Simulation Study of 4-bit and 8-bit Kogge Stone Parallel Prefix Adder Circuit Characteristics Using SILVACO

Siti Rohaya binti Ya'acob @ Selamat
Faculty of Electrical Engineering (Electronic)
Universiti Teknologi MARA
Shah Alam, Malaysia
rohayaacob@gmail.com

Abstract— **This paper describes the design of Radix-2, Radix-3 and Radix-4 of 4-bit and 8-bit Kogge Stone Parallel Prefix Adder (KSPPA) architecture. The objective is to study and investigate the effects of these different radix to the various characteristics of KSPPA in terms of logical depth, number of transistors used, propagation delay, and average power consumption. The simulation study is carried out by Gateway SILVACO EDA Tools software and the design is mapped for a 0.18µm CMOS technology with 1.8V of supply voltage. The result shows that for 4-bit KSPPA, the Radix-4 is the best design while Radix-2 is the worst design, as Radix-4 reduced logical depth by 50%, reduced transistors used as much as 23%, 3% faster, and lower average power consumption by 1.2%. Then for 8-bit KSPPA, Radix-3 is the best design while Radix-2 still the worst design, as Radix-3 reduced logical depth by 50%, reduced transistor used as much as 18%, 9% faster, and lower average power consumption by 6.6%.**

Keywords-component; KSPPA; Radix; characteristics

I. INTRODUCTION

In VLSI implementation, adders are critically an important element in processor chips because they are used in floating-point arithmetic units, arithmetic logic units (ALU), and address generation units [1]. Adders delay can significantly affect the maximum operating speed of these processors. Therefore, even a small improvement in the design of an adder can result in significant improvements in the performance of the entire processor. This has result a large number of adder architectures, which include parallel prefix adder architecture. The literature describes the parallel prefix adder are the most suitable for VLSI implementation due to their regular structure and efficient design [2, 3, 4]. In parallel prefix adder structure, the logical depth, k is determine by $\log_r n$, where n is the bit width of the input signals and r is radix. Radix represents the total fan in into *gray* cell and *black* cell in parallel prefix adder architecture.

There are several types of parallel prefix adders; they are Ladner-Fischer, Kogge-Stone, Brent-Kung, Han-Carlson, Knowles and Sklansky [5, 6]. A recent comparison among these architectures has been done using logical effort calculation. As the result, the Kogge-Stone is considered as the fastest design possible however requires larger area and consumed more power [5]. As the requirement of the adder design are; primarily fast and secondarily efficient in terms of power consumption and chip area [7], Kogge-Stone architectures is focused in this paper. This paper describes the simulation study of 4-bit KSPPA and 8-bit KSPPA by using radix-2, radix-3 and radix-4 design. The number of 2, 3 and 4 in the radix design representing the maximum fan-in that are allowed into the *gray* cell and *black* cell in each radix design respectively [7, 8]. The objective is to study and investigate the effects of different radix to the logical depth, number of transistors used, propagation delay, and average power consumption of Kogge-Stone architecture. The simulation study is carried out by Gateway SILVACO EDA Tools software and the design is mapped for a 0.18µm CMOS technology with 1.8V of supply voltage. All design constraints, such as output load (C=1pF, R=500kΩ) and transistor ratio, were held constant for each architecture. It was expected that radix-4 design would come out as the best design as revealed in [8].

II. THEORY AND DEFINITION

The complete functioning of Kogge-Stone parallel prefix adder can be easily comprehended by analyzing it in terms of three distinct part: Pre-processing, prefix carry tree, and post-processing [1, 9, 10].

A. Pre-processing

This step involves computation of carry generate bits g_i and carry propagate bits p_i corresponding to each pair of bits in A and B. These signals are given by the logic equations below:

$$p_i = a_i \oplus b_i \qquad (1)$$

$$g_i = a_i \,.\, b_i \qquad (2)$$

where;

- $0 \le i \le n-1$ which $i = 0, 1, 2, \ldots, n-1$.
- . and $\oplus$ denote the logical AND and exclusive-OR operations respectively.

The combination of equation (1) and (2) represents as the half adder.

B. *Prefix Carry Tree*

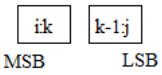
This part differentiates Kogge-Stone parallel prefix adder from other adders. This step involves computation of group propagate bit $P_{i:j}$ and group generate bit $G_{i:j}$ to compute carry signals c_i corresponding to each bit, which is given by the logic equation below:

$$G_{i:j} = G_{i:k} + \left(P_{i:k} . G_{k-1:j}\right) \qquad (3)$$

$$P_{i:j} = P_{i:k} . P_{k-1:j} \qquad (4)$$

$$c_i = G_{i:0} \qquad (5)$$

where:



Or else, c_i can be computes by:

$$c_i = g_i + (p_i . c_{i-1}) \qquad (6)$$

C. *Post-Processing*

This is the final step which involves computation of sum bits s_i. Sum bits are computed by the logic given below:

$$s_i = p_i \oplus c_{i-1} \qquad (7)$$

The input carry, C_{in} of the adder is assumed to be 0.

These three parts is illustrated in the block diagram in Fig. 1 [5]. a and b represent the n bit operands. p and g represent the carry propagate bits and carry generate bits. These signals are utilized to compute the carry c_i through the recurrence equation given in (6).

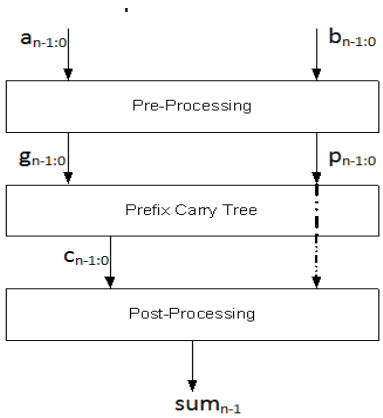


Figure 1. Block Diagram of Parallel Prefix Adder.

