

IMPLEMENTATION OF SYNCHRONOUS UP COUNTER BY USING SELF RESETTING LOGIC

VIJAYA BHASKAR M, SUPARSHYA BABU SUKHAVASI*,
SUSRUTHA BABU SUKHAVASI*, G SANTHI SWAROOP VEMANA **.**

*Assistant Professor, Department of ECE, K L University, Guntur, AP, India.
**M.Tech -VLSI Students, Department of ECE, K L University, Guntur, AP, India.

Abstract— Self-resetting logic is a commonly used piece of circuitry that can be found in use with memory arrays as word line drivers. Self resetting logic implemented in dynamic logic families have been proposed as viable clock less alternatives. While these circuits can produce excellent performance, they display serious limitations in terms of area cost and power consumption. A middle of the road alternative, which can provide a good performance and avoid the limitations seen in dynamic self resetting circuits, would be to implement self resetting behavior in static circuits. This alternative has been introduced recently as self resetting logic. The dynamic circuits are becoming increasingly popular because of the speed advantage over static CMOS logic circuits; hence they are widely used today in high performance and low power circuits.

This paper says that by using this self resetting logic the low power VLSI circuits can be designed efficiently for counters.

Keywords— High speed, VLSI, Self-resetting logic (SRL), topologies, power dissipation

I. INTRODUCTION

Dynamic logic circuits are widely used in modern low power VLSI circuits. These dynamic circuits are becoming increasingly popular because of the speed advantage over static CMOS logic circuits; hence they are widely used today in high performance and low power circuits. Normally in the design of flip-flops and registers, the clock distribution grid and routing to dynamic gates presents a problem to CAD tools and introduces issues of delay and skew into the circuit design process [1]. There are situations that permit the use of circuits that can be automatically precharge themselves (i.e., reset themselves) after a prescribed delay. These circuits are called post charge or self-resetting logic which are widely used in memory decoders.

A novel synthesis methodology to design and verify and implemented in SRL by taking advantage of the maturity of current CAD tools. Micro wind tool is efficiently used in this paper to verify the circuits having self resetting logic, and layouts are also implemented and simulated successfully.

A fundamental difficulty with dynamic circuits is the monotonicity requirement. In the design of dynamic logic circuits numerous difficulties may arise like charge sharing, feed through, charge leakage, single-event upsets, etc. In this paper novel energy-efficient self-resetting primitive gates followed by the design of adder logic circuits are proposed.

Addition is a fundamental arithmetic operation that is generally used in many VLSI systems, such as application-specific digital signal processing (DSP) architectures and microprocessors [2]. This module is the core of many arithmetic operations such as addition/subtraction, multiplication, division and address generation. Such high performance devices need low power and area efficient adder circuits. So this paper presents a design construction for primitive gates and adder circuits which reduce delay and clock skew when compared to the dynamic logic adder implementation. The operation of primitive and adder circuits are elucidated and it is simulated using micro wind and LT-SPICE simulator and it is compared with dynamic logic circuit in terms of area, power dissipation and propagation delays at 45nm technology is carried out.

II. LIMITATIONS OF DYNAMIC CIRCUITS

In today's fast processing environment, the use of dynamic circuits is becoming increasingly popular [5]. Dynamic CMOS circuits are defined as those circuits which have an additional clock signal inputs along with the default combinational circuit inputs of the static systems. Dynamic systems are faster and efficient than the static systems.

The main problem is the potential for logic upset due to charge loss on a capacitor, and that is not acceptable to most designers. Charge may be lost via charge sharing, noise injection due to capacitive coupling, charge leakage, or α -particle hits. Once lost, it cannot be recovered and the circuit ceases to function correctly [3]. Dynamic circuits suffer from charge sharing problem because of parasitic capacitances at different nodes of a circuit. This results in lower voltage levels at the output terminals. In such circuit designs redistribution of charges takes place leading to charge sharing problems. It is common to use several stages of dynamic circuits to realize a Boolean function. Although same clock is applied to all these stages, it suffers from delay due to resistance and parasitic capacitances associated with the wire that carry the clock pulse. This delay is approximately proportional to the square of the length of the wire [4]. As a result, different amount of delays are experienced at different points in the circuit and the signal state changes that are supposed to occur in coincidence may never actually occur at the same time [9].

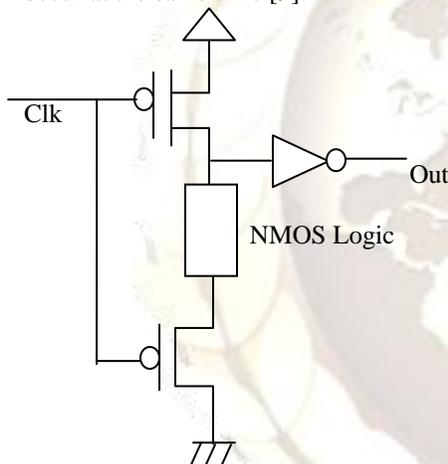


Fig. 1. Dynamic circuit

The general topology of dynamic circuit is depicted in the Fig. 1. The circuit operation is defined in two modes. They are precharge and evaluation mode. During the precharge phase (when $\text{clk} = 0$), the output node of the dynamic CMOS stage is precharged to a high logic level, and the output of the CMOS inverter (buffer) becomes low. When the clock signal rises at the beginning of the evaluation phase, there are two possibilities: the output node of the dynamic CMOS stage is either discharged to a low level through the NMOS circuitry (1 to 0), or it remains high. Consequently, the inverter output voltage can also make at most one transition during the evaluation phase, from 0 to 1. Regardless of the input voltages applied to the

dynamic CMOS stage, it is possible for the buffer output to make a 1 to 0 transition during the evaluation phase. Fig. 2 shows the primitive gates using dynamic circuit.

