I.J. Intelligent Systems and Applications, **2023, 3, 33-43**
Published Online on June 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2023.03.03

Static Timing Analysis of Different SRAM Controllers

Jabin Sultana
Electronics and Communication Engineering Discipline, Khulna University, Khulna 9208 Bangladesh
E-mail: jabinsultana54@gmail.com

S. M. Shamsul Alam*
Electronics and Communication Engineering Discipline, Khulna University, Khulna 9208 Bangladesh
E-mail: alam_ece@yahoo.com
ORCID iD: https://orcid.org/0000-0003-3376-5950
*Corresponding author

Received: 03 October 2022; Revised: 04 January 2023; Accepted: 27 January 2023; Published: 08 June 2023

Abstract: Timing-critical path analysis is one of the most significant terms for the VLSI designer. For the formal verification of any kinds of digital chip, static timing analysis (STA) plays a vital role to check the potentiality and viability of the design procedures. This indicates the timing status between setup and holding times required with respect to the active edge of the clock. STA can also be used to identify time sensitive paths, simulate path delays, and assess Register transfer level (RTL) dependability. Four types of Static Random Access Memory (SRAM) controllers in this paper are used to handle with the complexities of digital circuit timing analysis at the logic level. Different STA parameters such as slack, clock skew, data latency, and multiple clock frequencies are investigated here in their node-to-node path analysis for diverse SRAM controllers. Using phase lock loop (ALTPLL), single clock and dual clock are used to get the response of these controllers. For four SRAM controllers, the timing analysis shows that no data violation exists for single and dual clock with 50 MHz and 100 MHz frequencies. Result also shows that the slack for 100MHz is greater than that of 50MHz. Moreover, the clock skew value in our proposed design is lower than in the other three controllers because number of paths, number of states are reduced, and the slack value is higher than in 1st and 2nd controllers. In timing path analysis, slack time determines that the design is working at the desired frequency. Although 100MHz is faster than 50MHz, our proposed SRAM controller meets the timing requirements for 100MHz including the reduction of node to node data delay. Due to this reason, the proposed controller performs well compared to others in terms slack and clock skew.

Index Terms: Slack, Static Timing Analysis (STA), Clock Skew, Register Transfer Level (RTL), SRAM Controllers.

1. Introduction

Timing analysis of the digital circuits is required to see if the timing constraints given for every connection are satisfied. This usually signifies that demonstrate all set-up, hold, and pulse-width times have been met. The most important circuits in the digital circuits are sequential circuits. No automation is possible without sequential circuits. Flip flops are the main building components of such circuits, and the role of the clock is crucial [1]. The connection between the clock and the data might make the entire circuit more or fewer efficient. As a result, data timing in relation to time must be accurate and interchangeable, or the circuit will enter short-steady condition and produce incorrect output. When creating sequential circuits, we encounter procedures such as logic synthesis, floor planning, placement, routing, and layout. But every step by step is to check the static timing analysis and so that the design works properly.

The primary goal of static timing analysis is to ensure that, despite these probable fluctuations, all signals arrive to ensure proper circuit operation by not starting too early or too late. Static timing analysis (STA) can detect other issues like as defects, sluggish routes, and clock skew because it can validate every path. Dynamic timing analysis evaluates performance of the system by applying input parameters and testing for correct output, whereas static timing analysis examines static timing constraints of the circuit without any input or output parameters. It calculates the duration of time because it is necessary for a signal to propagate along each path. Synchronous sequential circuits require exact timing parameters, if it fails to meet these requirements the chip leads to run at a lower frequency and ultimately fails. Setup time, hold time, clock skew, data delay and a few additional characteristics are examples of static timing

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

parameters or specifications.

P.P. Chu explains how the ASMD chart was used to describe the development of an SRAM controller. There are a safe controller design and other three types of SRAM controller design mentioned in this book [2]. J. Sultana et al. analyzed three controllers, proposed their design, and also calculated power consumption and resource utilization in terms of number of registers, maximum fanout, total number of logic elements [3]. According to other designs, proposed design consumed less power and utilized less amount of logic elements. In static timing analysis, a methodology is presented to exploit the interdependence between setup-time and hold-time limitations.