The prefix carry tree is an interconnection of a number of *black*, *gray* and *buffer* cells. *black* cells compute $G_{i:j}$ and $P_{i:j}$ as defined in (3) and (4) respectively. *gray* cells computes only $G_{i:j}$ [5]. *buffer* cells is optional either to put in or can contain no cells. The radix-2, radix-3 and radix-4 *black* cell and *gray* cell that represent fan-in of two, three and four respectively, and *buffer* cell are illustrated in Fig. 2.

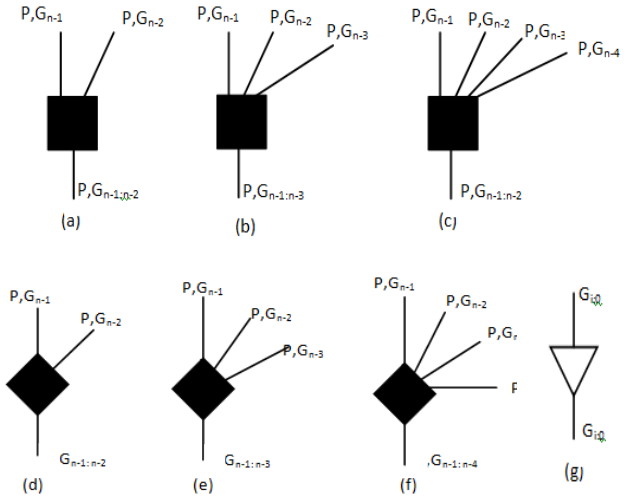


Figure 2. (a)radix-2 (b)radix-3 (c)radix-4 of *black* cell, (d)radix-2 (e)radix-3 (f) radix-4 of *gray* cell, and (g) *buffer* cell.

III. 4-BIT KOGGE STONE PARALLEL PREFIX ADDER

This study focus on the effect of different radix to the characteristics of 4-bit KSPPA and 8-bit KSPPA in terms of logical depth, number of transistors used, propagation delay and average power consumption. So upon the completion of this study, there are five main step are involved. This section describes the 4-bit KSPPA implementation first.

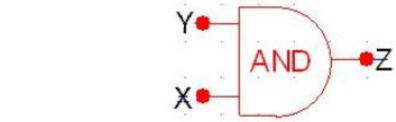
The first step is by doing theoretical analysis on each stage of parallel prefix adder and determining the radix design algorithm. At this stage, all the theoretical computation is done in order to make sure theoretical equations that are used and the comprehension of 4-bit KSPPA is right. The second step is the development on the mathematical model of 4-bit KSPPA. At this stage all the basic gates and cells needed is determine. The basic gates that are required are; OR, AND, NOT, and exclusive-OR. The cells that are required are; *black*, *gray*, *buffer*, *sum* and *half adder.*

The third step is the most complex step, which is the implementation of all the basic gates and cells required. At this stage, the process is divided into two main parts: basic gates implementation and cells implementation.

A. *Basic Gates Implementation*

The gates that are implemented are AND, OR, NOT, and exclusive OR [6]. The symbol and circuit for these gates is shown in Fig. 3, Fig. 4, Fig. 5 and Fig. 6 respectively.

1) *AND gate:* AND is implemented by the combination of NAND and NOT.

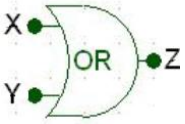


(a)

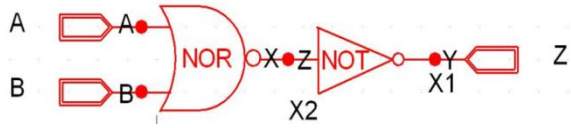
(b)

Figure 3. (a) symbol and (b) circuit of AND

2) *OR gate*: OR is implemented by the combination of NOR and NOT.



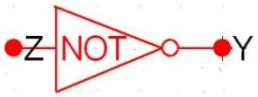
(a)



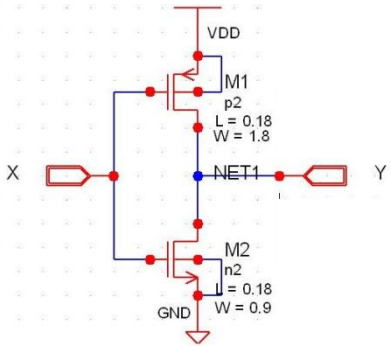
(b)

Figure 4. (a) symbol and (b) circuit of OR

3) *NOT gate*



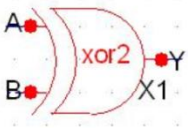
(a)



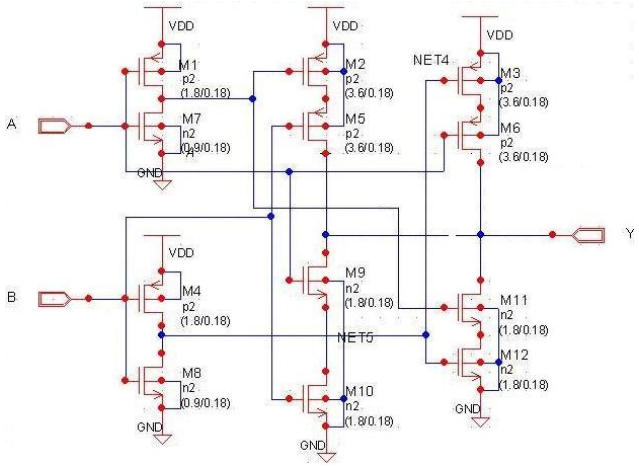
(b)

Figure 5. (a) symbol and (b) circuit of NOT

4) *Exclusive-OR*



(a)



(b)

Figure 6. (a) symbol and (b) circuit of exclusive-OR

B. Cells Implementation

There are five cells are implemented in this section. The cells are *black*, *gray*, *buffer*, *sum* and *half adder*. As different *black* cell and *gray* cell is used for different radix design, the process of *black* cell and *gray* cell implementation is divided into three parts, which is for radix-2, radix-3 and radix-4 design respectively. However there are cells used that are same for all the radix design. The cells are *half adder*, *buffer* and *sum*. *Half adder* is used in pre-processing part while *sum* is used in post-processing part. *Buffer* contained of two series NOT, *half adder* contained AND, exclusive-OR and *buffer*, and *sum* contain only exclusive-OR.