The largest difference between static and dynamic logic is that in dynamic logic, a clock signal is used to evaluate combinational logic. However, to truly comprehend the importance of this distinction, the reader will need some background on static logic. In most types of logic design, termed *static logic*, there is at all times some mechanism to drive the output either high or low. In many of the popular logic styles, such as TTL and traditional CMOS, this principle can be rephrased as a statement that there is always a low-impedance path between the output and either the supply voltage or the ground. As a side note, there is of course an exception in this definition in the case of high impedance outputs, such as a tri-state buffer; however, even in these cases, the circuit is intended to be used within a larger system where some mechanism will drive the output, and they do not qualify as distinct from static logic.

In contrast, in *dynamic logic*, there is not always a mechanism driving the output high or low. In the most common version of this concept, the output is driven high or low during distinct parts of the clock cycle. Dynamic logic requires a minimum clock rate fast enough that the output state of each dynamic gate is used before it leaks out of the capacitance holding that state, during the part of the clock cycle that the output is not being actively driven. Static logic has no minimum clock rate—the clock can be paused indefinitely. While it may seem that doing nothing for long periods of time is not particularly useful, it leads to two advantages:

- Being able to pause a system at any time makes debugging and testing much easier, enabling techniques such as single stepping.
- Being able to run a system at extremely low clock rates allows low-power electronics to run longer on a given battery.

In particular, although many popular CPUs use dynamic logic [6], only static cores -- CPUs designed with fully static CMOS technology -- are usable in space satellites due to their higher radiation hardness [7]. Dynamic logic, when properly designed, can be over twice as fast as static logic. It uses only the faster N transistors, which improve transistor sizing optimizations. Static logic is slower because it has twice the capacitive loading, higher thresholds, and uses slow P transistors for logic. Dynamic logic can be harder to work with, but it may be the only choice when increased processing speed is needed. Most electronics running at over 2 GHz these days require the use of dynamic, although some manufacturers such as Intel

have completely switched to static logic to save on power [8].

In general, dynamic logic greatly increases the number of transistors that are switching at any given time, which increases power consumption over static CMOS [8]. There is several power saving techniques that can be implemented in a dynamic logic based system. In addition, each rail can convey an arbitrary number of bits, and there are no power-wasting glitches. Power-saving clock gating and asynchronous techniques are much more natural in dynamic logic.

III. SELF RESETTING CMOS DYNAMIC LOGIC

Self-resetting logic is a commonly used piece of circuitry that automatically precharge themselves (i.e., reset themselves) after a prescribed delay. They find applications where a small percentage of gates switch in a cycle, such as memory decoder circuits. It is a form of logic in which the signal being propagated is buffered and used as the precharge or reset signal. By using a buffered form of the input, the input loading is kept almost as low as in normal dynamic logic while local generation of the reset assures that it is properly timed and only occurs when needed [6].

A generic view of a self-reset logic is shown in Fig.2. In the domino case, the clock is used to operate the circuit. In the self-resetting case, the output is fed back to the precharge control input and, after a specified time delay, the pull-up is reactivated. There is an NMOS sub block where the logic function performed by the gate is implemented which is represented as NMOS_LF through which the input data's are loaded. The output of the gate F provides a pulse if the logic function becomes true. This output is buffered and it is connected to PMOS structure to precharge. The delay line is implemented as a series of inverters. The signals that propagate through these circuits are pulses. The width of the pulses must be controlled carefully or else there may be contention between NMOS and PMOS devices, or even worst, oscillations may occur.

Self-resetting logic (SRL) can be classified as a variant of domino logic that allows for asynchronous operation. A basic SRL circuit is shown in Figure. A careful inspection of the schematic shows that the primary differences between this gate and the standard domino circuit are (a) the addition of the inverter chain that provides feedback from the output voltage $V_{out}(t)$ to the gate of the reset pFET M_R , and (b) the elimination of the evaluation nFET.

