Journal of Engineering and Applied Sciences 13 (15): 6002-6007, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

Routing Architecture for Efficient Network on Chip Using Agent

[1]Y.L. Ajay Kumar, [2]D. Satyanarayana and [1]D. Vishnu Vardhan
[1]Department of ECE, Jawaharlal Nehru Technological University (JNTUA), Ananthapuramu, India
[2]Department of ECE, Rajeev Gandhi Memorial College of Engineering and Technology (RGMCET), Nandyala, India

Abstract: Network-on-chip is one of the promising computing paradigms. Continuing research is carried out to improve the architecture and performance of networking techniques on the chip. In this research, we will demonstrate that routing efficiency can be improved in a mesh network using inbuilt "Agents" inside the processing elements. Moreover, this architecture is convenient to be used in either homogeneous or heterogeneous mesh networks. The packets are injected into various processing elements thorough multiple ports. The packet frame is designed and managed in such a way that keeping track of processed and un-processed objects is easy to track and helps maintain data integrity. We see that implementation results on FPGA show better area performance, minimized routing complexity and better throughput in a processing element.

Key words: NOC, packet injection, data integrity, processing element, routing agents, routing efficiency

INTRODUCTION

In nowaday's electronics VLSI plays an important role to reach the evolving computation-intensive researches and this makes the necessity high-performance, low power systems, the number of computing applications in single-chip has drastically increased but the present VLSI technology can favors such an excessive integration of transistors. By integrating more number of processors on a single chip such as CPU, DSP and specific IP's. The inter connection between an integration of processors become more challenging task. This task is managed by a applying the computer networking concepts inside a chip which is quite often termed as the Network-on-Chip abbreviated as "NOC". According to (http://www.arteris.com/noc_whitepaper.pdf, 2005), the NOC approach has a clear advantage over traditional busses and most notably system throughput.

The network techniques in a computer network are developed well earlier but it is not possible to implement on chip-level intercommunication environment without any changes. For NOC architecture, The basic structure should be light weight and simple because NOC architecture should be small enough for implementation of basic components of SOC and further it has to meet basic requirement of NOC, i.e., it must be low-power consumptive. If any device has to be low powered many parameters has to be considered, like operating voltage, clock rate and power management scheme. Thus, designers to project future chips have to find the appropriate design and process technologies as well as the ability to interconnect the existing components-including processors, controllers and memory arrays-providing the functionally correct and reliable operations conducive to the proper interaction of the components. Therefore, this topic requires in-depth investigation and research for the benefit of the industry.

In this study, we present a novel technique of routing packets to the neighboring processing elements using routing logic which we term as "Agents" which is in-built in to the processing elements. We evaluate this idea by developing a verilog model of the complete SOC with an cryptography application as seen by Holden *et al.* (2010). This design is implemented in full-scale and realized on Xilinx FPGAs. We also highlight the advantages of the system and discuss about how the framework can be scaled.

Literature review: Sharma *et al.* (2014) a brief description of how the NOC evolved from bus and crossbar systems can be found. Using NOC technology, the SOC interconnect is constructed by interconnecting any number of NOC interconnect IP elements such as switches, synchronizers, width converters, power isolator cells and others through a set of links into a user-defined topology. Several NOC design challenges are presented by Duarte (2003). We hope to overcome several of them in the current research.

Corresponding Author: Y.L. Ajay Kumar, Department of ECE, Jawaharlal Nehru Technological University (JNTUA), Ananthapuramu, India

A two step minimum path mapping algorithm based on genetic algorithm is proposed by Luo-Feng *et al.* (2008). In this case, first the IP cores are mapped to the NOC and then the shortest path is evaluated. This technique may induce more latency. Research by Kavyashree *et al.* (2016) tries to address the congestion problem in NOC routing architecture and talks about an architecture without a separate router module. However, the study of the network architecture and the processing element is not published.

