Enhancing Computer Processing Speed by Redesigning the Element of Its Main Memory

Samson Ogunlere, Chigozirim Ajaegbu, Olusola Maitanmi, Olawale Somefun
[1234]Computer Science Department, Babcock University, Ilishan-Remo, Nigeria
[1]ogunleres@babcock.edu.ng, [2]ajaegbuc@babcock.edu.ng,
[3]maitanmis@babcock.edu.ng, [4]somefuno@babcock.edu.ng

***Abstract* - The performance of computers is mainly determined by their speed of operation which is governed by the Processor and Memory. In order to improve the speed of a computer, both the Processor and main Memory design must be improved. Hence, this research work is aimed at increasing the main memory speed of a computer by redesigning the components that make up the computer main memory. The design of basic memory devices, using the different Memory Elements developed from practical experimentation, analytical and numerical frameworks, was done. Analysis of each design was established using Propagation Route Framework in order to determine which of them gave higher computer speed. It was observed that the number of transitions required to complete a propagation route in the proposed SET/RESET memory element tagged SR_{ALT} shows an operational maximum number of fourteen (14) transistors as against sixteen (16) transistors of the conventional SET/RESET memory element tagged SR_{CONV}. Likewise, the SR_{ALT} maximum data route delay passes through (3) gates as against SR_{CONV} that has its maximum data route delay passing through (4) gates. Thus the delay was drastically reduced, thereby increasing the speed of the computer processing when compared with the conventional (SR_{CONV}), commercially available RAMs.**

Keywords - SET/RESET Memory Element, SR_{ALT}, SR_{CONV}, Propagation Route Framework, Quantum Computing.

I. INTRODUCTION

Computing technology, from desktop and laptop computers through cell-phones and embedded computing devices in everything from automobiles and consumer appliances to life saving medical equipment permeates nearly all aspects of modern life. Computer performance over the past fifty years has seen dramatic improvement due to technological advancements in semiconductor and silicon process. These advancements have enabled the number of transistors on a single chip to double every two years as suggested by Moore's Law [1]. Processor performance has also doubled every two years in the same time period as a corollary to Moore's Law due to a combination of larger transistor budget and the increased switching speed of transistors. However, comparable increase in the performance of computers may not be as a result of increase in processor performance for all types of applications. The reason is that computer performance is governed by the interaction between the memory and processor devices. Moreover, in contrast to the rapid improvements in processor performance, memory devices performance has seen only relatively modest improvements in the past fifty years [2,], [3] as depicted in Fig. 1. The source for this gap is directly related to the gap between transistor performance progresses versus on-chip interconnect delay.

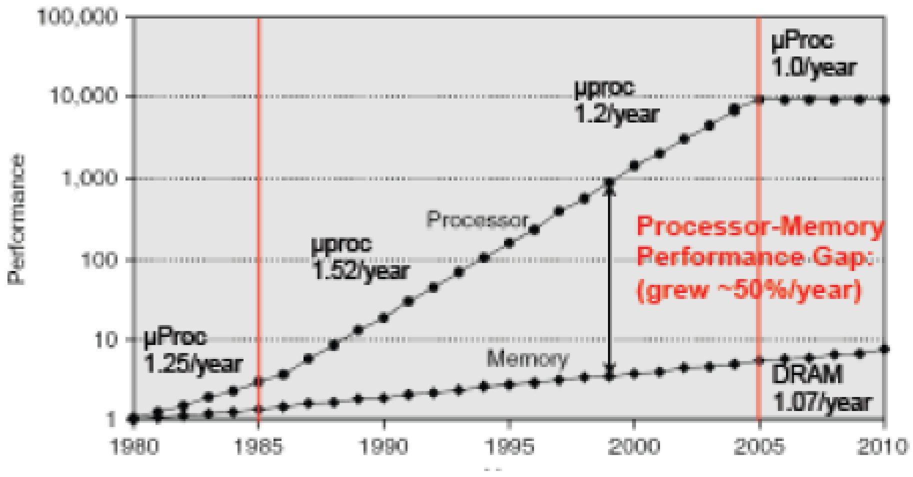

Fig. 1. Processor versus Memory Gap, [2], [3]

Printed circuit board (‘PCB’) which is used to connect processor to memory also adds significantly to the ‘gap,’ [4] as illustrated in Fig. 2.