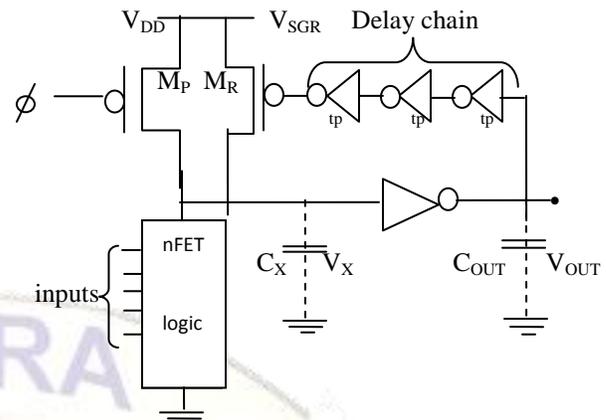


Fig.2. Basic Structure of a Self Resetting Logic Circuit

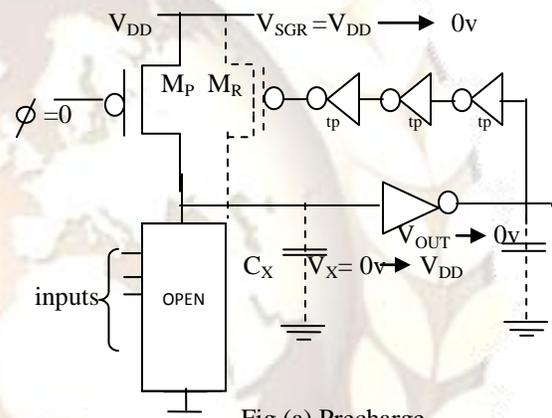


Fig (a) Precharge

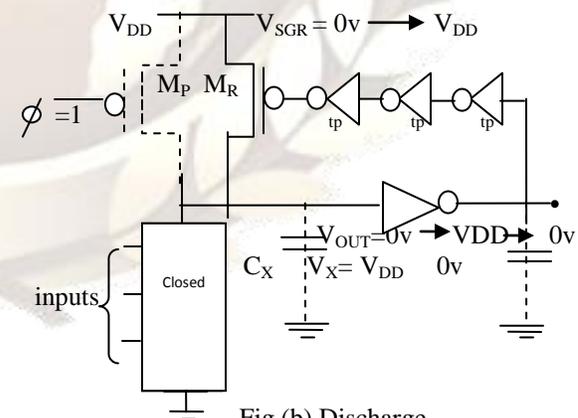


Fig (b) Discharge

Note that an odd number of inverters (3) are used in the feedback. As discussed below, the feedback loop has a significant effect on both the internal operation of the circuit and the characteristics of the output voltage. Precharging of C_X occurs when the clock is at a value $\Phi=0$ and the circuit conditions are

shown in Figure (a). During this time, and which is identical to the event in a standard domino circuit. As we will see, the timing of the input signals precludes the possibility of a DC discharge path to ground by insuring that the inputs to all logic nFET are 0 during precharge.

The voltage on the gate of MR is at a value of so that insures that MR is in cutoff during this time. The distinct features of SRL arise when a discharge occurs. The circuit conditions are shown in Figure (b).

In this case, and charges to give an output voltage of this is fed through the triple-inverter chain to drive the gate voltage of MR to 0v after a delay of where is the delay through one inverter.

Since now we have that MR is active which allows to flow and recharge back up to a voltage of This action resets the output voltage to its original precharge value of giving the logic family its name: it automatically resets its output to 0. To gain a better understanding of the voltage transitions involved in a self-resetting logic gate, let us analyze the circuit shown in figure which implements the AO function

IV. SRL PRIMITIVE GATES AND ADDERS

This section presents the basic construction and simulation of primitive gates and adders. The cells shown in Fig. is a self-resetting implementation of 2-input primitive gates. The logical functions are implemented by the NMOS stack with two input signals A and B. The delay path in this circuit is implemented with single inverter. The mechanism of self-resetting in this circuit is achieved through the PMOS transistors. The implementation of AND/OR can be obtained by placing the NMOS stack in series and parallel connection, whereas the gates NAND/NOR can be implemented with De Morgan's law, an OR gate with inverted input signals behaves as a NAND gate. Similarly an AND gate with inverted input signals behaves as a NOR gate [8].

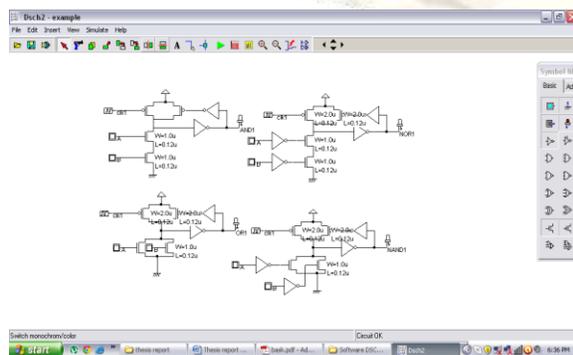


Fig 3. Schematic for AND1 OR1 NAND1 NOR1

However, it is observed that the logic functionality refers to operations on —pulses at inputs, and, if no pulses are present at the inputs, the outputs will remain at logic LOW state.

The waveforms in Fig. shows the result of spice simulation of primitive gates, implemented in a 0.12- μm CMOS process with $V_{DD} = 1.2\text{V}$. In the waveform the first three waveforms from the top correspond to input signals clk, A and B, followed by the output signals AND1, OR1, NOR1, NAND1.