There are two stages to the approach. The interdependent characterization of consecutive cells occurs in the first phase, resulting in many constraint pairings. In STA, the second phase comprises an efficient algorithm that takes advantage of these many pairs. The technique enhances accuracy by removing superfluous pessimism and decreasing optimism. Furthermore, in STA, the tradeoff between setup and hold times is used to reduce timing violations dramatically. These advantages are demonstrated using industrial circuits and tools, which show a reduction of up to 53% in the number of constraint breaches and a reduction of up to 48% in the worst negative slack, corresponding to a 15% reduction in the clock period is shown in this paper [4]. R. Abdollahi et al. describe a fully digital approach for detecting and correcting setup/hold time violations in digital circuits. Simple, low-power, small-area, and easy-to-modify design are all advantages of the suggested system. The system is made up of two distinct blocks: a detector and a corrector, which are coupled by a 2-bit control signal. The detector gives control signals to the corrector and is situated near flip-flops or memory elements. Only CMOS inverters and a multiplexer make up the corrector circuitry, which is placed after the global clock source to correct the clock worldwide or before the elements to correct them locally. This technique can be utilized frequently across the chip because of its simplicity. Up to the highest feasible working frequency a given process allows, the system constantly detects and corrects any setup/hold time breaches [5]. Since the analysis of the design is done statically and is independent of the data values applied at the input pins, the STA is static [6-10]. Otherwise, static timing analysis offers a quicker and easier method of examining all the timing channels in a system for any timing violations. Low power consumption has become increasingly important in recent decades as the market for portable electronic gadgets has grown. Asynchronous circuits, in general, offer low power consumption qualities due to dynamic power scaling and the lack of a global clock distribution. As a result, asynchronous circuit design is becoming increasingly widespread, and the design of asynchronous SRAM utilized in FPGA devices is likewise becoming more complex. A synchronous SRAM controller logic converter interface has been used in certain implementations to simulate asynchronous SRAM [11]. L. Li et al. present that when optimizing global and local pathways, designers need accurate clock skew budgets to avoid hold-time failures and effectively allocate resources. Voltage jitter and process variation, which are becoming important elements in otherwise balanced -trees, are often overlooked in published clock skew budgets. Worst-case process variation assumptions, on the other hand, are extremely pessimistic. Using a modified -tree model, this study describes the key sources of clock skew in a microprocessor and applies the model to a second-generation Itanium-M processor family microprocessor that is currently under development. In a four-level skew hierarchy, Monte Carlo simulation is utilized to produce statistical clock skew budgets for setup and hold time limitations. The majority of skew budgets are accounted for via voltage jitter through the phase locked loop (PLL) and clock buffers [12]. The DE2-115 board has an SRAM of ISSIIS61WV102416BLL model that operates a maximum performance frequency of about 125MHz [13]. To confirm that the data has been collected accurately, all successive items require a minimum pulse width with high or low [14]. If the clock used to feed a sequential object has a pulse width that is smaller than the minimum requirements, then the circuit may capture the correct data, and the FSM will work properly or the flip may totally miss the clock pulse, resulting in no new data being captured. As a result, the FSM will result in an invalid state. Clock skew occurs when the clock signal (sent from the clock circuit, source, or clock definition point) arrives at separate components at different times in synchronous circuits [15].

The main purpose of this research paper is to study static timing analysis of multiple SRAM controllers which has different parameters such as setup time, hold time, clock skew, data delay, maximum clock frequency, metastability, etc. for proposed SRAM controller. The performance of these SRAM controllers in terms of resource utilization had been discussed in authors' another paper [3]. In this paper, we have performed the STA of these SRAM controllers including the reduction of the clock skew techniques. There are first 3 controllers mentioned in [2]. We proposed our 4th SRAM controllers that is better than others compatibility few factors such as static power consumption, dynamic power consumption, resource utilization, timing for 100MHz, reduce clock skew using two methods. The main limitation of our design is frequency. According to Altera DE2 manual, this SRAM has maximum frequency of about 125MB. For safety purpose, we use 50 MHz and 100 MHz frequencies. In this paper, authors' hope to achieve that for both frequencies, the proposed controller is the best for various timing issues such as setup time, hold time, clock skew, data delay, maximum clock frequency, metastability, etc. No data violation creates in our design with node to node timing requirements.

