

Universal Stage Switching Algorithm for Hypercube Interconnection Network in Multi-Core System

Rakin Muhammad Shadab, MdAhidulIshlam, MdRakibul Islam, Foysal Mahmud Farabi, Jamal Uddin Ahmed

Department of Electrical & electronic Engineering Ahsanullah University of Science & technology 141-142, Love Road, Tejgaon Industrial Area, Dhaka, Bangladesh

Abstract: Over the past few decades, general-purpose processor designs have evolved by leaps and bounds with the ability to automatically find and exploit increasing amount of parallelism in sequential programs [1]. In this continuation, the first was narrow pipelined single-issue processors. One huge bottleneck that a particular single core processor faces is the inability of its ALUs to perform more than only a subset of its full instruction set at higher frequencies [2]. Its complex instruction set can only be fully executed during normal (lower) clock speeds. All these limiting factors lead to the advent of multi-core system. But even in multi-core system, efficient networking and fast switching is the key to performance gain. Many unique and complex interconnection networks are used for performance improvements in multi-processor and multi-core systems [3]. This paper emphasizes on developing a universal algorithm for employing linear or through stateswitching in a general hypercube network. Such an algorithm will be very helpful to design higher order interconnection blocks.

Keywords: Bit sequence, Hypercube network, Linear switching, Multi-core, Universal algorithm

I. Introduction

Microchip manufacturers today focus on increasing parallel compute performance by adding more and more processor cores on a single die. This is very useful to multiply the overall performance by operating all the cores at a stable and optimized frequency compared to a single core processor operating at a higher frequency. Several network structures are used to create interconnections between cores to cores and cores to memory pool. Widely used interconnect topologies include single bus as well as multiple bus network, tree network, crossbar network, simple and hybrid hypercube networks [3], [4]. Static interconnection networks can employ different paths including star, mesh, ring, serial or through state and hypercube links. We shall limit our discussion to general static indirect hypercube blocks only. The hypercube network that we are going to analyze has $N \log_2 N = 64$ connections with maximum node degree and maximum internode distance of $\log_2 N = 16$, where $N = 16$ is the number of input lines. Ultimately, it should lead us to the development of a switching algorithm that can be used uniformly [4], [5].

II. General 4-stage indirect hypercube switching network

Hypercube networks strike a good balance between wire delay and internode distance. Computer system architects regard this architecture as one of the most versatile switching techniques available. It is an efficient and prudent structure that is well recognized by microchip designers. In a simple 4D hypercube network each core has different ways to establish connection with the remaining cores. This provides greater flexibility as far as cache sharing and task/data sharing between cores are concerned. For an $N \times N$ indirect hypercube topology with n stages (where $N = 16$ and $n = 4$), each stage can employ either serial or cross state interconnections [6]. Our goal is to find out a switching algorithm in a 4 stage indirect hypercube structure where switching bits are arranged for serial or linear state switching. Such network is based on a concept of 'Universal Switching Blocks' [7].

III. Developing a universal algorithm

To establish a relationship that would indicate the arrangement of switching keys in higher order networks, we can start with a four stage network structure. This comprises of cascaded 16×16 hypercube blocks (each stage consisting of one multiplexer and one de-multiplexer). The final throughputs are controlled by selection switches used by multiplexers and de-multiplexers used in every stage. So these switching bits are arranged in specific sequences to produce the desired outputs. The output of the first stage of a linearly switched network, in this case, will follow a definite shape where the first and last output for every small group of inputs (consisting of four inputs) experience through state travelling and the second and third inputs interchange their