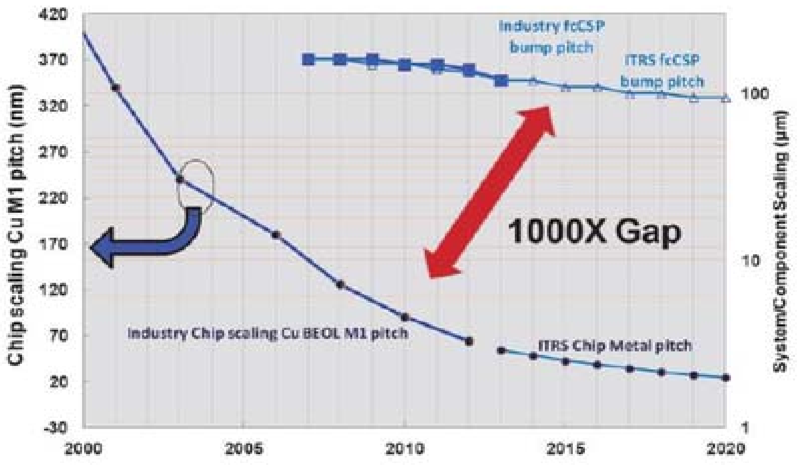

Fig. 2. Processor to Memory connection through PCB, [4]

For processors, this translates into performance, but for memory, it is not because processing power is optimized for computation while RAM is optimized for capacity. It is logical therefore to say that capacity affects performance, hence the result of the imbalance in performance between memory and processor in modern computers are increasingly inhibited by the performance of memory.

A. Growth in Processor Performance

After the release of the first microprocessor, Intel 4004 in 1971 [5], we have continually seen the launch of a new microprocessor every year, with each new one delivering significant performance improvements over previous ones unlike its memory counterpart. Some studies estimated this growth to have been exponential (in the order of about 52% per year) between 1986 and 2003 [6] as depicted in Fig. 3.

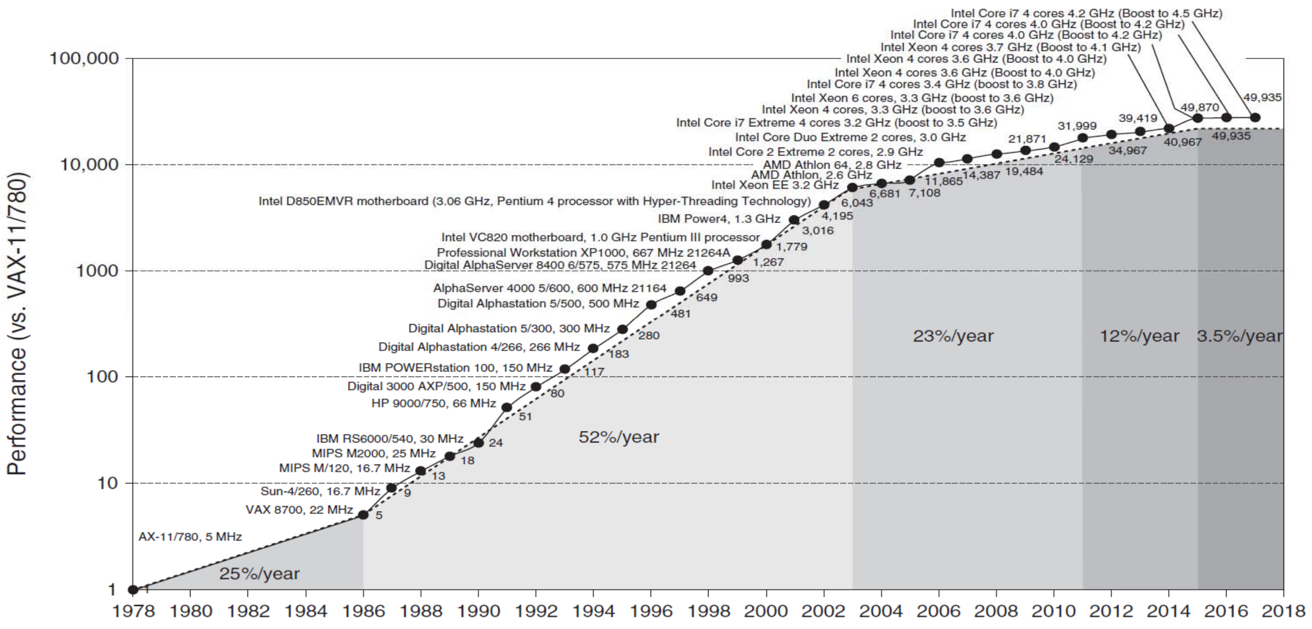

Fig. 3. Growth in processor performance over 40 years [6]

From Fig. 3, it can be seen that the early growth in processor performance averaged about 22% per year, or doubling performance every 3.5 years. Then an increase in growth to about 52% starting in 1986 or put in another way, doubling every 2 years was realized. This speed was also majorly made possible because of the emergence of the Reduced Instruction Set Computing (RISC) architecture. By 2003, the power limits were reached due to the end of Dennard scaling and the overall performance due to instruction-level parallelism significantly slowed down; hence up to 2011 we got performance growth doubling every 3.5 years.