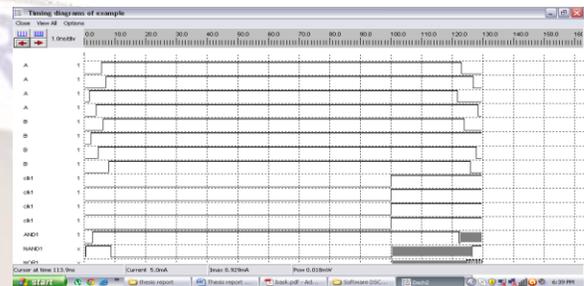


Fig 4. Simulations for and1 nand1 nor1 or1 respectively

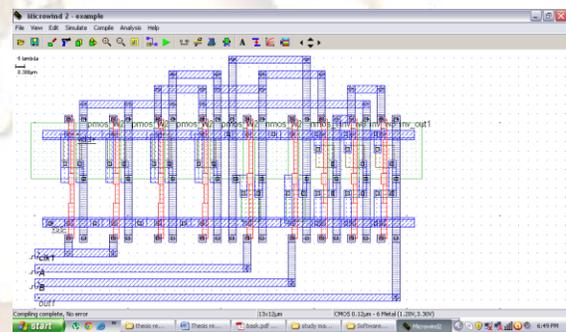


Fig 5. Layout for nand gate

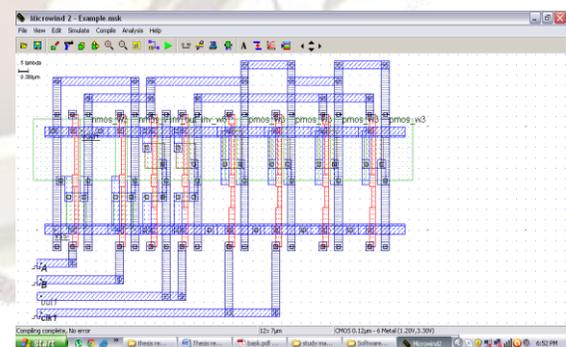


Fig 6. Layout for and gate

The SRL full adder circuit is shown in Fig. This adder consists of sum and carry block. The sum block is implemented by SRL XOR and SLR XNOR gates. The carry block is implemented with SRL AND and SLR OR gates. The input to this full adder circuit are A and B, and the outputs are SUM and Cout. Results of SPICE simulation of the adder, implemented

in a 0.12µm CMOS process, running at 500MHz data rate, with VDD = 1.2V is shown in Fig. 8.
The sum output can be obtained by

$$\text{Sum} = A \oplus B \oplus C.$$

The carry output is obtained by using the expression

$$C_{\text{out}} = AB+BC+AC.$$

The simulation cycle for this adder is 500 ns and the input combinations are fed into the system and its performance was analyzed. It is observed that the adder implemented with dynamic logic has more latency when compared to SRL adder and the power dissipation of SRL adder is less when compared to dynamic logic design. Using the SRL full adder circuit, the SRL 4-bit parallel is implemented as shown in Fig. In this adder circuit, the input to each full-adder will be A_i , B_i and C_i , and the outputs will be SUM_i and C_{i+1} , where 'i' varies from 0 to 3. Also, the carry output of the lower order stage is connected to the carry input of the next higher order stage. In the least significant stage A_0 , B_0 and C_0 (which is grounded) are added resulting in $SUM[0]$ and $C[1]$. This carry $C[1]$ becomes the carry input to the second stage. Similarly the remaining stages are executed. The simulation setup cycle for is adder is 500 ns and the input combinations are fed into the system and its performance was analyzed. Results of SPICE simulation of the adder, implemented in a 0.12µm CMOS process, running at 500MHz data rate, with VDD = 1.2V is shown in Fig. The propagation delay occurred in the parallel adders can be eliminated by carry look ahead adder. This adder is based on the principle of looking at the lower order bits of the augends and addend if a higher order carry is generated. This adder reduces the carry delay by reducing the number of gates through which a carry signal must propagate. This adder consists of three stages: a propagate block/ generate block, a sum generator and carry generator.