Gu *et al.* (2016) proposes a new congestion sensing mechanism and control methods. But it is not a physical implementation on the FPGA, thus, effectiveness of the proposed material for physical implementation is not clear. Research regarding emerging interconnect solutions are presented by Carloni *et al.* (2009). However, they seem to move towards more complex solutions. Fault tolerant routing mechanisms are discussed by Liu *et al.* (2016) and Chen *et al.* (2017). In our opinion, though excellent algorithms are published, it may create too many connections which may not be suitable for FPGA implementation. An analysis of interconnection effects on the FPGA implementation of the NOCs is given by Zakaria *et al.* (2016). It says using long wire utilization in FPGA chip causes negative impact to chip's performance in technology scaling.

Various published works (Luo-Feng *et al.*, 2008; Kavyashree *et al.*, 2016; Schelle and Grunwald, 2008; Swain *et al.*, 2016; Cai *et al.*, 2015; Kumar *et al.*, 2016; Prasad *et al.*, 2016; Kumar and Gordon-Ross, 2016; Kurokawa and Fukushi, 2016) have implemented the NOC on an FPGA and have published the results. In our research, we propose an improvised architecture whose performance is individually studies across the FPGA families and compared with the above mentioned research as well.

MATERIALS AND METHODS

Design: Our NOC architecture contains processing elements interconnected in a 2D mesh network in the 3×3 format. All processing elements are interconnected with their neighboring processing elements. A global route interconnects all the peripheral processing elements. Global route is the significant design feature of this project which is facilitated by a simple routing management switch at the periphery. Apart from the interconnections, every PE has an input bus and an output bus through which the host can inject original packets and also, read the processed packets. The arrangement is as depicted in Fig. 1. All processing elements connect to the Global Routing Switch abbreviated as GRS.

Information is divided into smaller units called packets before sending them to the PE network. The input

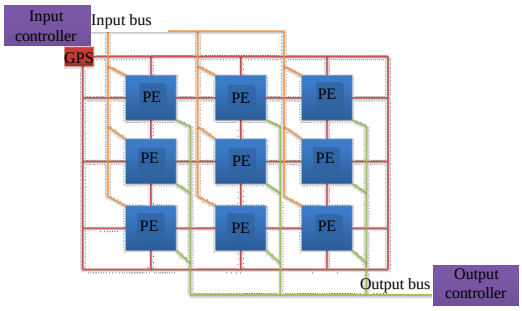

Fig. 1: Arrangement of PEs in 3×3 mesh optimized for FPGA implementation

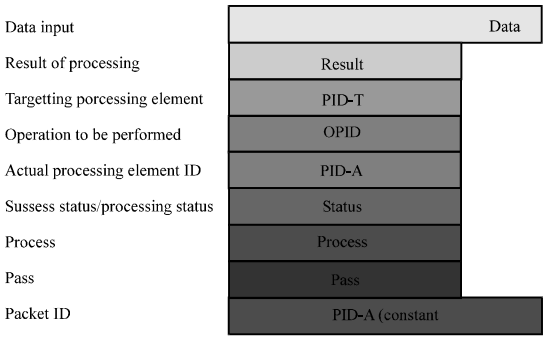

Fig. 2: Packet structure used in the NOC

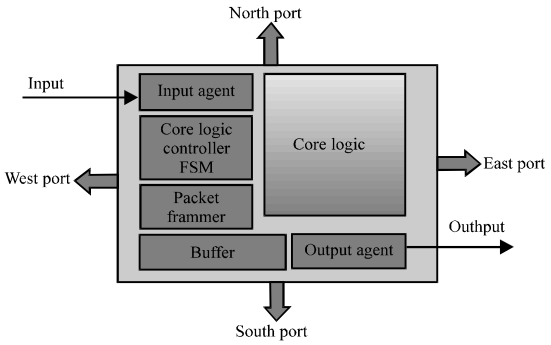

Fig. 3: Processing element's internal architecture

controller is assumed to distribute the packets to all the PEs evenly and maintain the balance. The structure of the packet is shown in Fig. 2. In this architecture, we propose the use of 64 bit packets consisting of data and several useful data to maintain data integrity and management.