Between 2011 and 2015, an annual improvement that was less than 12% or doubling every 8 years was achieved. This was in part due to the limits of parallelism of Amdahl's Law and the end of Moore's law. By 2015, with the impending end of Moore's Law, improvement has been reduced, just at 3.5% per year, or doubling every 20 years [6].

B. *What next after Moore's Law?*

Though microprocessor has evolved all through the years with probably higher speed of operations than its memory counterpart; however, with the end of Dennard scaling, the impending complete demise of Moore's law and huge limits in parallelism due to Amdahl's law, there is the need to start seeing microprocessor and general computing in a new light. According to [6], there is a need for drastic change in computer architecture from general-purpose cores to Domain-Specific Architectures (DSAs). A computer of the future should consist of standard processors to run conventional large programs such as operating systems along with domain-specific processors that do only a narrow range of tasks extremely well [6]. This implies that computer engineers and architects need to rethink the way computers are built in order to leverage on fast processing and implementation of some Machine Learning Algorithms on huge dataset. Hence it is imperative that Architects and Engineers must be aware of the environment for which they must design a computer for.

For these reasons, it is highly imperative to re-analyze and re-design an efficient and high performance memory element for enhanced computer processing speed that can compete favorably with its processor counterpart while waiting for the proposed Quantum Computing Technology or other technologies that will take over from Moore's Law. One way for chip designers to overcome the slowing down of advances in general purpose chips is to make ever more specialized processors and memories. Graphics processing units (GPUs), Custom specialized processors for neural networks, computer vision for self-driving cars, voice recognition, and Internet of Things devices are just some few examples. These special designs can boast a range of improvements for greater levels of performance.

Thus, the optimization of existing hardware structures is necessary when the requirement of the memory element is for low-power, high-speed or low-noise applications. Even if Moore's Law was to end tomorrow, optimizing today's software would still provide years, if not decades, of growth, with little hardware improvements. Whether it is new configuration of machines, chips made out of entirely new materials, or new types of subatomic research that open up new ways of packing transistors on to chips, it is believed that the future of computing, with all the ingenuity it involves will continue to be reckoned with.

II. DESIGNING A FEASIBLE ALTERNATIVE OF S/R MEMORY ELEMENT

Alternative SR- memory element, to conventional one is proposed. The alternative SR- memory element is tagged **SR$_{ALT}$-memory element** and the conventional one tagged **SR$_{CONV}$ memory element** as highlighted in Tables 1a and 1b.

TABLE 1a & 1b
Building Memory Element Using SR$_{ALT}$ and SR$_{CONV}$ NOR Gate Configurations respectively

1a. SR$_{ALT}$ Memory Element

S/N	S	R	Q_n	Q_{n+1}	REMARKS
0	0	0	0	0	Resting State
	0	0	1	1	Resting State
1	0	1	0	0	Active Sate
	0	1	1	0	Active State
2	1	0	0	x	Forbidden
	1	0	1	x	Forbidden
3	1	1	0	1	Active Sate
	1	1	1	1	Active State
Transition	$0 \rightarrow 0$: S = 0,0 & R = 0,1 = [S = 0 & R = x]				
	$0 \rightarrow 1$: S = 1 & R = 1 = [S = 1 & R = 1]				
	$1 \rightarrow 0$: S = 0 & R = 1 = [S = 0 & R = 1]				
	$1 \rightarrow 1$: S = 0,1 & R = 0,1 = [S = x & R = x]				

1b. SR$_{CONV}$ Memory Element

S	R	Q_n	Q_{n+1}	TransitionState
0	0	0	0	Resting
0	0	1	1	Resting
0	1	0	0	Active
0	1	1	0	Active
1	0	0	1	Active
1	0	1	1	Active
1	1	0	D	Forbidden
1	1	1	D	Forbidden

Note that, Q_n = Present Output and Q_{n+1} = Future Output or Q_n = Previous Output and Q_{n+1} = Present Output, D and X = don't care term (0, 1)

A. Developing SR$_{ALT}$ Memory Element Circuit Logic Diagram Using NOR Gate

The K-map of the **SR$_{ALT}$** memory element is shown in Table 2 based on Truth Table of Table 1a. The mathematical simplification of equations (1.1) and (1.2) using De-Morgan's Theorem and Boolean algebra rules in which real circuit design of the **SR$_{ALT}$** is developed as shown in Fig. 4.