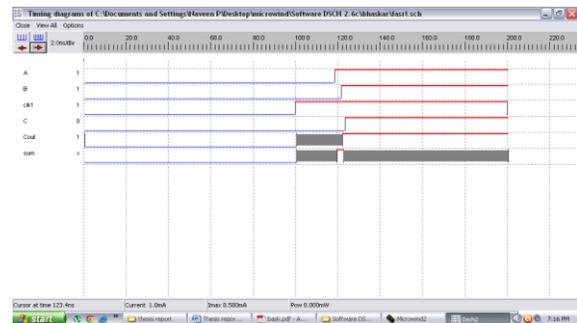


Fig 8. Simulations for full adder

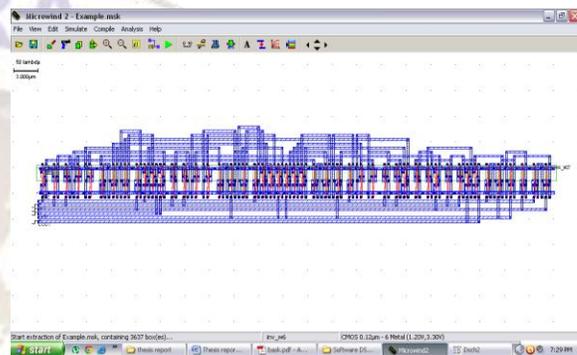


Fig 9. Layout for full adder using srl

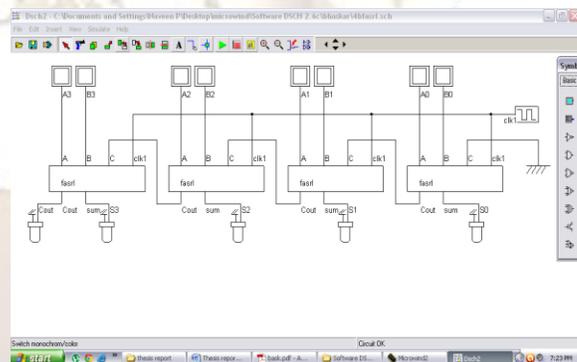


Fig 10. 4 Bit Full Adder Schematic Diagram

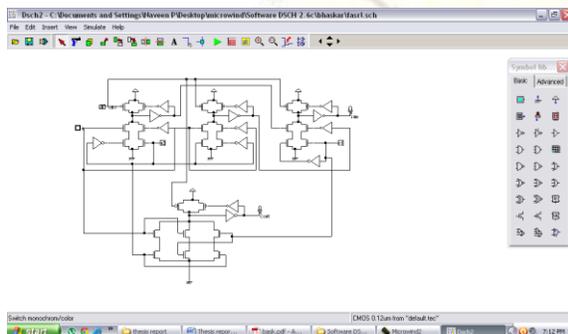


Fig 7. Full Adder Circuit

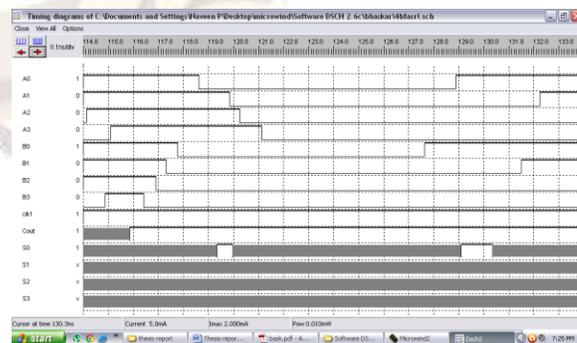


Fig 11. Simulations For 4 Bit Full Adders

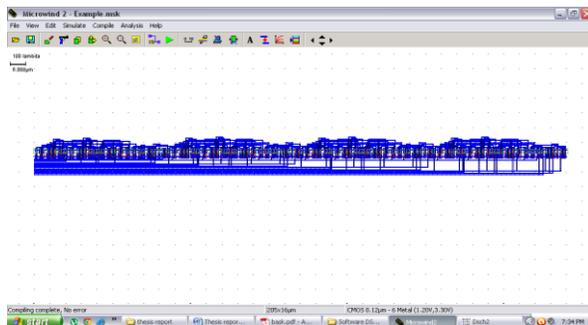


Fig 12. Layout for 4-Bit Full Adder Using SRL

V. SRL FLIP FLOPS

A. Basic R-S Flip-Flop

Basic flip-flops are designed to 'store' a logic state. The simplest flip-flop is known as the R-S flip-flop, which consists of two NAND gate wired to give negative feedback and is shown in Figure.

In normal operation then, with the reset (R) set to logic '0' the output Q will follow the logic level applied to the set (S) input. When R is set to logic '1' whatever was set on Q will remain regardless of whether the S input continues to change until the reset input is returned to logic '0'. The major drawback of this circuit is that it is not possible to predict the output when logic '1' is simultaneously applied to the S & R inputs. Therefore, more complex Flip-flops are designed to ensure that these indeterminate states do not exit and the most common circuits are D-type and J-K Flip-flops.

B. J-K Flip-Flop

The basic J-K Flip-flop is like the R-S flip-flop the outputs follow the inputs when the Clk is logic, but there are two inputs, traditionally labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs.

If J and K are both high at the clock edge then the output will toggle from one state to the other. It can perform the functions of the R-S Flip-flop and has the advantage that there are no ambiguous states. Due to the extra logic that ensures only one of the R and S inputs is enabled at any time. This prevents possible oscillation, which can occur when both inputs of an RS flip-flop are active at the same time.

The truth table of this J-K flip-flop is shown in Table 1.