1) *Radix-2 Design: black* cell and *gray* cell for radix-2 design is named PP2 (parallel prefix-2) and GPP2 (generate parallel prefix-2) respectively. The transistor level for generate PP2 and GPP2 cell is shown in Fig. 7 and Fig. 8 respectively. The PP2 cell realize the following logic funtions for the generate output.

$$PPP2 = P_1.P_0 \quad (8)$$

$$GPP2 = G_1 + P_1.G_0 \quad (9)$$

a) PP2

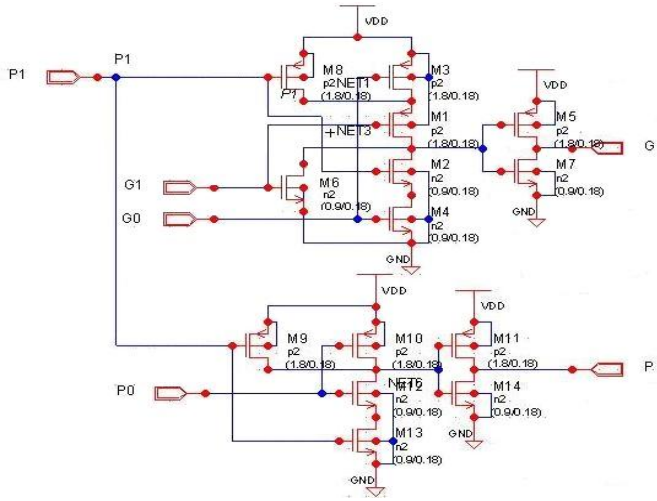


Figure 7. Transistor level of radix-2 PP2

b) GPP2

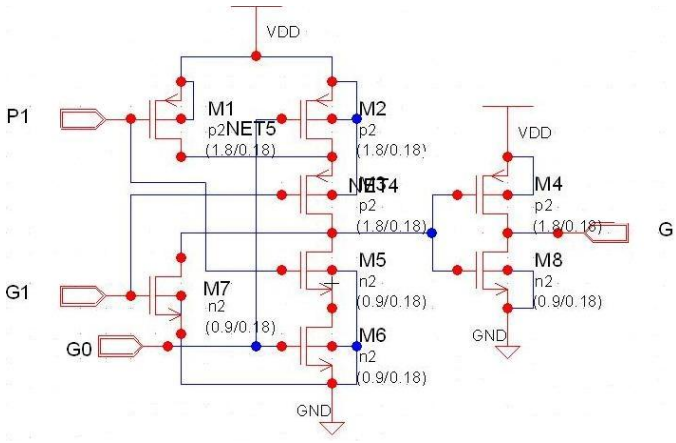


Figure 8. Transistor level of radix-2 GPP2

2) *Radix-3 design:* In radix-3, *black* cell is named PP3 and *gray* cell is named GPP3. For gray cell, radix-3 used both GPP2 and GPP3. The PP3 cell realize the following logic funtions for the generate output.

$$PPP3 = P_2.P_1.P_0 \quad (8)$$

$$GPP3 = G_2 + (P_2.(G_1 + P_1.G_0)) \quad (9)$$

The transistor level for generate PP3 and GPP3 cell is shown in Fig. 9 and Fig. 10 respectively.

a) PP3

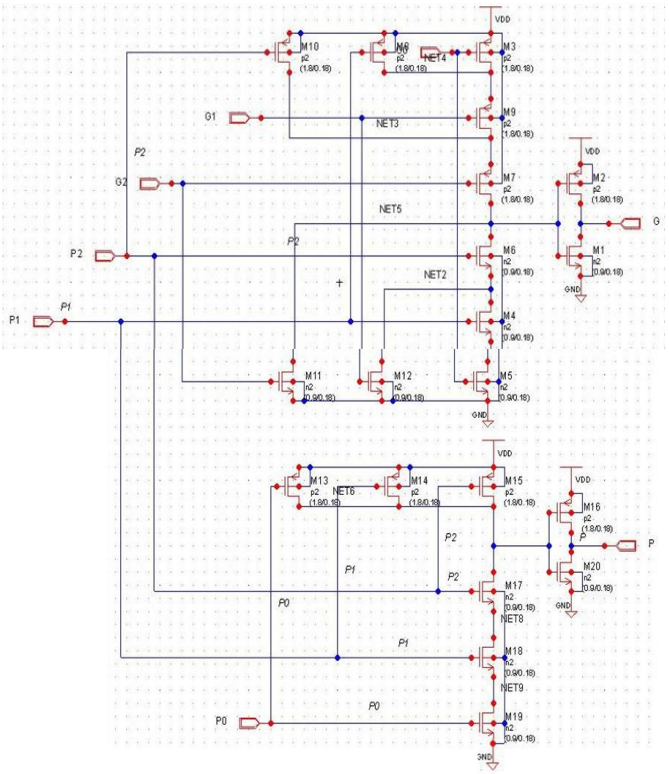


Figure 9. Transistor level of radix-3 PP3

b) GPP3

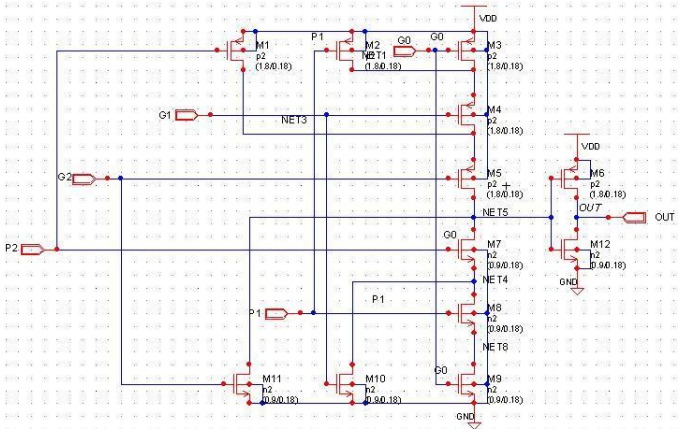


Figure 10. Transistor level of radix-3 GPP3

3) *Radix-4 design:* In radix-4, *gray* cell is named GPP4. No *black* cell is used. For *gray* cell, radix-4 used GPP2, GPP3, and GPP4. The GPP4 cell realize the following logic funtions for the generate output.

$$GPP4 = G_3 + (P_3\,(G_2 + (P_2.(G_1 + P_1.G_0)))) \quad (10)$$

The transistor level for generate GPP4 cell is shown in Fig. 11 below.

