Crossbar switch module implementation for laboratory purpose

A Sani, Suherman*, Rahmad Fauzi
Electrical Engineering Department, Universitas Sumatera Utara, Indonesia

*E-mail: suherman@usu.ac.id

Abstract. Current communication and internet networks are underpinned by the switching technologies that interconnect one network to the others in forms of switches and routers. Students’ understanding on networks rely on how they convey the theories. However, understanding theories without touching the reality may exert spots in the overall knowledge. This paper reports the switch module design and implementation for student laboratory activities. The hardware and software designs are based on three stages clos switching architecture with modular 3x3 switches, controlled by an arduino microcontrollers. The designed modules can also be extended for batcher and banyan switch, and working on circuit and packet switching systems. The circuit has been successfully implemented by using 9 switch modules of 3x3, each module contains 9 relays and one arduino. The test shows 94% successful connection while 6% blocking caused by busy destination, not busy switches.

1. Introduction
Switching technologies route connection either circuit or packet switching from sources to destinations [1]. Communication network technologies start from switching design. It was started by Charles Clos [2] who published multistage interconnection network in 1953. The clos switch forms an N x N network by using some stages of smaller n x n crossbar switches to produce a non-blocking. His work was then improved by Benes using re-configurable non-blocking switches. Some researchers proposed algorithms for this re-routing process [3] [4] [5] [6]. The drawback is, Benes network needs a complex control to accommodate new calls [7].

Banyan is then reducing Benes complexity by using a self-routing switches [8]. Batcher removes Banyan drawback which is blocked-able shuffling Banyan input [9]. Later, Batcher Banyan dominates switching improvements for telephone switches and routers. However, the laboratory devices for student learning are limited. Although laboratory equipment that shows how technology is developed exists such as in [10], most switches or routers modules are dominated by software, emulators, or commercial products that are hardly representing the way those technologies working.

This paper is aimed to develop a simple laboratory-based switch development with many arduinos, so that students easily understand how switching be developed. By using this simple switch example, students are expected to be able to develop a more complex, a more efficient and a single controller based multi-stage switching system.
2. Research Methods

2.1 Basic Requirement
As the smallest component within switching system is a full squared crossbar switch, this paper reports a 9 x 9 switch module design. The designed crossbar switch should be able to connect all inputs to all outputs. The schema is shown in Figure 1. For n input and n output, the number of cross-points is $n^2$.

![Figure 1. 9 x 9 Crossbar Architecture](image)

This module should be able to be interconnected one to another to form a multi-stage switching system as shown in Figure 2, so that for large input output ports, the number of cross-points reduced without experiencing blocking. The probability of blocking should fulfill the Equation 1 requirement:

$$P_B = \frac{(u)!^2(2-u)!^m}{m!(2u-m)!}$$

where $u=n-1$, $n$ is number of input in the first stage module, $m$ is number of switching modules in second stage, $p$ is the probability of an input link busy.

![Figure 2. 9 x9 of 3-stage Switching System](image)

2.2 Hardware Software Requirement
As the system should be designed as simple as possible to make students easily understand how switch or router works, relay modules and arduino controllers are used. System should be able to connect source to destination ports and be confirmed either connection successful or not. The selected cross-point is based on setup information (for telephone) or header information (for router).
3. Implementation Results

3.1 Single Module

Figure 3 shows the hardware design of the 3 x 3 crossbar switch. This switch is formed by using an arduino as the controller and six relays as the cross-points. Nine arduino output ports are connected to the 9 relay controllers. The three horizontal connections of relays are for 3 inputs, while other 3 vertical connections are for output ports.

![Module circuits](image1)

![Module implementation](image2)

Figure 3. Single 3x3 Crossbar Switch

3.2 Powering the Multiple Modules

The power supplies of relays make use of powering ports within arduino. Since this hardware should be able to be interconnected one another, the power supply should have enough current to drive all the relays. Two powering configuration of later stages are proposed. The first one uses common power supply in parallel, the other one arranges module in series as shown in Figure 4. Each arduino represents a single 3x3 crossbar switch.

![Parallel powering](image3)

![Serial powering](image4)

Figure 4. Powering plan
3.3 Complete Module
When the crossbar modules in Figure 3 are arranged forming a multi-stages switch, as in Figure 2, an arduino is allocated as the common controller that controls all arduinos in every crossbar switches. To do so, serial communications are established between every arduino in every crossbar switches to the common control arduino. Number of input output ports in common control should be sufficient. Figure 5 shows the complete module.