The processing element has several elements as shown in Fig. 3. They are explained in the following paragraphs. The solid blue colored blocks are fixed elements and the core logic can change from PE to PE depending upon the application. This places the routing framework in a vantage point that it can be re-used and re-configured for multiple applications. It should also

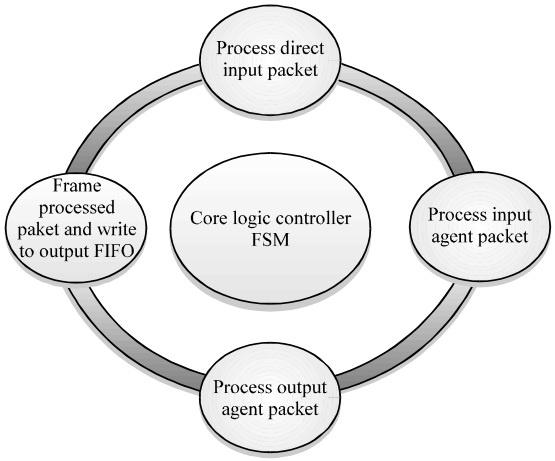

Fig. 4: Activities of the core logic controller FSM

be noted that the data can be injected to the every processing element separately and this ability is leveraged to design the multi-port packet injection mechanism to quickly transport the packets to the PEs.

The input agent is responsible for managing all the incoming packets and providing with the core logic. This is a memory element that will handle the incoming packets. FIFO is required because from the point of view of processing elements, incoming packets will be asynchronous. Thus, to manage the asynchronicity of packets, we will need an input FIFO. The control state machine is responsible for controlling the processing related activities such as depicted in Fig. 4.

The incoming packets are structured in a certain specific way to maintain their identity. For the data to be processed, we add many headers. The following picture shows how the packets look like. After the packets are received and processed, the headers should be updated. This is what aframer does. Output Agent is responsible for managing all the outgoing packets. This employs the round robin algorithm and priority based routing as a secondary algorithm to send outgoing packets to either the neighboring elements or the external packet manager to the host. Core logic is the functional block. In this project, we will implement a cryptography core for evaluation purposes. It has encryption and decryption blocks (Holden *et al*., 2010) which give us an intrinsic advantage for evaluating our design.

North, East, West, South ports through which processing element will talk to the neighboring elements and also connected to the global interface. Packets arrive into the processing element through the input port and the packet will read outside through the output port. The protocol used here is a round-robin method, however, any other protocol can be chosen to be implemented depending upon specifications.

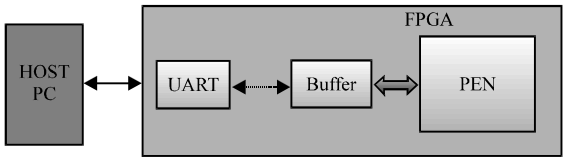

Fig. 5: FPGA and PC for real-time sending and receiving data

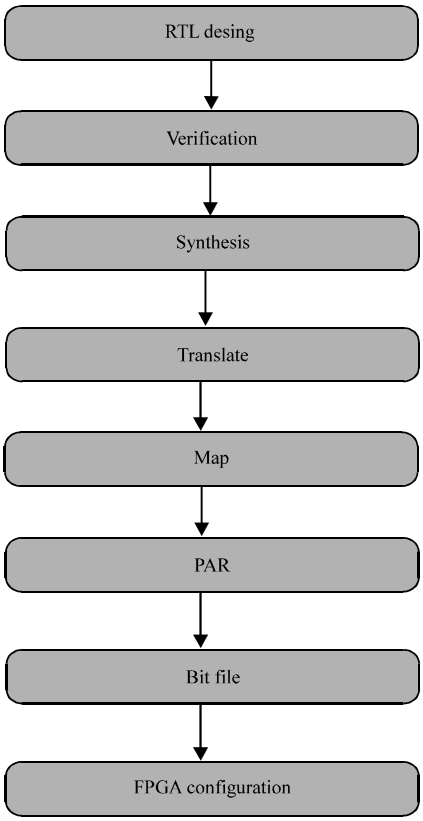

Fig. 6: Xilinx FPGA design flow

This research is implemented by the Xilinx ISE 14.2 tool. As this design is created to be tested on an FPGA, it is important that some communication feature should be used to transfer some live data from a host computer to the FPGA board. The setup is shown in Fig. 5 and 6.

RESULTS AND DISCUSSION

Performance analysis: Implementation of the NOC design will follow the steps as shown in the flow diagram in Fig. 6. This is in accordance to the FPGA design flow specified by Xilinx. All analysis is done using the Xilinx tools.