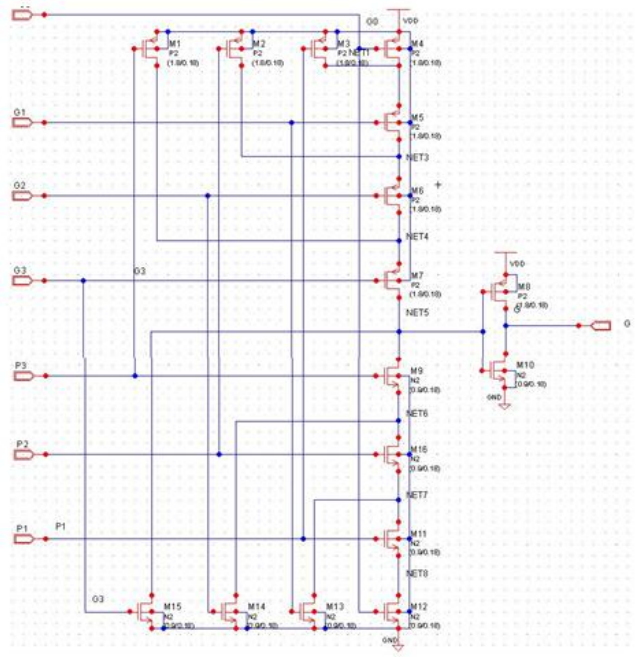


Figure 11. Transistor level of radix-4 GPP4

The next step taken after implementation process is the integration process. At this stage, the implemented cells that are required to develop the full circuit of radix-2, radix-3 and radix-4 design of the 4-bit KSPPA is integrated correspondingly. The result that is obtained at this stage is the

total number of transistors used and the logical depth for each radix design of the 4-bit KSPPA. The result will be detailed described in section V. Lastly, the final step in this study is the simulation process. At this stage, each of radix design of the 4-bit KSPPA is simulate. The simulation result that is focus on is the propagation delay and the average power consumption for 4-bit KSPPA for each radix design. The result also will be detailed described in section V.

IV. 8-Bit Kogge Stone Parallel Prefix Adder

This section described the implementation of full circuit design of radix-2, radix-3 and radix-4 of the 8-bit KSPPA. The objective is to predict the KSPPA characteristics in terms of logical depth, number of transistors used, propagation delay and average power consumption for higher bit width. The methods taken for 8-bit KSPPA implementation is the same as 4-bit KSPPA implementation. All radix design uses the same *black* and *gray* cell as in 4-bit KSPPA. The only different is, *black* cell (PP4) is used for radix-4 8-bit KSPPA implementation. The PP4 cell realizes the following logic functions for the generate output: GPP4 as in equation (10), and

$$PPP4 = P_3.P_2.P_1.P_0 \qquad (11)$$

The results that obtained in both implementation process and simulation process will be detailed described in section V.

V. Result and Discussion

This section is presenting all the results obtained from this study. There are two parts in presenting the result, first part is the result from implementation and second part is the result from the simulation.

A. Implementation Result

The results presented in this part are; the radix tree, the logical depth and the number of transistors used for each radix design of the 4-bit and 8-bit KSPPA respectively.

The figures of radix-2, radix-3 and radix-4 tree of 4-bit KSPPA are shows in Fig. 12, Fig. 13 and Fig. 14 correspondingly. Based on these figures, it shows that Radix-2 tree used two PP2 and three GPP2. Radix-3 tree used one PP3, one GPP3, and two GPP2. Radix-4 tree used one GPP4, one GPP3 and one GPP2. For *buffer* cells, the entire radix tree used the same amount that is four.

Fig. 15, Fig. 16 and Fig. 17 shows the radix-2, radix-3 and radix-4 tree of the 8-bit KSPPA respectively. Based on these figures, it shows that radix-2 tree used ten PP2, seven GPP2 and seven *buffer*. Radix-3 used five PP3, three GPP3, four GPP2 and four *buffer*. Radix-4 used four PP4, one GPP4, one GPP3, five GPP2 and five *buffer*.