J	K	CLK	Q	Q _{bar}
0	0	Pos-edge	No Change	No Change
0	1	Pos-edge	0	1
1	0	Pos-edge	1	0
1	1	Pos-edge	Toggle	Toggle

TABLE 1 Truth table for the simple J-K Flip-flop

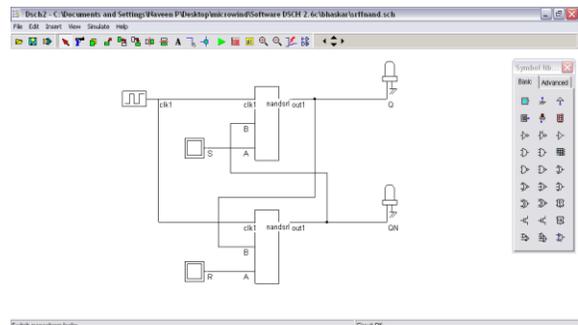


Fig 13. SR Flip Flop Using NAND Gates Designed By SR Logic

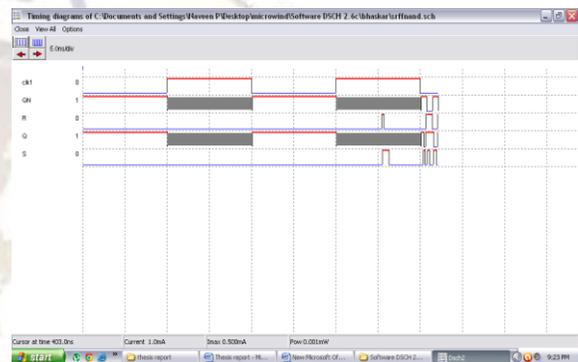


Fig 14. Simulations For SR Flip Flop

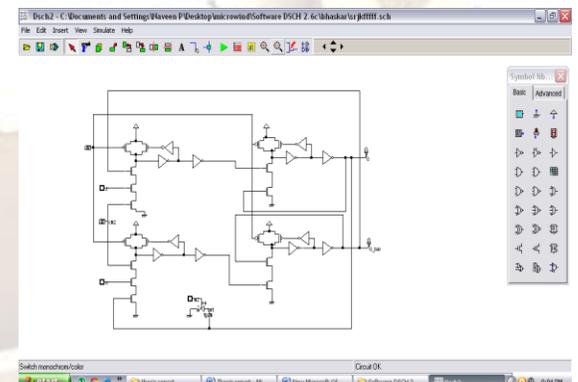


Fig 15. JK Flip Flop By SR Logic



Fig 16. Simulations For JK Flip Flop

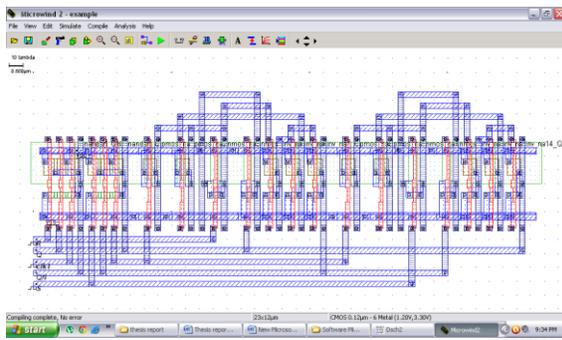


Fig 17. Layout For SR Flip Flop

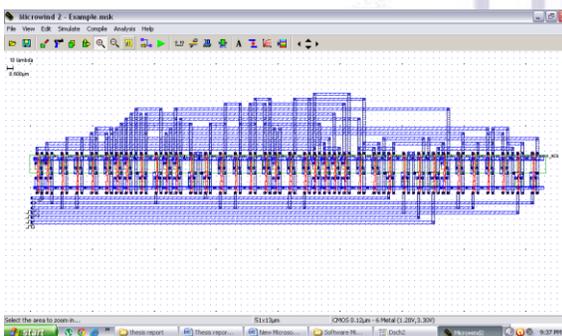


Fig 18. Layout for JK Flip Flop

VI. SRL 4-BIT COUNTER AND COMPARISON

A. Synchronous counter

A simple way of implementing the logic for each bit of an ascending counter (which is what is depicted in the image to the right) is for each bit to toggle when all of the less significant bits are at a logic high state. For example, bit 1 toggles when bit 0 is logic high; bit 2 toggles when both bit 1 and bit 0 are logic high; bit 3 toggles when bit 2, bit 1 and bit 0 are all high; and so on. Synchronous counters can also be implemented with hardware finite state machines, which are more complex but allow for smoother, more stable transitions.

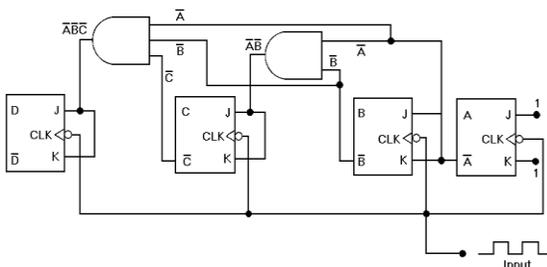


Fig 19. Synchronous counter

B. DECADE COUNTER

A decade counter is one that counts in decimal digits, rather than binary. A decade counter may have

each digit binary encoded (that is, it may count in binary-coded decimal, as the 7490 integrated circuit did) or other binary encodings (such as the bi-quinary encoding of the 7490 integrated circuit). Alternatively, it may have a "fully decoded" or one-hot output code in which each output goes high in turn (the 4017 is such a circuit). The latter type of circuit finds applications in multiplexers and demultiplexers, or wherever a scanning type of behavior is useful. Similar counters with different numbers of outputs are also common. The decade counter is also known as a mod-counter when it counts to ten (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). A Mod Counter that counts to 64 stops at 63 because 0 counts as a valid digit.