TABLE 2
K-Map for SR$_{ALT}$ Memory Element

00	01	11	10
0^0	0^2	1^6	X^4
1^1	0^3	1^7	X^5
$\bar{Q}_{n+1} = \bar{S}R + \bar{Q}_n\bar{S}$			
$Q_{n+1} = S + Q_n\bar{R}$			

From the K-map of Table 2, equations 1.1 and 1.2 were derived concerning the **SR$_{ALT}$** memory element as the mathematical simplification using De-Morgan's Theorem and Boolean algebra rules for the **SR$_{ALT}$** memory element is shown in Table 3.

TABLE 3
Mathematical Simplification Using De-Morgan's Theorem

$\bar{Q}_{n+1} = \bar{S}\bar{Q}_n + \bar{S}R$................ (1.1)	$Q_{n+1} = S + Q_n\bar{R}$................ (1.2)
$\overline{\overline{\bar{S}\bar{Q}_n}} = \overline{Q_n + S}$	$\overline{\overline{Q_n\bar{R}}} = \overline{\bar{Q}_n + R}$
$\overline{\overline{\bar{S}R}} = \overline{\bar{R} + S}$	$Q_{n+1} = S + \overline{\bar{Q}_n + R}$
$\bar{Q}_{n+1} = \overline{Q_n + S} + \overline{\bar{R} + S}$	$\bar{Q}_{n+1} = \overline{S + \overline{\bar{Q}_n + R}}$................ (1.2a)
$Q_{n+1} = \overline{\overline{Q_n + S} + \overline{\bar{R} + S}}$.............. (1.1a)	

The overall equations are as follows:

$$\bar{Q}_{n+1} = \bar{S}R + \bar{Q}_n\bar{S} \quad \text{..} \quad \textbf{(1.1)}$$

$$Q_{n+1} = S + Q_n\bar{R} \quad \text{..} \quad \textbf{(1.2)}$$

Combining equations 1.1a and 1.2a will produce the circuit diagram of Fig. 4.

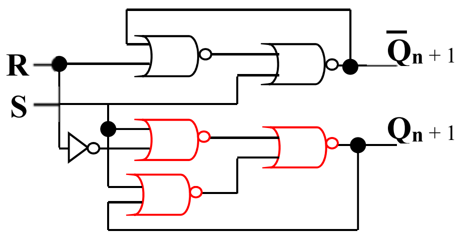

Fig. 4. Logic Circuit of SR$_{ALT}$ Using NOR gates configuration (Positive Logic Design)

B. Developing SR$_{CONV}$ Memory Element Circuit Logic Diagram Using NOR Gate

The Truth Table of Table 1b is converted into a K-Map in order to obtain the minimized logic equations of the SR$_{CONV}$ as shown in Table 4.

TABLE 4
K-Map for SR_{CONV} Memory Element

(SR-FF) Using NOR Gates				
Q_n	00	01	11	10
0	0	0	d	1
1	1	0	D	1

$$Q_{n+1} = S + \bar{R}Q_n \ldots\ldots (2.1)$$
$$\bar{Q}_{n+1} = R + \bar{S}Q_n \ldots\ldots (2.2)$$

Logic equations (2.1) and (2.2) are derived from the K-Map, and they can be used to construct the memory element using NOR gate configuration as given by the equations. The mathematical analysis of these equations using De-Morgan's theorem and Boolean algebra rules is as follows:

$$Q_{n+1} = S + \bar{R}Q_n \ldots\ldots (2.1)$$
$$\therefore \qquad \overline{\overline{\bar{R}Q_n}} = \overline{R + \bar{Q}_n} \ldots\ldots (2.1a)$$

Put equation (2.1a) into (2.1), we have

$$Q_{n+1} = S + \overline{R + \bar{Q}_n} \ldots\ldots (2.1b)$$

Complement equation (2.1b), we have

$$\bar{Q}_{n+1} = \overline{S + \overline{R + \bar{Q}_n}} \ldots\ldots (2.1c)$$

Also;

Since, $\bar{Q}_{n+1} = R + \bar{S}Q_n$ in equation 2.2,

Then,

$$\overline{\overline{\bar{S}\bar{Q}_n}} = \overline{S + Q_n} \ldots\ldots (2.1d)$$

Substituting equations (2.1d) into (2.1a), we have

$$\bar{Q}_{n+1} = R + \overline{S + Q_n} \ldots\ldots (2.1e)$$

Complementing equation (2.1e), we have

$$Q_{n+1} = \overline{R + \overline{S + Q_n}} \ldots\ldots (2.1f)$$

Combining equations (2.1c) and (2.1f) results in the construction of **SR_{CONV}** Logic Circuit Diagram, using only NOR gates (Positive Logic Design) configuration as shown in Fig. 5.