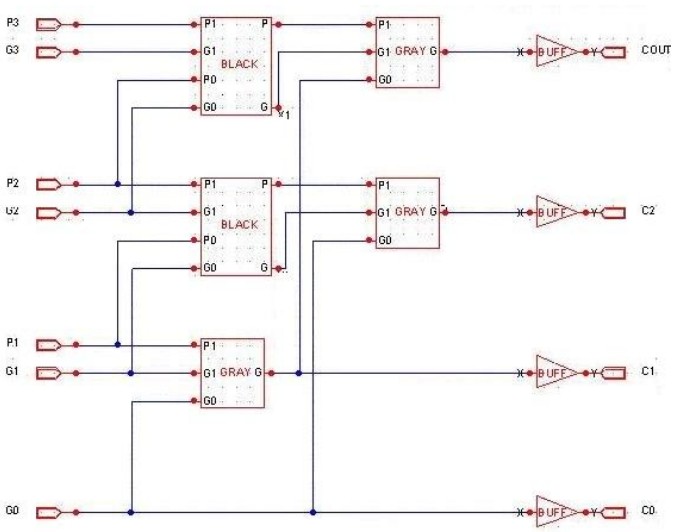


Figure 12. Radix-2 tree of 4-bit KSPPA

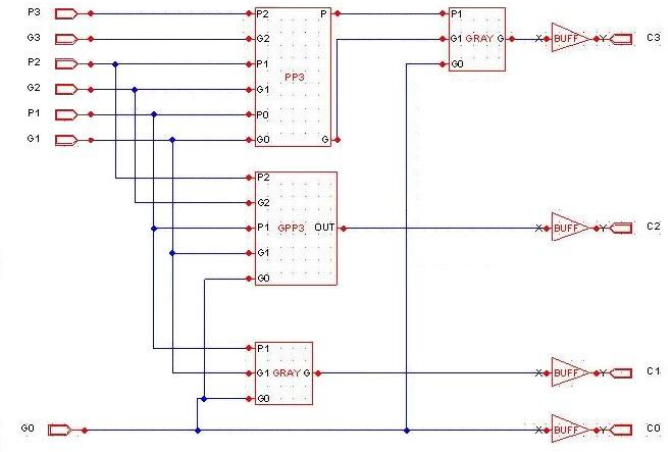


Figure 13. Radix-3 tree of 4-bit KSPPA

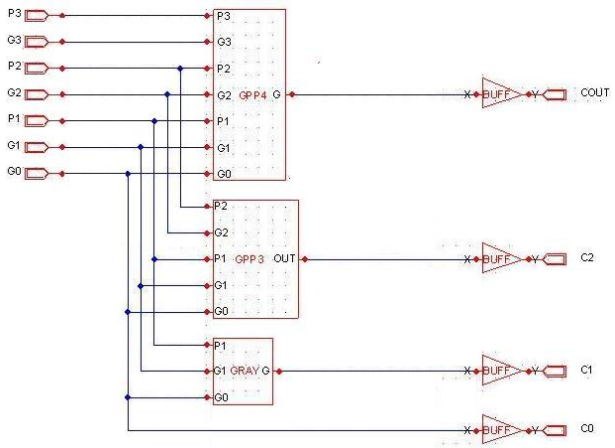


Figure 14. Radix-4 tree of 4-bit KSPPA

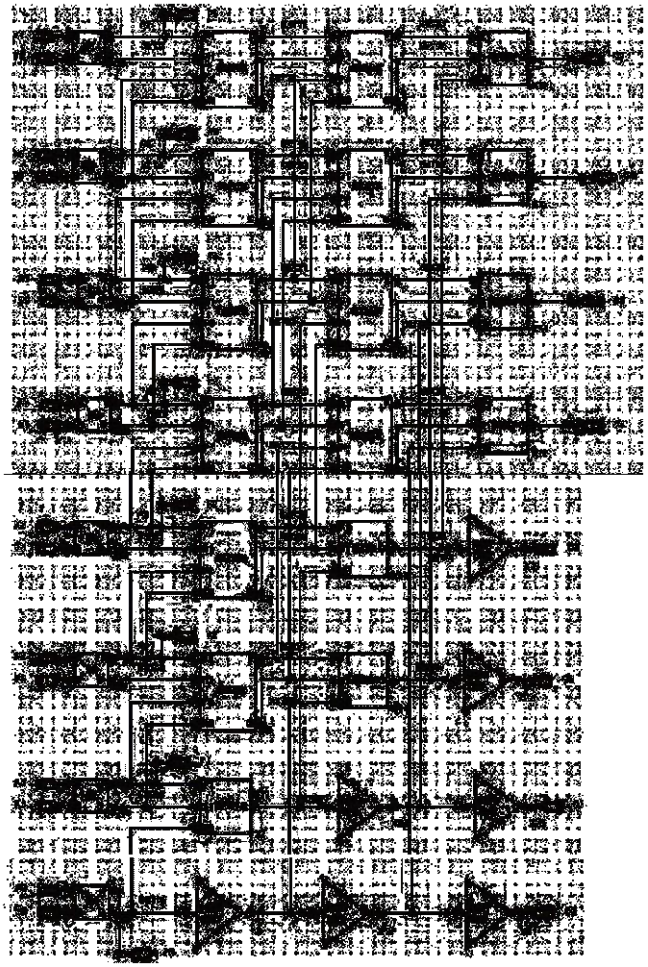


Figure 15. Radix-2 tree of 8-bit KSPPA

In better understanding on the computation process of the number of transistors used, all the cells count in each radix tree of 4-bit and 8-bit KSPPA tabulated in Table I, II and III, representing radix-2, radix-3 and radix-4 respectively. Table IV is the summarization of these tables.

TABLE I. TOTAL NUMBER OF TRANSISTOR USED FOR RADIX-2

Cell	**number of transistor in each cell**	**Amount of cells used**	
		4-bit KSPPA	***8-bit KSPPA***
PP2	14	2	10
GPP2	8	3	7
buffer	4	4	7
Total number of transistor used		68	224

TABLE II. TOTAL NUMBER OF TRANSISTOR USED FOR RADIX-3

Cell	**number of transistor in each cell**	**Amount of cells used**	
		4-bit KSPPA	***8-bit KSPPA***
PP3	20	1	5
GPP3	12	1	3
GPP2	8	2	4
buffer	4	4	4
Total number of transistor used		64	184