This paper's overview is as follows. Section 2 covers four different types of SRAM controllers, as well as static timing parameters and clock skew reduction techniques. The simulation results are provided in Section 3, and the conclusions are described in Section 4.

2. Material Methods

Direct access to SRAM by the synchronous system is challenging because SRAM is asynchronous and the main system is synchronous [2]. The signals must be controlled by a particular circuit referred to as an SRAM controller. SRAM controller behavior serves as an interface between the external SRAM and the main system. The SRAM controller receives input from the FPGA board and generates specific logical timing signals to communicate with the SRAM.

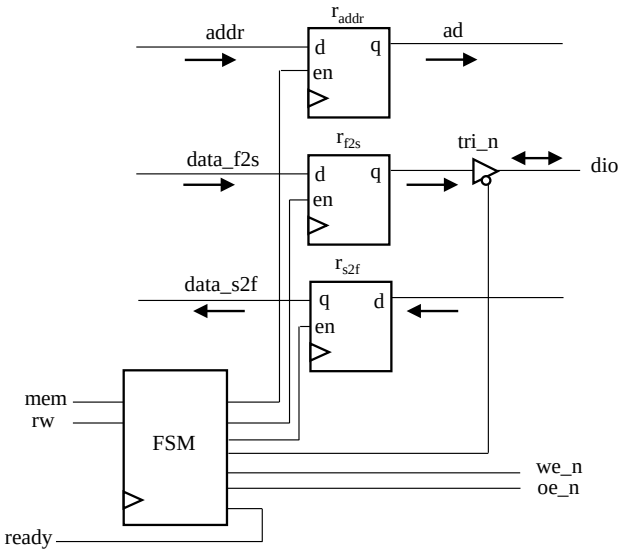

Fig.1. Block diagram of SRAM controller [2,3]

Table 1. Parameters of the SRAM controllers [3]

addr	20-bit address signal
data_f2s	Exchange of 16-bit data from FPGA to SRAM.
data_s2f	Exchange of 16-bit data from SRAM to FPGA.
mem	It is asserted when a memory action is required.
rw	It's where the operation type is defined (write or read).
ready	When a subsequent command is given, a status signal is sent.
ce_n	Enables or disables the chip.
we_n	Activate or deactivate the write operation.
oe_n	Activate or deactivate the read operation.
dio	Bi-directional data of SRAM.
ad	20-bit address of SRAM.
tri_n	To control the data flow

The controller covers up the main system's detailed timing and makes memory access appear to be synchronous. The efficiency of a memory controller is evaluated by the number of memory accesses that can be finished in a specific time frame. While it is straightforward to design a simple memory controller, achieving maximum performance is difficult due to many timing issues. In Fig. 1, the left-side parameters are the controller side and the right-side parameters are the memory side. The write data is saved in data_f2s_reg, whereas the data that was read is retained in data_s2f_reg. The address is saved in addr_reg. The timing properties of an asynchronous SRAM are highly complicated, involving over 2 dozen parameters.

2.1. Four Design of SRAM Controllers

A. 1st SRAM Controller

Fig. 2 shows the ASMD chart of the 1st SRAM controller, which has 5 phases in total, the first of which is *idle*. In the beginning the *mem* signal is examined, then the *rw* signal to identify that whether function is write or read. If *rw* is set to high, the FSM occupies the *r1* state for read operations and the *w1* state for write operations. The memory address is checked and preserved in the address register during the transition (*raddr* in Fig. 1). The *oe_n* signal is present both in

r1 and *r2* states. At the state, the obtained data is kept in the *data_s2f* register. Perhaps of returning to *idle*, it verifies conditional boxes before deciding where to go next. The controller starts a new memory operation if there is an existing request. The conditional boxes are evaluated in the *r2* and *w2* states, and the FSMD can proceed directly to the *r1* or *w1* state if additional memory operation is required.