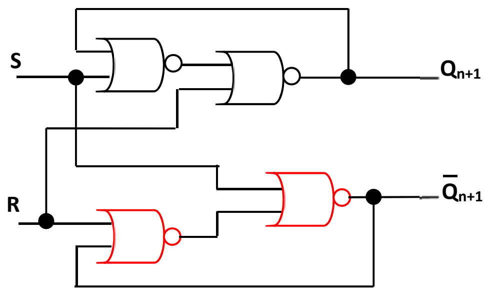

Fig. 5. Logic Circuit of SR_{CONV} Using NOR gates configuration (Positive Logic Design)

III. DATA PRESENTATION AND THE RESULTING DESIGN OF SR_{ALT} AND SR_{CONV} AT 50% ACTIVE STATES

Considering using the Set and Reset (SR) Memory Element, the Data Presentation and the Resulting Design can be derived from the Input Combination Table 5.

TABLE 5
Truth Table of Memory Elements on the SR_{ALT} and SR_{CONV}

S/N	S_e	I	W	Q_n	Q_{n+1}		S_{ALT}	R_{ALT}		S_{CONV}	$R_{CONV.}$
0	0	0	0	0	0		0	x		0	X
1	0	0	0	1	1		x	x		X	0
2	0	0	1	0	0		0	x		0	X
3	0	0	1	1	1		x	x		X	0
4	0	1	0	0	0		0	x		0	X
5	0	1	0	1	1		x	x		X	0

6	0	1	1	0	0		0	x	0	X
7	0	1	1	1	1		x	x	X	0
8	1	0	0	0	0		0	x	0	X
9	1	0	0	1	1		x	xx	X	0
10	1	0	1	0	0		0	x	0	X
11	1	0	1	1	0		0	1	0	1
12	1	1	0	0	0		0	x	0	X
13	1	1	0	1	1		x	x	X	0
14	1	1	1	0	1		1	1	1	0
15	1	1	1	1	1		x	0	X	X
Input Combination Table										

The values of S_{ALT} and R_{ALT} are plotted into their respective K-Maps as shown in Table 6 from where the corresponding logic equations 3.1 and 3.2 are derived.

$$S_3 = S_e IW \text{ ………………………………………….. (3.1)}$$
$$R_3 = S_e \text{ ……………………………………………… (3.2)}$$

TABLE 6
K-MAP FOR SR_{ALT} MEMORY ELEMENT

$S_3 = S_e IW$ …. (3.1)						$R_3 = S_e$ ….. (3.2)					
S_eI		**00**	**01**	**11**	**10**	**S_eI**		**00**	**01**	**11**	**10**
WQ_n	**00**	0^0	0^4	0^{12}	0^8	**WQ_n**	**00**	x^0	x^4	x^{12}	x^8
	01	x^1	x^5	x^{13}	x^9		**01**	x^1	x^5	x^{13}	x^9
	11	x^3	x^7	x^{15}	0^{11}		**11**	x^3	x^7	x^{15}	1^{11}
	10	0^2	0^6	1^{14}	0^{10}		**10**	x^2	x^6	1^{14}	x^{10}

The resulting circuit diagram is shown in Fig. 6.

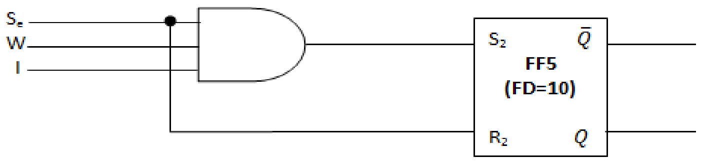

Fig. 6. Basic Memory Element Using SR_{ALT}

For the Conventional (SR_{CONV}) designated $S_{Conv.}$ & R_{Conv}; the values of S and R, are plotted into their respective K-Maps as shown in Table 7 from where the corresponding logic equations 4.1 and 4.2 are derived.

$$S = S_e IW \text{ ………………………………………………......... (4.1)}$$
$$R = S_e \bar{I} W \text{……………………………………………………………….. (4.2)}$$