Table 1: Resource analysis of PEs

FPGA family	Area (slices)	FFS	LUTs	Speed (MHz)	Memory (%)	IOs	Area (%)	Max. combinational delay
Spartan 3	365	356	625	122	9	679	1.10	4.802
Spartan 3A	360	362	616	134	10	679	3.16	7.430
Spartan 6	306	298	298	191	3	679	0.24	5.234
Virtex 4	368	361	627	226	2	679	0.40	4.421
Virtex 5	360	437	437	276	1	679	1.00	3.613
Virtex 6	305	343	343	350	0.50	679	0.04	2.662
Virtex 7	305	343	343	375	0.40	679	0.02	1.523

Table 2: Resource analysis of 3×3 networks using the agent based PEs

variables	Area (slices)	FFS	LUTs	Speed (MHz)	Memory blocks (%)	IOs	Area (%)	Max.combinational delay (nsec)	Throughput theoretical calculation in GBPS
Spartan 3	2049	2684	2846	105.662	92(<100)	159	0.06	9.464	0.650
Spartan 3 A	2066	2676	2832	121.198	90(<100)	159	0.18	8.251	0.740
Spartan 6	2398	2781	2398	200.698	90(33)	159	0.03	4.983	1.228
Virtex 4	2176	2667	3177	211.372	90(26)	159	0.02	4.731	1.290
Virtex 5	2487	2710	2487	286.640	45(13)	159	0.01	3.489	1.754
Virtex 6	2443	2746	2443	350.152	45(6)	159	0.30	2.856	2.142
Virtex 7	2443	2746	2443	375.672	45(3)	159	0.10	2.662	2.299

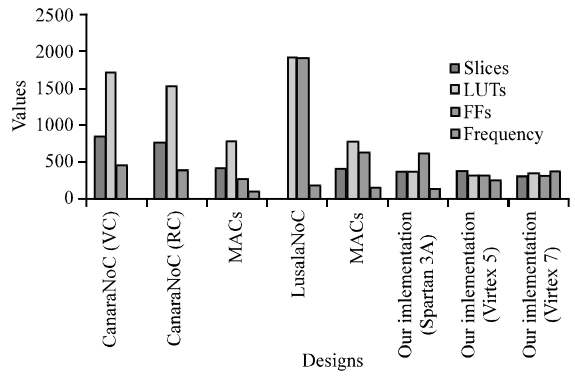

Fig. 7: Comparison of various PE designs without implementation of PE on a few FPGA

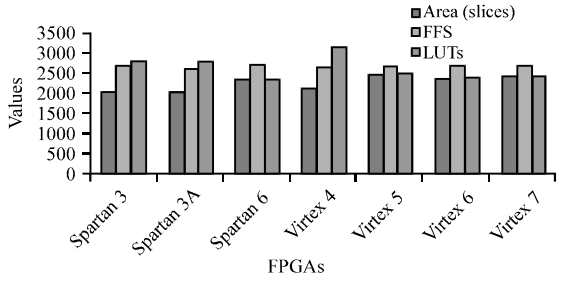

Fig. 8: The graphical representation of resource utilization in various FPGAs for implementation of 3×3 network

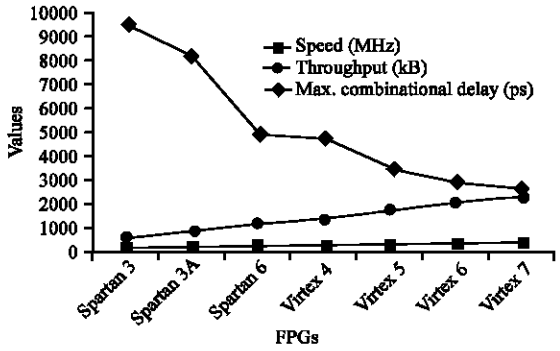

Fig. 9: Variation of combinational delay, throughput and frequencies across FPGAs

First, we will present the performance metrics of the PE and network separately. The PE that is designed can be used as a standalone IP as well. Because of this reason, we made a study on the area and speed performance of the PE using 4 input LUT devices and 6 input LUT devices.