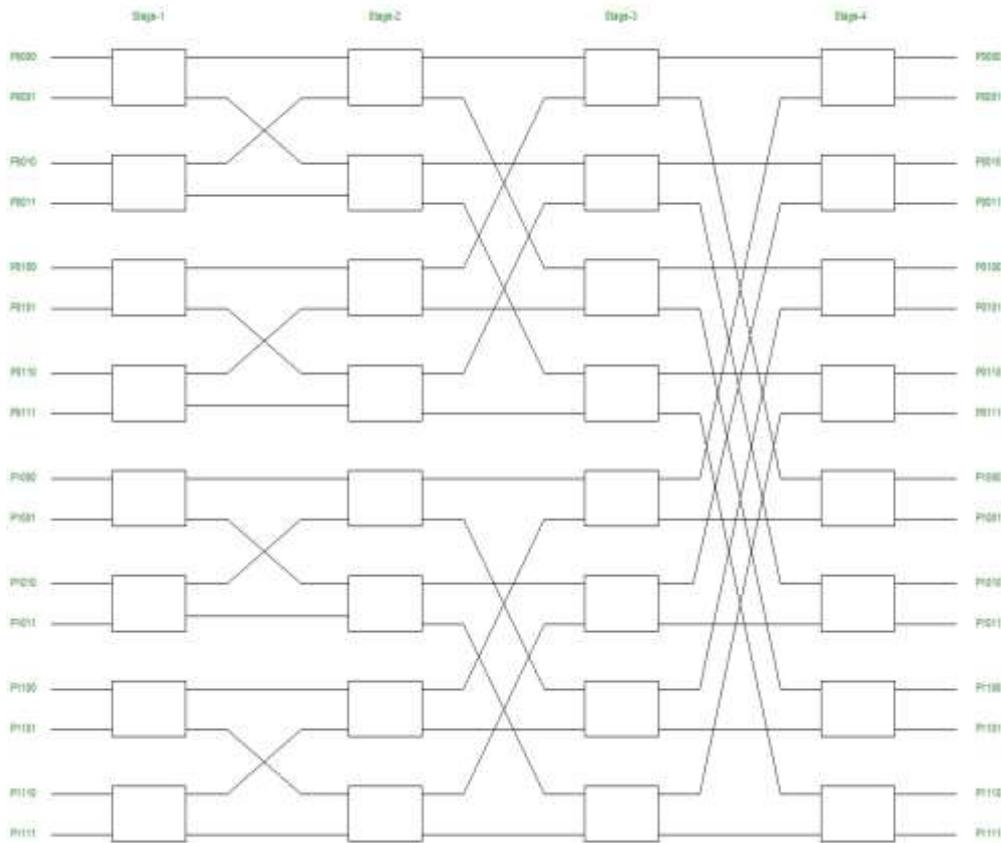


Fig.1 : 16 X 16 indirect hypercube network with linear switching

respective positions at the output. Each multiplexer in a new stage must follow the sequence of selection bits used by the de-multiplexer in immediate previous stage to provide continuity from one stage to the next stage. As with the scope of this paper, for serial or linear switching, the positions of all the switches maintain a certain sequence. A stable algorithm can be derived to represent this sequence for our particular network. If S_3, S_2, S_1, S_0 represent the selection bits from most significant bits to least significant bits, then the corresponding algorithm for stage to stage switching can be written as shown in the following Table:

Switching arrangement:

Stage of switching	Bit sequence	Remarks
Initial	$S_3S_2S_1S_0$	None
1 st	$S_3S_2S_0S_1$	S_1 and S_0 bits are interchanged
2 nd	$S_1S_2S_0S_3$	S_1 and S_3 bits are interchanged
3 rd	$S_2S_1S_0S_3$	S_1 and S_2 bits are interchanged
4 th	$S_3S_2S_1S_0$	A right shift has occurred

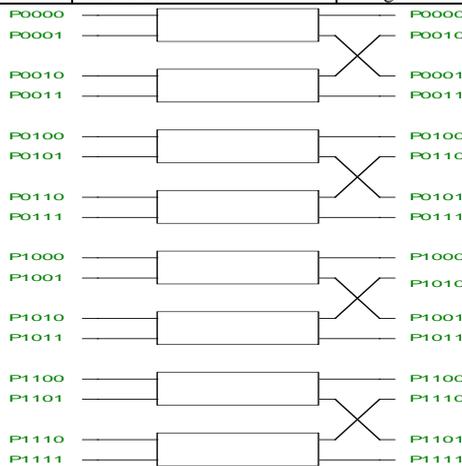


Fig.2: First stage of linearly switched 16 X 16 indirect hypercube network

One important fact to note from this algorithm is that to get the same final output sequence as the initial inputs, the switching pin arrangement of the de-multiplexer of last stage must be the same as that of the multiplexer of first stage. An $N \times N$ hypercube network with n number of stages (N and n both are, obviously integer numbers) can be explained by expanding the same algorithm for higher stages. It leads to the following establishment:

