 8x8 switch. Several problems were found in the wiring of the input_blocks to the scheduler and output blocks, and yet more linked-list update