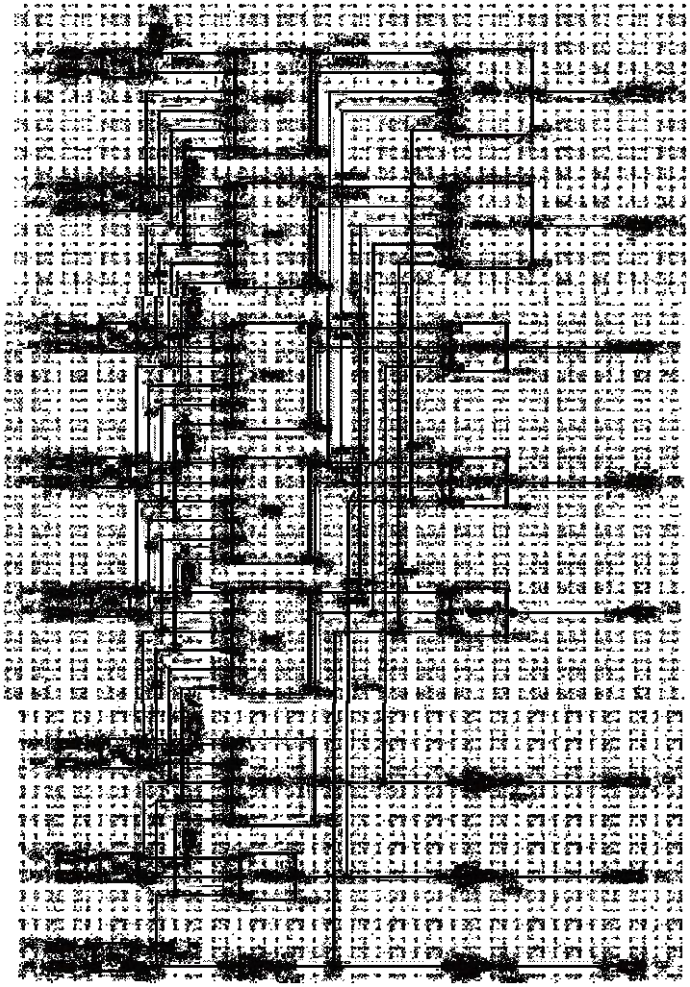


Figure 16. Radix-3 tree of 8-bit KSPPA

TABLE III. TOTAL NUMBER OF TRANSISTOR USED FOR RADIX-4

Cell	**number of transistor in each cell**	**Amount of cells used**	
		4-bit KSPPA	***8-bit KSPPA***
PP4	26	0	4
GPP4	16	1	1
GPP3	12	1	1
GPP2	8	1	5
buffer	4	4	5
Total number of transistor used		52	192

Based on Table IV, it shows for 4-bit KSPPA, radix-4 used least transistors while radix-2 used highest transistors. The percentage of transistors reduced by radix-4 than radix-2 is 23%. Then for 8-bit KSPPA, it shows that radix-3 uses least transistors and radix-2 remained uses highest transistors. By radix-3, the percentage reduced is as much as 18%.

TABLE IV. NUMBER OF TRANSISTORS USED

	Number of transistors	
	4-bit	***8-bit***
Radix-2	68	224
Radix-3	64	184
Radix-4	52	192

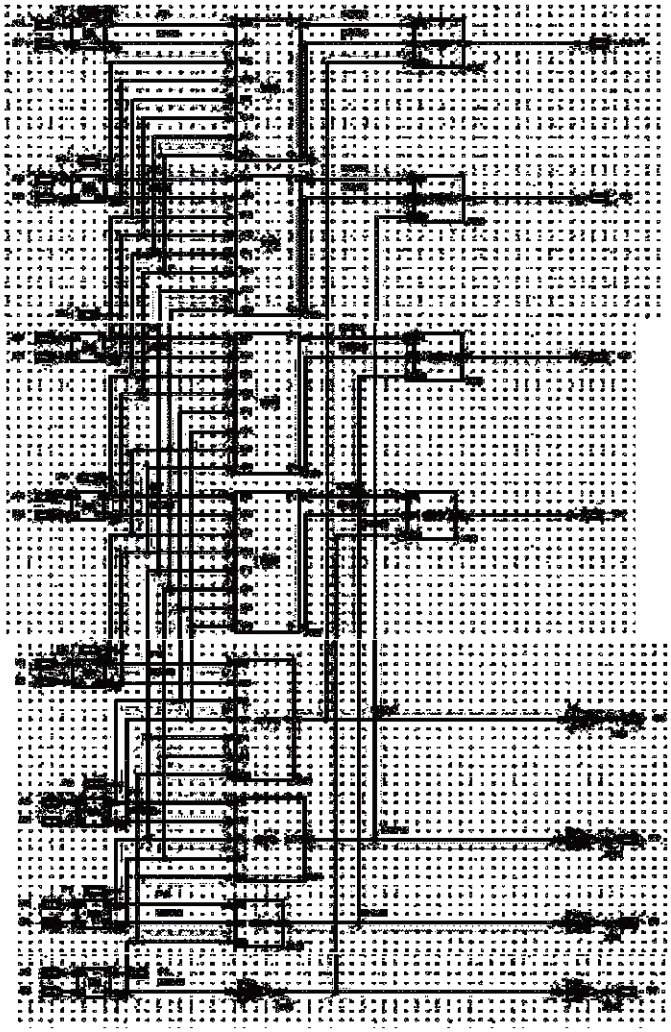


Figure 17. Radix-4 tree of 8-bit KSPPA

Table V shows the computation of logical depth. Based on Table V, it clearly shows that for both 4-bit and 8-bit KSPPA, the logical depth of radix-4 reduced by 50% compared to radix-2.

TABLE V. LOGICAL DEPTH

Logical depth, $k = log_r n$	**Kogge Stone parallel prefix adder**	
	4-bit	***8-bit***
Radix- 2	2.0	3.0
Radix-3	1.3	1.9
Radix-4	1.0	1.5

B. Simulation Result

In this part, the result of simulation waveform, propagation delay, and average power consumption of each radix design of the 4-bit KSPPA and 8-bit KSPPA is presented. These results presented in three main divisions. The first division is the result of simulation waveform, second division is the propagation delay and third division is the average power consumption.

1) *Simulation waveform*: In this division, the Fig. 18 shows the simulation waveform for radix-2 of 4-bit KSPPA. Based on Fig. 18, it showed that for the given input A and B of 1111 and 1010 respectively, the output produced is 1001 with Cout of 1. Same goes with radix-3 and radix-4, as the input A and B given is 1111 and 1010 respectively, the output produced is 1001 with Cout of 1. Hence it is proven that all the radix design of 4-bit KSPPA is correct.