On this controller, back-to-back procedures need 2 clock cycles to finish. The (*dio*) port is controlled by a tristate buffer in the *w2* state, which involves a slight delay to manage the transfer of data between the FPGA and the SRAM [2].

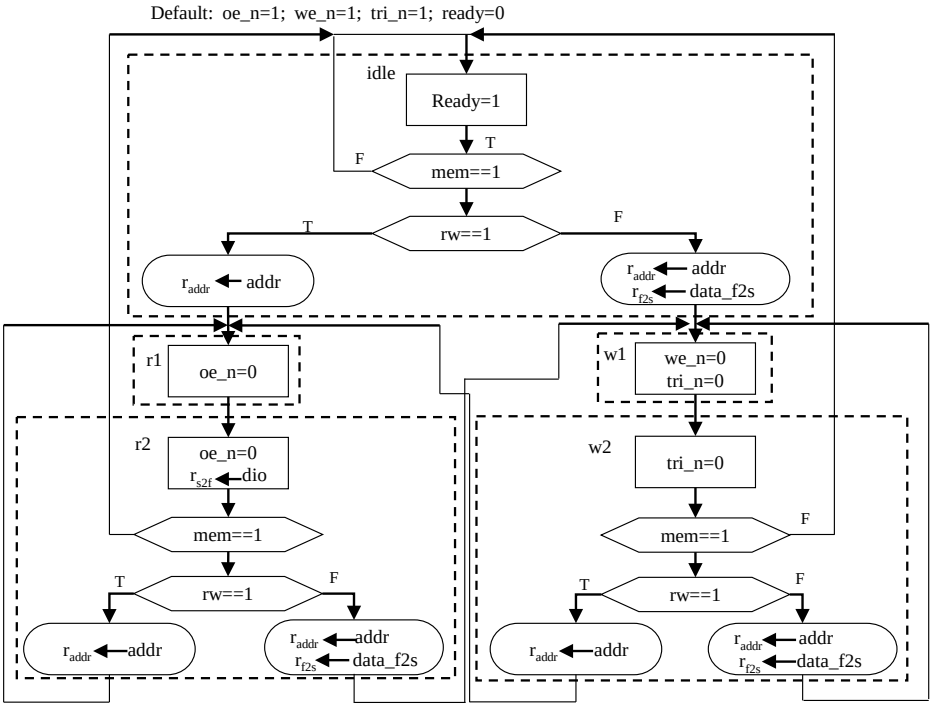

Fig.2. ASMD chart of 1st SRAM controller [2,3]

The SRAM controller produces signals that deactivate the SRAM's tristate buffer while also activate the FPGA's. The *r2* state changes to the *w1* state for write operations at this time, according to the ASMD chart. However, if the SRAM tristate buffer's transmission delay is too quick or too slow, it will cause a difficulty with memory access. Fighting occurs whenever both buffers do not sustain a small delay to the data on the bus, resulting in considerable device damage and a large transient current, reducing the 1st SRAM controller's stability [2]. To avoid the issue, the 2nd SRAM controller features a limited number of states and logic parts.

B. 2nd SRAM Controller

Fig. 3 shows the 2nd SRAM controller architecture that avoids conflict concerns by omitting the *r2* and *w2* states from the ASMD chart.