Next, we will present how the PE and the network perform compared to other similar research published. Table 1 and 2. The performance of the PE and 3×3 network of PEs on Various FPGA families from Xilinx.

The processing element that we designed was compared with a research given by Kumar *et al.* (2016). The results are plotted in Fig. 7 and 8.

In the 3×3 network as the combinational delay is decreasing across the FPGAs, operating frequency of the network is increasing and hence, the throughput. This is clearly depicted in Fig. 9.

Now we present some of the comparative studies with respect to similar research that has been done. In research by Kavyashree *et al.* (2016), a similar study is done on a 3×3 network and in Fig. 10, we can see that our implementation performs better in most of the cases with respect to operation frequency.

In research by Schelle and Grunwald (2008) an simple NOC and a virtual channel NOC is implemented and the resource utilization values are given. In Table 3, we tabulate the result of our agent based design in the speed

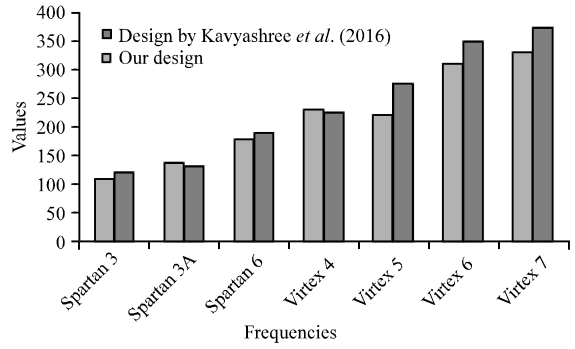

Fig. 10: Comparisons of operational frequencies

Table 3: Area and speed optimized settings for 3x3 agent based NOC

Variables	VC	Simple No.	Agent based (S)	Agent based (A)
FF	6621	1246	2170	2449
LUT	9237	1961	2487	2387
Max.freq	NA	NA	286.64	216.706

and area optimized settings in the tool. The design is implemented on a similar platform as given by Schelle and Grunwald (2008).

It must be noted that, we used a 64 bit packet where as a lot of compared research uses only 32 bit packets. The operation frequency is not published by Prasad *et al.* (2016), however, we have documented that out design in a similar configuration researchar at 735.132 MHz. From simulation results given above the communication between processing elements will take about 6 cycles in our implementation and by Prasad *et al.* (2016) it is published that it takes about 13 cycles. We conclude that this decrease in number of cycles in our implementation is because of absence of the external router logic which can induce more latency.

CONCLUSION

In this research, we have presented a technique to create an "Agent" based framework for NOC implementation. This is an attempt to get rid of external routers and allow the data communication to happen from an external layer of the processing elements. The systems route the packets with awareness of the state of the neighboring elements. It can also be noted that this framework is suitable for heterogeneous implementation of NOCs as well. The data integrity issues are handled by framing data along with appropriate headers. FIFOs are used to handle data rate mismatch issues. We have designed the PEs to be self contained, so that, they can be individually controlled. This helps in scalability and when not required they can be turned off, so that, power can be saved. This will also allow scale-up and scale-down of the system with ease. This is because the strategy of NOC should be made available for all families of FPGAs small or big. Multi-port top level packet injection allows quick transfer of packets into the networked processing elements.

RECOMMENDATIONS

Future research should involve testing the network for different loads and studying the performance of the design for various applications. Fault tolerance capability can also be included in the design. This should give way to an improvised scalable networking mechanism on chip. Apart from these, we also foresee through intuitive analysis that several other issues should be addressed such as scale-up modality, a deeper study on the impact of the state of neighboring elements on routing, timing closure for FPGA and ASIC implementation, deadlock recovery, load balancing, energy factor, etc.

REFERENCES

Cai, Y., K. Mai and O. Mutlu, 2015. Comparative evaluation of FPGA and ASIC implementations of bufferless and buffered routing algorithms for on-chip networks. Proceedings of the 2015 16th International Symposium on Quality Electronic Design (ISQED), March 2-4, 2015, IEEE, Santa Clara, CA, USA., pp: 475-484.

Carloni, L.P., P. Pande and Y. Xie, 2009. Networks-on-chip in emerging interconnect paradigms: Advantages and challenges. Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, May 10-13, 2009, IEEE Computer Society, Washington, DC, USA., ISBN:978-1-4244-4142-6, pp: 93-102.