TABLE 7
K-Maps for Conventional SR- Memory Element at 50% active state

K-Map for S						**K-Map for R**					
S = S_eIW ……………. (4.1)						R = $S_e \bar{I} W$……………. (4.2)					
		S_eI						**S_eI**			
WQ_n		**00**	**01**	**11**	**10**	**WQ_n**		**00**	**01**	**11**	**10**
	00	0^0	0^4	0^{12}	0^8		**00**	X^0	X^4	X^{12}	X^8
	01	X^1	X^5	X^{13}	X^9		**01**	0^1	d^5	0^{13}	0^9
	11	X^3	X^7	X^{15}	0^{11}		**11**	0^3	0^7	0^{15}	1^{11}
	10	0^2	0^6	1^{14}	0^{10}		**10**	X^2	X^6	0^{14}	X^{10}

When values of S and R, are plotted into their respective K-Maps as shown in Table 6, from where the corresponding logic equations (**S = S$_e$. I. W** and **R = S$_e$.** $\bar{I}$ **.W),** are derived. The resulting circuit diagram for conventional SR memory element is developed as shown in Fig. 7.

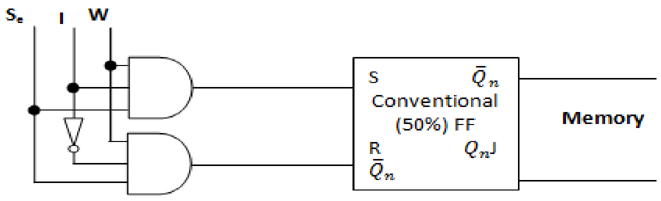

Fig. 7. Basic Memory Element Using SR_{CONV} at 50% utilization

A. *Summary of Memory Elements Design*

The summary of Basic Memory Elements of all the different Configurations is presented in Table 8.

TABLE 8
Summary of the Different Memory Element Designs

S/N	TYPE	STORAGE DEVICE
1.	**Basic Memory Element made of SR_{CONV} will be active only when SR = 00, 01 & 10 (50%)**	This is the conventional memory element used to build Storage Media
2.	**Basic Memory Elements made of the SR_{ALT} will only be active when SR = 00, 01 & 10 (50%)**	This can be used to build Storage Media This presents fewer network gate(s)

IV. COMPARATIVE PERFORMANCE ANALYSIS OF THE TWO CONFIGURATIONS WITH RESPECT TO PROPAGATION TIME FRAMEWORK.

In digital logic design, analysis of Propagation Time is a measure of performance, which in this case speed performance of computer memory. The propagation time is determined by the number of transitions required to complete a propagation route in memory element configuration. This is used to examine the performance sensitivity of the various unit configurations in other to ascertain their comparative performances. To demonstrate the utility and flexibility of this framework, it is important to know the number of transistors per gate that make up a basic memory element. This is paramount in determining the performance or how fast a memory element is. The following should be noted in using Complementary Metal Oxide Semiconductor (CMOS).

1. Inverter gate has 2Transistors
2. AND gate has 6Transistors
3. OR gate has 6Transistors
4. NAND gate has 4Transistors, and
5. NOR gate has 4Transistors

A. *Determination of Transition Routes for SR_{CONV} Memory Element*

Starting with SR conventional (**SR_{CONV}**) memory elements, the propagation time for this is examined in Fig. 8 where 'T' represents Transistor(s) and 'G' represents Gate(s). Table 9 shows the number of transitions required to complete a propagation route in this configuration.

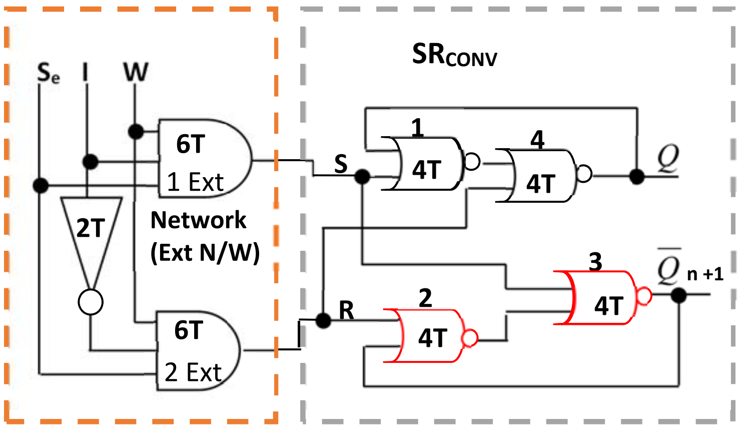

Fig. 8. Basic Memory Element using $_{SRCONV}$ 50%

TABLE 9
Number of transitions required for the SR_{CONV} Memory Element Configuration