C. UP/DOWN COUNTER

A counter that can change state in either direction, under the control of an up/down selector input, is known as an up/down counter. When the selector is in the up state, the counter increments its value. When the selector is in the down state, the counter decrements the count.

D. RING COUNTER

A ring counter is a Shift Register (a cascade connection of flip-flops) with the output of the last one connected to the input of the first, that is, in a ring. Typically, a pattern consisting of a single bit is circulated so the state repeats every n clock cycles if n flip-flops are used. It can be used as a cycle counter of n states.

E. JOHNSON COUNTER

A Johnson counter (or switch tail ring counter, twisted-ring counter, walking-ring counter, or Moebius counter) is a modified ring counter, where the output from the last stage is inverted and fed back as input to the first stage.^{[2][3][4]} The register cycles through a sequence of bit-patterns, whose length is equal to twice the length of the shift register, continuing indefinitely. These counters find specialist applications, including those similar to the decade counter, digital-to-analog conversion, etc. They can be implemented easily using D- or JK-type flip-flops.

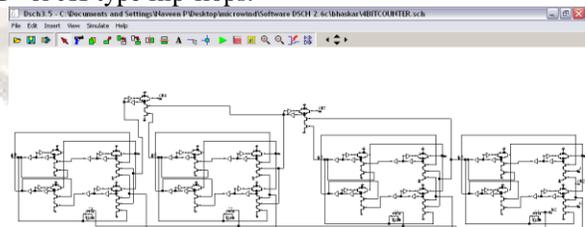


Fig 20. Schematic for 4 bit counter using self resetting logic

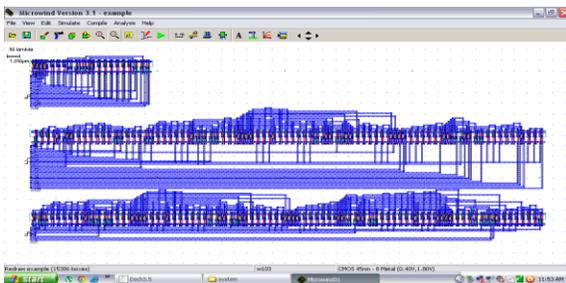


Fig 21. Layout for 4 bit counter using self resetting logic

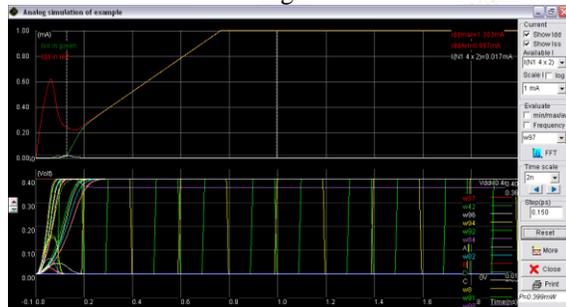


Fig 22. Power Dissipation Of 4 Bit Counter By Using SRL

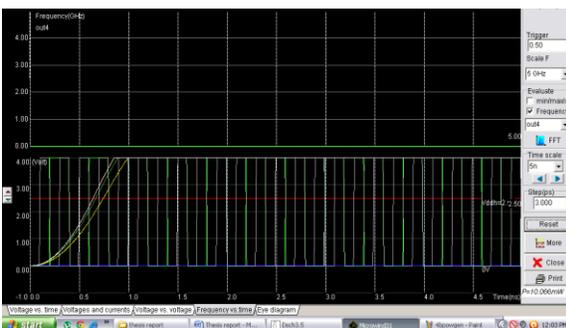


Fig 23. Power Dissipation Of 4 Bit Counter By Dynamic Logic

Comparison between conventional domino logic and self resetting logic circuits

Topology	Transistor count	Rise delay	Fall delay	Power dissipation mW	Area μm^2
AND1	8	0.005	0.001	23.12	20.34
OR1	8	0.004	0.001	23.12	20.35
NOR1	12	0.005	0.001	30.45	22.67
NAND1	12	0.015	0.002	30.45	23.56
XOR1	14	0.015	0.002	36.67	35.99
XNOR1	14	0.015	0.002	36.78	36.89

FULL ADDE R	54	0.050	0.043	45.123	100.9
PARALLEL ADDE R	216	0.506	0.498	154.54	1123.89
4 BIT COUNTER	180	0.435	0.065	2.360	330.7

Table2 Power dissipation and area of SRL circuit

Topology	Transistor count	Rise delay	Fall delay	Power dissipation mW	Area μm^2
AND1	6	0.004	0.002	26.45	18.90
OR1	6	0.002	0.001	26.45	18.92
NOR1	10	0.002	0.001	35.67	20.98
NAND1	10	0.003	0.002	36.67	20.92
XOR1	12	0.003	0.002	47.89	30.45
XNOR1	12	0.003	0.002	47.23	30.54
FULL ADDER	46	0.012	0.01	56.90	90.12
PARALLEL ADDER	184	0.1	0.12	175.65	993.89
4 BIT COUNTER	120	0.087	0.013	10.066	220.18