Chen, Y.Y., E.J. Chang, H.K. Hsin, K.C.J. Chen and A.Y.A. Wu, 2017. Path-diversity-aware fault-tolerant routing algorithm for network-on-chip systems. IEEE. Trans. Parallel Distrib. Syst., 28: 838-849.

Duarte, M.E., 2003. Networks on Chip (NOC): Design Challenges. In: International Conference on Computer Architecture, Omondi, A. and S. Sedukhin (Eds.). ICCA, Amsterdam, The Netherlands, pp: 121-128.

Gu, X., H. Cai, K. Cui, N. Huang and Y. Yang, 2016. Research on congestion perception and control of network on chip. Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), October 15-17, 2016, IEEE, Datong, China, ISBN: 978-1-5090-3711-7, pp: 866-870.

Holden, J., R. Hulman, L. Holden, M. Musa and E. Schaefer *et al*., 2010. A simplified AES algorithm. J. Eng. Algorithm, 1: 1-5.

Kavyashree, G.S., S.S. Bhusare and S. Shirahatti, 2016. Architectural based congestion management for network on chip implemented on FPGA. Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), July 21-23, 2016, IEEE, Bangalore, India, pp: 258-263.

Kumar, M., K. Kumar, S.K. Gupta and Y. Kumar, 2016. FPGA based design of area efficient router architecture for Network on Chip (NoC). Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA), April 29-30, 2016, IEEE, Noida, India, ISBN: 978-1-5090-1667-9, pp: 1600-1605.

Kumar, R. and A. Gordon-Ross, 2016. MACS: A highly customizable low-latency communication architecture. IEEE. Trans. Parallel Distrib. Syst., 27: 237-249.

Kurokawa, Y. and M. Fukushi, 2016. A prototype router circuit for extended 2D mesh network on chips. Proceedings of the 2016 IEEE 5th Global Conference on Consumer Electronics, October 11-14, 2016, IEEE, Kyoto, Japan, ISBN:978-1-5090-2334-9, pp: 1-2.

Liu, J., J. Harkin, Y. Li and L.P. Maguire, 2016. Fault-tolerant networks-on-chip routing with coarse and fine-grained look-ahead. IEEE. Trans. Comput. Aided Des. Integr. Circuits Syst., 35: 260-273.

Luo-Feng, G., D. Gao-ming, Z. Duo-Li, G. Ming-Lun and H. Ning *et al*., 2008. Design and performance evaluation of a 2D-mesh network on chip prototype using FPGA. Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2008), November 30-December 3, 2008, IEEE, Macao, China, ISBN:978-1-4244-2341-5, pp: 1264-1267.

Prasad, E.L., A.R. Reddy and M.G. Prasad, 2016. EFASBRAN: Error free adaptive shared buffer router architecture for network on chip. Procedia Comput. Sci., 89: 261-270.

Schelle, G. and D. Grunwald, 2008. Exploring FPGA network on chip implementations across various application and network loads. Proceedings of the International Conference on Field Programmable Logic and Applications (FPL 2008), September 8-10, 2008, IEEE, Heidelberg, Germany, ISBN: 978-1-4244-1960-9, pp: 41-46.

Sharma, S., C. Mukherjee and A. Gambhir, 2014. A comparison of network-on-chip and busses. Proceedings of National Conference on Recent Advances in Electronics and Communication Engineering (RACE-2014), March 28-29, 2014, The Education Group, New Zealand, Oceania, pp: 1-7.

Swain, A.K., K.R. Babu, S.N. Satpathy and K.K. Mahapatra, 2016. FPGA prototyping and parameterised based resource evaluation of network on chip architecture. Proceedings of the IEEE Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), August 13-14, 2016, IEEE, Mangalore, India, ISBN: 978-1-5090-1624-2, pp: 227-231.

Zakaria, F.F., N. Latif, S.J. Hashim, P. Ehkan and F.Z. Rokhani, 2016. Cooperative virtual channel router for adaptive hardwired FPGA network-on-chip. Proceedings of the 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), October 25-28, 2016, IEEE, Jeju, South Korea, ISBN: 978-1-5090-1571-9, pp: 358-361.