SIGNAL	**TRANSITION ROUTE**	**REMARKS**
Select, S_e	3-Input AND-gate 1 Ext→S→2-Input NOR-gate 1→2-Input NOR-gate 4→ ***Q***	One 3-Input AND-gate 1 Ext has [6T] Two 2-Input NOR-gate 1 & 4 has [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 1 Ext→S→2-Input NOR-gate 3→$\bar{Q}$	One 3-Input AND-gate 1 Ext has [6T] One 2-Input NOR-gate 3 has [4T] **Total No of Gates [Transistors] = 2 [10T]**
	3-Input AND-gate 2 Ext→R→2-Input NOR-gate 2→2-Input NOR-gate 3→$\bar{Q}$	One 3-Input AND-gate 2 Ext has [6T] Two 2-Input NOR-gate 2 & 3 have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 2 Ext→R→2-Input NOR-gate 4→ ***Q***	One 3-Input AND-gate 2 Ext has [6T] One 2-Input NOR-gate 4 has [4T] **Total No of Gates [Transistors] = 2 [10T]**
Write Command, W	3-Input AND-gate 1 Ext→S→2-Input NOR-gate 1→2-Input NOR-gate 4→ ***Q***	One 3-Input AND-gate 1 Ext has [6T] Two 2-Input NOR-gate 1 & 4 have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 1 Ext→S→2-Input NOR-gate 3→$\bar{Q}$	One 3-Input AND-gate 1 Ext has [6T] One 2-Input NOR-gate 3 has [4T] **Total No of Gates [Transistors] = 2 [10T]**
	3-Input AND-gate 2 Ext→R→2-Input NOR-gate 2→2-Input NOR-gate 3→$\bar{Q}$	One 3-Input AND-gate 2 Ext has [6T] Two 2-Input NOR-gate 2 & 3 have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 2 Ext→R→2-Input NOR-gate 4→ ***Q***	One 3-Input AND-gate 2 Ext has [6T] One 2-Input NOR-gate 4 has [4T] **Total No of Gates [Transistors] = 2 [10T]**
Data to be written, I	Inverter→3-Input AND-gate 1 Ext→S→2-Input NOR-gate 1→2-Input NOR-gate 4→ ***Q***	One Inverter has [2T] One 3-Input AND-gate 1 Ext has [6T] Two 2-Input NOR-gate 1 & 4 have [8T] **Total No of Gates [Transistors] = 4 [16T]**
	Inverter→3-Input AND-gate 1 Ext→S→2-Input NOR-gate 3→$\bar{Q}$	One Inverter has [2T] One 3-Input AND-gate 1 Ext has [6T] One 2-Input NOR-gate 3 has [4T] **Total No of Gates [Transistors] = 3 [12T]**
	Inverter→3-Input AND-gate 2 Ext→R→2-Input NOR-gate 2→2-Input NOR-gate 3→$\bar{Q}$	One Inverter has [2T] One 3-Input AND-gate 1 Ext has [6T] Two 2-Input NOR-gate 2 & 3 have [8T] **Total No of Gates [Transistors] = 4 [16T]**
	Inverter→3-Input AND-gate 2 Ext→R→2-Input NOR-gate 4→ ***Q***	One Inverter has [2T] One 3-Input AND-gate 1 Ext has [6T] One 2-Input NOR-gate 4 has [4T] **Total No of Gates [Transistors] = 3 [12T]**
NOTE: The maximum delay is caused by the data route passing through four (4) gates with 16 transistors		

In determining the transition route for SR_{CONV} memory element, the propagation time is determined by the number of gates the signals have to pass through from the inputs to its outputs. Having known the number of gates, the number of Transistors is determined as shown in Table 9 with 16 Transistors as maximum delay caused by data route passing through 4 gates.

B. *Determination of Transition Routes for SR_{ALT} Memory Element*

For the **SR_{ALT}** memory element configuration, the propagation time route is analysed as shown in Fig. 9 Table 10 respectively.

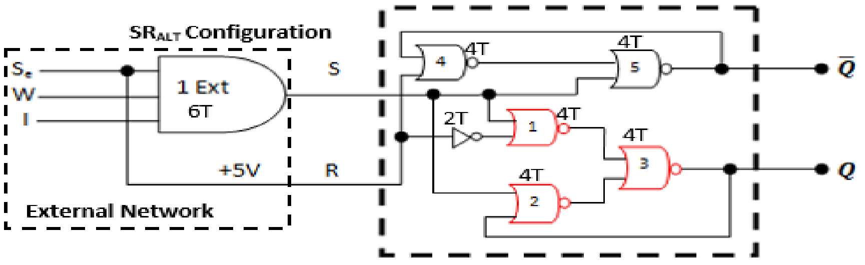

Figure 9: Basic Memory Element using SR_{ALT} Configuration

TABLE 10
Number of Transitions Required for the SR_{ALT} Memory Element Configuration