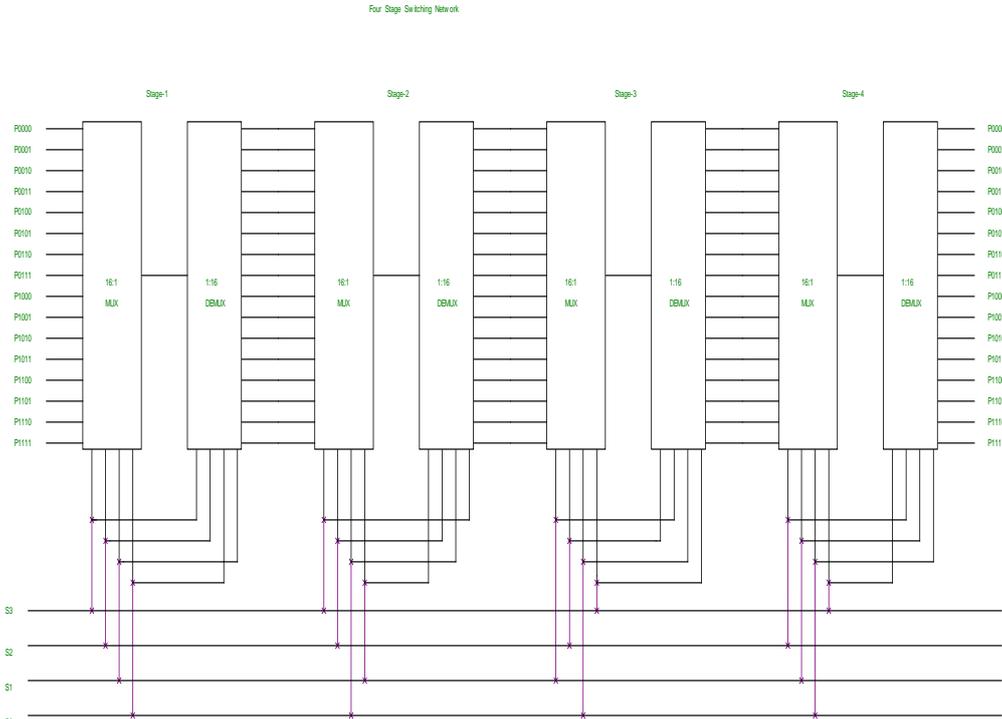


Fig.3: Complete schematic of linear 4-stage switching network

Initial switching:

$$S_n S_{n-1} S_{n-2} S_{n-3} \dots S_{n-(n-1)} S_{n-n}$$

1st stage:

$$S_n S_{n-1} S_{n-2} S_{n-3} \dots S_{n-n} S_{n-(n-1)}$$

2nd stage:

$$S_{n-(n-1)} S_{n-1} S_{n-2} S_{n-3} \dots S_{n-n} S_n$$

3rd stage:

$$S_{n-1} S_{n-(n-1)} S_{n-2} S_{n-3} \dots S_{n-n} S_n$$

4th stage:

$$S_{n-1} S_{n-2} S_{n-(n-1)} S_{n-3} \dots S_{n-n} S_n$$

5th stage:

$$S_{n-1} S_{n-2} S_{n-3} S_{n-(n-1)} \dots S_{n-n} S_n$$

.....

(n-1)th stage:

$$S_{n-1} S_{n-2} S_{n-3} \dots S_{n-(n-1)} S_{n-n} S_n$$

nth stage:

$$S_n S_{n-1} S_{n-2} S_{n-3} \dots S_{n-(n-1)} S_{n-n}$$

This universal algorithm can be used for any hypercube network with n stages which implements serial switching.

IV. Conclusion

The motivation of this paper lies in inaugurating a universally applicable algorithm for all linearly switched indirect hypercube networks. This goal has been achieved through analyzing a lower order (4 stage) network and then expanding the result to apply on a network with n stages. We have kept cross state switching out of context here. We have also excluded the combination of selection bits for that particular situation. The timing diagram or die area analysis have not been included here either. So there are plenty of areas where further work can be possible.

References

- [1] Stephen W. Keckler, KunleOlukotun, H. Peter Hofste, Multi-core processors and systems.
- [2] Sung-Mo Kang and Yusuf Leblebici, CMOS Digital Integrated Circuits, 3rd edition.http://www.ece.eng.wayne.edu/~czxu/ece7660_f05/network.pdf
- [4] John P. Hayes, "Computer Architecture and Organization", 4th Edition, 2003.
- [5] Morris Mano, "Computer System Architecture", 3rd Edition, 1992.
- [6] Young, S.D.; Yalamanchili, S. "Adaptive routing in generalized hypercube architectures", Parallel and Distributed Processing, on page(s): 564-571, Dec 1991.
- [7] Mariam, T.; Anam.,S.M.T.; Fahaduzzaman, A. S.M.; Ahmed, J.U. "Universal Functional Block Design for All Purpose Network Realization in Multi-core System", IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), e-ISSN: 2278-1676, p-ISSN: 2320-3331, Volume 9, Issue 2 Ver. V (Mar – Apr. 2014), PP 07-11, www.iosrjournals.org