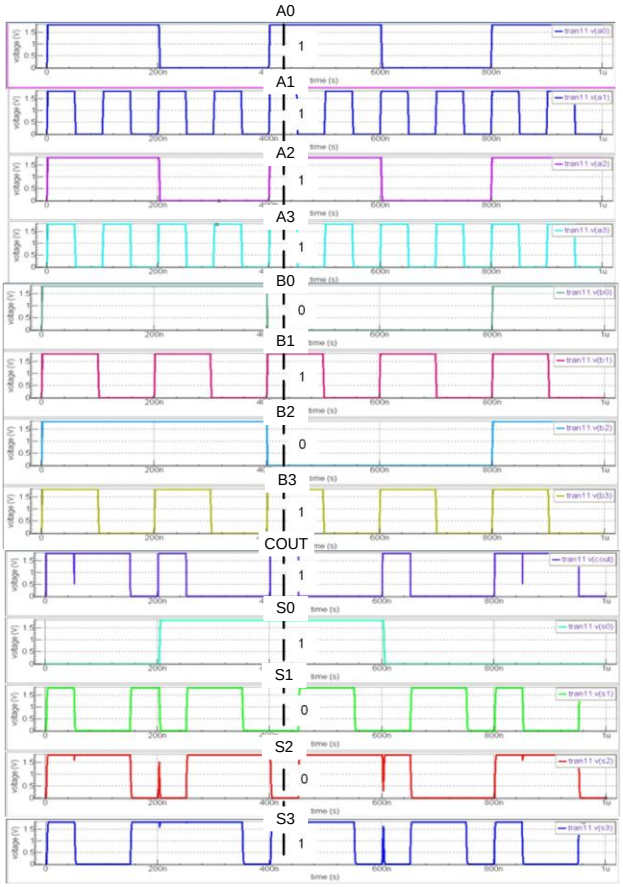


Figure 18. The simulation result of radix-2 of 4-bit KSPPA

Fig. 19 shows the simulation waveform for radix-2 of 8-bit KSPPA. Based on Fig. 19, shows that for the given input A and B of 10101010 and 11111111 respectively, the output produced is 10101001 with Cout of 1. Similar as radix-3 and radix-4, for the input A and B of 10101010 and 11111111 respectively, the output produced is 10101001 with Cout of 1. Therefore it is proven that the design of radix-2, radix-3 and radix-4 of the 8-bit KSPPA is truthful.

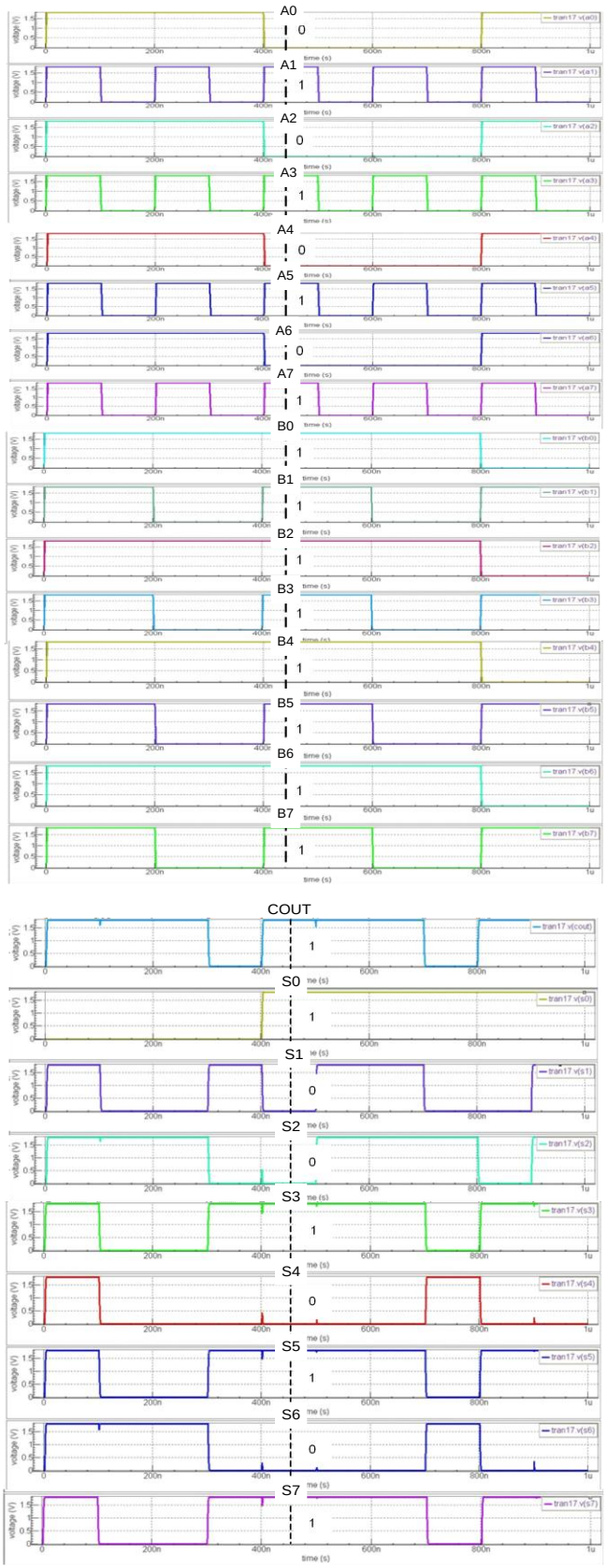


Figure 19. The simulation result of radix-2 of 8-bit KSPPA

2) *Propagation delay:* this division shows the tabulated result of propagation delay. Table VI list the propagation delay of radix-2, radix-3 and radix-4 of the 4-bit KSPPA. By referring toTable VI, it shows that the radix-4 is the fastest and radix-2 is the slowest. When computed, radix-4 is faster than radix-2 by 3%.