Table3 Power dissipation and area of dynamic logic circuit

VII. CONCLUSION

In this paper, an exhaustive analysis and design methodology for commonly used high-speed primitive gates, adder and counter circuit using self-resetting logic is implemented in 45-nm CMOS technologies. The goal was to obtain a family of gates that could simplify the implementation of fast processing circuit which overcomes the restriction due the pulses being elongated and shortened as signal traverse the logic stages. In this thesis work exhaustive comparison between conventional dynamic logic and SRL were carried in terms of its parasitic value, area and power dissipation. It is observed that the proposed circuits have offered an improved performance in power dissipation, charge leakage and

clock skew when compared to dynamic logic with additional burden of silicon area.

Hence, it is concluded that the proposed designs will provide a platform for designing high performance and low power digital circuits and high noise immune digital circuits such as, digital signal processors and multipliers.

SCOPE FOR RESEARCH

Till now self resetting logic is applied in primitive gates, adders and counters. The scope is this design can be used in digital signal processing and multipliers.

REFERENCES

- [1] Woo Jin Kim, Yong-Bin Kim, —A Localized Self-Resetting Gate Design Methodology for Low Power| IEEE 2001.
- [2] M. E. Litvin and S. Mourad, —Self-reset logic for fast arithmetic applications,| IEEE Transactions on Very Large Scale Integration Systems, vol. 13, no. 4, pp. 462–475, 2005.
- [3] L. Wentai, C. T. Gray, D. Fan, W. J. Farlow, T. A. Hughes, and R. K. Cavin, —250-MHz wave pipelined adder in 2- μ m CMOS,| IEEE Journal of Solid-State Circuits, vol. 29, no. 9, pp. 1117–1128, 1994.
- [4] D. Patel, P. G. Parate, P. S. Patil, and S. Subbaraman, —ASIC implementation of 1-bit full adder,| in Proc. 1st Int. Conf. Emerging Trends Eng. Technol., Jul. 2008, pp. 463–467.
- [5] M. Lehman and N. Burla, —Skip techniques for high-speed carry Propagation in binary arithmetic units,| IRE Trans. Electron.comput., vol. EC-10, pp. 691–698, Dec. 1962.
- [6] R. A. Haring, M.S. Milshtein, T.I. Chappell, S. H. Dong and B.A. Chapell, "Self resetting logic and incrementer" in Proc. IEEE Int. Symp. VLSI Circuits, 1996 pp. 18-19.
- [7] G. Yee and C. Sechen, " Clock-delayed domino for adder and combinational logic design" in proc.IEEE/ACM Int. Conf. Computer Design, Oct., 1996, pp. 332-337.
- [8] P. Ng, P. T. Balsara, and D. Steiss, —Performance of CMOS Differential Circuits,| IEEE J. of Solid-State Circuits, vol. 31, no. 6, pp. 841-846, June 1996.
- [9] P. Srivastava, A. Pua, and L. Welch, .Issues in the Design of Domino Logic Circuits, Proceedings of the IEEE Great Lakes Symposium on VLSI, pp. 108-112, February 1998.
- [10] W. Zhao and Y. Cao. —New generation of predictive technology model for sub-45nm design exploration,| In IEEE Intl. Symp. On Quality Electronics Design, 2006
- [11] CMOS Logic Circuit Design 2002 John P Uyemura



Vijaya Bhaskar Madivada was born in A.P,India. He received the **B.Tech** degree in **Electronics & communications Engineering** from Jawaharlal Nehru Technological University in 2010. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include FPGA Implementation, Low Power Design.



S.Suparshya Babu was born in India, A.P. He received the B.Tech degree from JNTU, A.P, and M.Tech degree from SRM University, Chennai, Tamil Nadu, India in 2008 and 2010 respectively. He worked as Assistant Professor in Electronics Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in K L University. He is a member of Indian Society for Technical Education and International Association of Engineers. His research interests include antennas, FPGA Implementation, Low Power Design and wireless communications and Robotics.



S.Susrutha Babu was born in India, A.P. He received the B.Tech degree from JNTU, A.P, and M.Tech degree from SRM University, Chennai, Tamil Nadu, India in 2008 and 2010 respectively. He worked as Assistant Professor in Electronics Engineering in Bapatla Engineering College for academic year 2010-2011 and from 2011 to till date working in K L University. He is a member of Indian Society for Technical Education and International Association of Engineers. His research interests include antennas, FPGA Implementation, Low Power Design and wireless communications and Digital VLSI.



G Santhi Swaroop Vemana was born in A.P,India. He received the B.Tech degree in Electronics & communications Engineering from Jawaharlal Nehru Technological University in 2008. He worked as a OFC engineer at United Telecoms ltd at GOA during 2009-2010. Presently he is pursuing M.Tech VLSI Design in KL University.