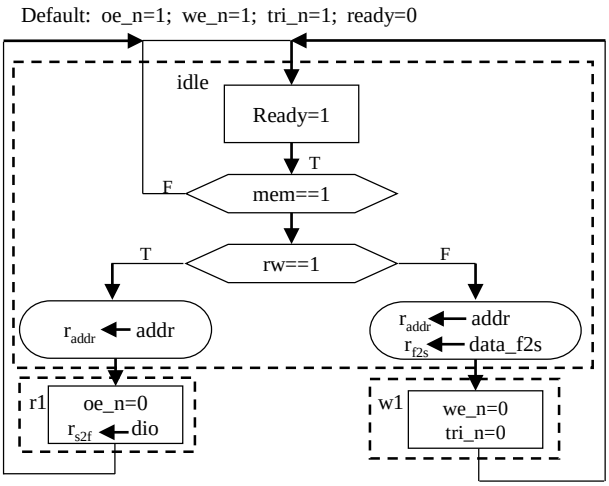

Fig.3. ASMD chart of 2nd SRAM controller [2,3]

The controller contains one clock cycle to perform a memory access at 50MHz and 100MHz, and 2 clock cycles (i.e., 20ns for 100MHz frequency) to complete a sequential procedure.

Rather of verifying the next operation, it comes back to the *idle* state after finishing the read or write action. As a result, each new address starts off in the *idle* state. This technique sets substantially tighter timing limitations for both operations by removing the two states [2]. If a signal flows through the FGPA's I/O pads, it is subjected to further latency. Pad delay is a time delay that varies per device and is often much longer than internal time. The *we_n* signal is silenced first in order to properly latch the data to the SRAM. As if the *we_n* and *tri_n* signals are deactivated concurrently at the end of the *w1* state, the controller goes back to idle state.

C. 3rd SRAM Controller

The 3rd SRAM controller concept is included in Fig. 4. One clock cycle is needed for memory access, and one clock cycle is required for consecutive operations.

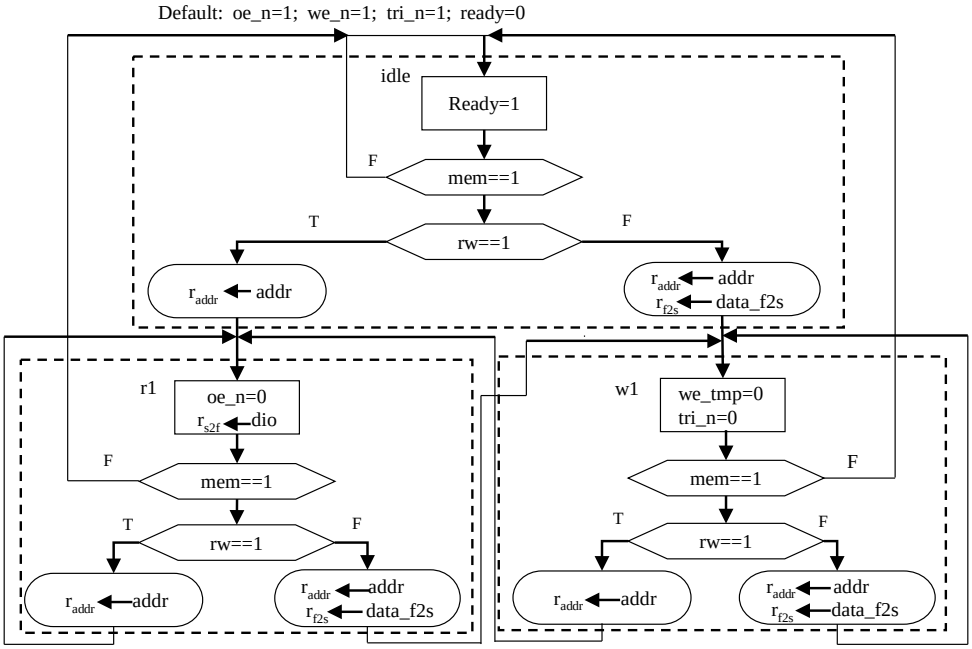

Fig.4. ASMD chart of 3rd SRAM controller [2,3]