TABLE VI. PROPAGATION DELAY OF 4-BIT KSPPA

Output	Time (s)		
	Radix- 2	***Radix- 3***	***Radix- 4***
COUT	849.53 p	850.37 p	786.12 p
S0	1.2945 n	1.2943 n	1.3018 n
S1	1.5036 n	1.5191 n	1.5312 n
S2	1.5676 n	**1.5691 n**	**1.5684 n**
S3	**1.6179 n**	1.5509 n	1.5511 n

Table VII list the propagation delay of radix -2, radix-3 and radix-4 of the 8-bit KSPPA. Based on Table VII, it shows that radix-3 is the fastest yet remained radix-2 as the slowest. When computed, radix-3 is faster than radix-2 by 9%.

TABLE VII. PROPAGATION DELAY OF 8-BIT KSPPA

Output	Time (s)		
	Radix- 2	***Radix- 3***	***Radix- 4***
COUT	1.5104 n	1.4483 n	1.4483 n
S0	1.2051 n	1.2047 n	1.2051 n
S1	**1.8854 n**	**1.7093 n**	**1.7197 n**
S2	1.7646 n	1.6042 n	1.6180 n
S3	1.6315 n	1.5899 n	1.6042 n
S4	1.6500 n	1.4833 n	1.5903 n
S5	1.5278 n	1.4835 n	1.4807 n
S6	1.5280 n	1.4683 n	1.4664 n
S7	1.5133 n	1.4506 n	1.4502 n

3) *Average Power Consumption:* Table VIII in this division shows the tabulated result of average power consumption for each radix design of 4-bit and 8-bit KSPPA. Based on this table, indicates that radix-4 consumed lowest power compared to radix-2 by 1.2% for 4-bit KSSPA. Then, for 8-bit KSPPA it shows that radix-3 consumed lowest power compared to radix-2 by 6.6%.

TABLE VIII. AVERAGE POWER CONSUMPTION

	Power (µW)		
	Radix-2	***Radix-3***	***Radix-4***
4-bit	223.58	222.19	220.97
8-bit	257.68	240.56	247.67

By referring to all the tabulated result above, this study resulting to two important findings. The first thing is, it proved that both characteristics of 4-bit KSPPA and 8-bit KSPPA is favorable when is implemented in radix-3 and radix-4 design compared to that radix-2 design to perform the same operation. This is due to radix-2 design used highest number of transistors, gone through highest logical depth, slowest propagation delay and consumed highest power for both 4-bit KSPPA and 8-bit KSPPA characteristics. Thus, as practical implementations has generally been limited to radix-2 design

[2][4], the feasible of implementation of the radix-3 and radix-4 implementation of the KSPPA can be said in this paper.

The other important result is, determining the best radix design for 4-bit KSPPA and 8-bit KSPPA. For 4-bit KSPPA, radix-4 is the best design while radix-2 is the worst, as radix-4 design reduced logical depth by 50%, reduced transistors used as much as 23%, 3% faster, and lower average power consumption by 1.2% compared to radix-2. Nevertheless, for 8-bit KSPPA radix-3 design came out as the best design instead of radix-4 design. This is because of radix-3 giving optimum value for most of the 8-bit KSPPA characteristics in terms of the number of transistors used, propagation delay and average power consumption. The radix-3 design reduced transistors used as much as 18%, 9% faster, and lower average power consumption by 6.6% compared to radix-2.

Therefore, even though radix-4 design is considered as the best design as described in [8], based on this study, it can be says that the optimum value for characteristics of KSPPA is not depends only on the radix design however is also influenced by other factors. One of the factors is the number of transistor used. As obtained in this study, when the number of transistors used in any design is least, the propagation delay will be fastest and average power consumption would be lowest. Only the value for logical depth is remained least for highest radix, as it already has unaltered formula $log_r n$.

VI. Conclusion and Recommendation

As the conclusion, it can be conclude that the radix-3 and radix-4 would be better design compared to radix-2 for both 4-bit KSPPA and 8-bit KSPPA implementation. Then, it is important to noted that the optimum value for characteristics of KSPPA is not only depends on the radix design however also depends by other factor where to be exact, in this study, the factor is the number of transistors used.

As the recommendation, the further study could be done in future by increasing the input bit width, which it was hope to study the factors that influenced the characteristics of KSPPA as the bit width is higher.

VII. Acknowledgment

The special thank goes to my helpful supervisor, Encik Syed Abdul Mutalib Al-Junid. The supervision and support that he gave truly help the progression and smoothness of this study.

References

[1] Giorgos Dimitrakopoulos and Dimitris Nikolos, "High-Speed Parallel Prefix VLSI Ling Adders," Science, vol. 54, Feb. 2005, pp. 225-231.

[2] Vibhuti Dave, Erdal Oruklu, and Jafar Saniie, "Performance Evaluation of Flagged Prefix Adders for constant Addition," Science, April 2006.

[3] Youngmoon Choi, B.S., M.S., "Parallel Prefix Adder Design", Science, December 2004.

[4] Matthew M. Ziegler and Mircea R.Stan, "A Unified Design Space for Regular Parallel Prefix Adders," Science, 2004.

[5] David Harris and Ivan Sutherland, "Logical Effort of Carry Propagation Adders,"Science, 2003, pp. 873-878.

[6] Neil H.E. Weste and David Harris, *CMOS VLSI Design: a circuits and systems perspective*. Pearson Education, 2005, pp. 15-17, pp. 689.

[7] Andrew Beaumont-Smith and Cheng-Chew Lim, "Parallel Prefix Adder Design," Science,. 2001, pp. 218-225.

[8] FrankK Gurkaynak, Yusuf Leblebici, Laurent Chaoqat and Patrick J. McGuinness "Higher Radix Kogge-Stone Parallel Prefix Adder Architectures," Science, May. 2000, pp. 609-612.

[9] Anurag Sindhu and Ashish Bhatia, "8-bit Kogge Stone Adder," Science, April 2009.

[10] Swaroop Ghosh, Patrick Ndai, Kaushik Roy, "A Novel Low Overhead Fault Tolerant Kogge-Stone Adder Using Adaptive Clocking," Science, 2008.