Figure 5. The hardware implementation

3.4 Software design
In order to connect inputs and outputs, arduino in each crossbar module initiate each relay state or cross-point as zero to represent that cross-point is free. As the designed crossbar is 3 x 3, there will be 9 cross-points (n=9). If there is a request, for instance, connecting or disconnecting input 1 to output 3, the relay of the related cross-point should be closed or opened by increasing the state by 1. Before that, arduino should check the cross-point state by operating the modulo-2 to the state value. If the result is zero, relay is free, otherwise is connected. Arduino may connect or send busy tone. Common control arduino should decide alternative routes for busy reply. Figure 6 shows arduino software flow in each module.

Figure 6. Software flow
3.5 Test Results
By plotting erlang to 50% that means probability of a destination busy is 0.5. Figure 7 depicts test results. The results are obtained by giving an input such as, 1*1 means 1 connects to 1 or 1 disconnects from 1. The plot shows 100 iterations producing 6% failures or blocked. However, it does not mean 6% blocking rates as the causes are that destinations numbers have been connected to other ports. The cross-points on the other hand, are non-blocking or 0% blocking rate. Theoretically, according to Equation 1, for n=3, and m=3, the blocking probability is closed to zero [12].

![Figure 7. Test results](image)

4. Conclusions
This paper has presented the crossbar switch module designed and implementation by using 9 of 3x3 crossbar switches, each switch contains 9 relays and one arduino. The test results satisfied the requirement and the switch blocking is theoretically almost zero. The blocking is caused by busy destination as the experiment conducted. Future work may consider switch close and release time that may increase blocking probability.

Acknowledgement
This research has been supported by the Penelitian Produk Terapan, funded by DIKTI through the grant No 154/UN5.2.3.1/PPM/KP-DRPM/2018.

References
[1] Sani A, “Multistage switching hardware and software implementations for student experiment purpose,” IOP Conf. Ser. Mater. Sci. Eng., vol. 309, no. 1, p. 12097, 2018.
[2] C. Clos, “A Study of Non-Blocking Switching Networks,” Bell Syst. Tech. Journal., 1953.
[3] Opferman D.C. and Tsao-Wu, “On a Class of Rearrangeable Switching Networks,” Bell Syst. Tech. Journal, vol. 50, no. 5, 1981.
[4] L. E. and Z. S.Q., “Parallel routing algorithms for non blocking electronic and photonic switching networks,” IEEE Trans. Parallel Distrib. Syst., vol. 16, no. 8.
[5] S. Chakrabarty, A., Collier, M., Mukhopadhyay, “Matrix-Based Nonblocking Routing Algorithm for Benes Networks,” Comput. World Futur. Comput. Serv. Comput. Cogn. Adapt. Content, Patterns.
[6] A. Karimi, “Introduction And Analysis Of Optimal Routing Algorithm In Benes Networks,” Int. Conf. Robot PRIDE 2013-2014 - Med. Rehabil. Robot. Instrum.
[7] L. G. V. G. F. Lev, N. Pippenger, “A Fast Parallel Algorithm for Routing in Permutation Networks,” *IEEE Trans. Comput.*, vol. 30, no. 2, pp. 93–100, 2015.

[8] G. J. Goke L.R. dan Lipovski, “Banyan Networks for Partitioning Multiprocessing Systems,” in *Proceeding 1st Annual Computer Architecture Conference*, 1973, pp. 21–28.

[9] J. H. Patel, “Performance of Processor-Memory Interconnections for Multiprocessors,” *IEEE Trans. Comput.*, vol. 30, no. 10, 1981.

[10] L. O. Onibonoje, M. O., Umeh, U. N., & Kehinde, “Development of an Arduino-based trainer for building a wireless sensor network in an undergraduate teaching laboratory,” *Int. J. Electr. Electron. Sci.* 2(3), 64-73., 2015.

[11] C. Y. Lee, “Analysis of Switching Network,” *Bell Syst. Tech. Journal.*, 1955.

[12] M. Zulfin, R. Fauzi, M. Razali, and M. Pinem, “Cross-Point Comparison of Multistage Non-Blocking Technologies,” vol. 7, pp. 703–708, 2018.