SIGNAL	TRANSITION ROUTE	REMARKS
Select, S_e	3-Input AND-gate 1 Ext→S→2-Input NOR-gate 5→$\boldsymbol{\bar{Q}}$	One 3-Input AND-gate has [6T] One 2-Input NOR-gate has [4T] **Total No of Gates [Transistors] = 2 [10T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 1→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 2→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	R→Inverter→2-Input NOR-gate 1 & 3→ $\boldsymbol{Q}$	One Inverter has [2T] Two 2-Input NOR-gate has [8T] **Total No of Gates [Transistors] = 3 [10T]**
	R→2-Input NOR-gate 4 & 5→$\boldsymbol{\bar{Q}}$	Two 2-Input NOR-gate has [8T] **Total No of Gates [Transistors] = 2 [8T]**
Write Command, W	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 5→$\boldsymbol{\bar{Q}}$	One 3-Input AND-gate has [6T] One 2-Input NOR-gate has [4T] **Total No of Gates [Transistors] = 2 [10T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 1→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 2→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
Data to be written, I	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 5→$\boldsymbol{\bar{Q}}$	One 3-Input AND-gate has [6T] One 2-Input NOR-gate has [4T] **Total No of Gates [Transistors] = 2 [10T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 1→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
	3-Input AND-gate 1 Ext →S→2-Input NOR-gate 2→2-Input NOR-gate 3→ $\boldsymbol{Q}$	One 3-Input AND-gate has [6T] Two 2-Input NOR-gates have [8T] **Total No of Gates [Transistors] = 3 [14T]**
NOTE: RESET (R) Terminal is permanently connected to logic HIGH. So, it does not contribute to any delay that might be experienced by RAM built with this memory element. The maximum delay is caused by the data route passing through (3) gates with 14 Transistors.		

B. Summary of Comparative Performance Analysis of SR_{CONV} and SR_{ALT} Memory Element

Fig. 9, Table 10 produce a Basic Memory Element that can be used to configure RAMs of different capacities towards faster computer processing speed judging from the analysis. It should be noted that the maximum delay is caused by the data route passing through (3) gates with 14 Transistors using SR_{ALT} memory element configuration. Likewise, the maximum route delay using SR_{CONV} memory element configuration of Fig. 8, Table 9, is caused by data passing through (4) gates with 16 Transistors. Comparing the number of transitions required to complete a propagation route in both configurations; it could be seen that number of gate is reduced by one while at the same time, the number of transistor is reduced by two using the alternative **(SR_{ALT})** configuration.

Since the ultimate metric of memory system performance is related to how fast it can service critical requests from processors; the rationale used to justify the focus of this study is that by improving the Memory Element used for designing memory system, the average request service time can be reduced. This study shows remarkable speed improvement.

V. CONCLUSION

This study has established that the developed design **SR_{ALT}** memory element has a lot of added performance advantages over the conventional memory element in terms of speed (because fewer gates enhance speed; i.e., gate delay represents performance), portability (less transistors, smaller size design of devices) and reduction in cost (because it requires fewer transistors as against the conventional one). It is an established fact that in digital device design, numbers of transistors represent hardware cost.

ACKNOWLEDGEMENT

This work was supported by the School of Computing and Engineering Sciences faculty of Babcock University. The authors would like to convey deep appreciation to the Computer Science departmental research group for a thorough experimental evaluation carried out on the design models used in this work.

REFERENCES

[1] Moore G. E. “Moore's law. Conference paper on Intel Corporation”. Retrieved from en.wikipedia.org/wiki/Moore's_law, 1965
[2] Hennessy, J. L. & David A. P. “Computer Architecture: A Quantitative Approach. 4th ed.”, p. 289. Elsevier Inc. 2007
[3] McCalpin, J. D. “ Memory Bandwidth and System Balance in HPC Systems, Invited Talk, International Conference for High Performance Computing, Networking, Storage, and Analysis, 2016; Available from: http://sites.utexas.edu/jdm4372/2016/11/22/sc16.
[4] Yeap, G. “Smart mobile SOCs driving the semiconductor industry”; Technology trend, challenges and opportunities. In Electron Devices Meeting (IEDM), 2013
[5] Federico, F. (n.d.). “The Silicon Gate Design of the 4004”. Retrieved December 01, 2018, from Intel 4004. Available from: http://www.intel4004.com/sgdm.htm
[6] Hennessy, J. L., & Patterson, D. A. “Computer Architecture: A Quantitative Approach (6th edition)”. Cambridge, MA: Elsevier Inc., 2019.