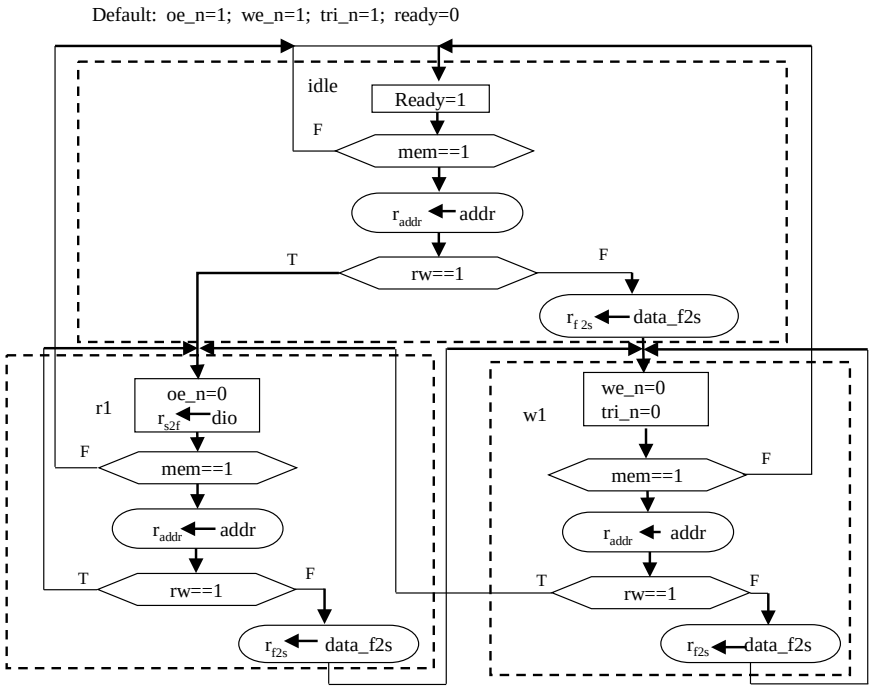

Fig.5. ASMD chart of 4th SRAM controller [3]

The formation of the *we_n* from *we_tmp* is a new difficulty in this case. Because the *we_n* signal needs to last only a fraction of a clock cycle, it is invisible on the chart in Fig. 4. Alternatively, in the *w1* state, use the *we_tmp* signal to obtain *we_n* [2]. This controller performs write operations swiftly from previous two controllers by utilizing half clock cycle. However, ***WE Pulse Width*** (t_{PWE1} = 6ns for 100MHz & 8ns for 50MHz) is not satisfied according to our device requirements mentioned in SRAM datasheet [6].

D. 4th SRAM Controller

As illustrated in Fig. 5, the proposed design is the 4th SRAM Controller. Without the exception of returning to the *idle* state and 1 clock cycle for memory access, the aim of this controller is to meet the timing requirements of the ***WE Pulse Width*** and to take 1 clock cycle sequential procedures. The previously mentioned timing issues are also considered.

2.2. Clock Implementation

By using Quartus prime tools, Phase Lock Loop (PLL) is instantiated in all controllers to get stable clock frequency. Separately (e.g., 50MHz, 100MHz) frequencies are firstly checked for all controllers. The node-to-node timing setup and hold summary are mentioned in Table 2. The PLL is considered to be phase-locked when the f_{REF} signal and the Feedback signal have the same phase and frequency. When the M counter is placed in the feedback path, the VCO oscillates at a frequency M times that of the f_{REF} signal.

The input clock (f_{IN}) divided by the pre-scale counter yields the f_{REF} signal (N). The reference frequency is described by the equation (1).

$$f_{REF} = \frac{f_{IN}}{N} \tag{1}$$

The VCO output frequency is explained in equation (2) and the output frequency of the PLL is described by the equation (3) for getting the desired clock signals.

$$f_{VCO} = f_{IN} \times \frac{M}{N} \tag{2}$$

$$f_{OUT} = f_{OUT} = \left(f_{IN} \times M\right)/\left(N \times K\right) \tag{3}$$

For dual clock implementation procedure, in which we instantiate two PLLs that can take two different frequencies, as illustrated in the PLL block diagram shown in Fig. 6 [7]. For the dual clock implementation, we can use a 50MHz clock frequency for write operations and a 100MHz clock frequency for read operations. When implementing 100MHz, read operations are faster than write operations. As a result, write cycle takes 20ns for one write operation whereas read cycle takes only 10ns for one read operation using 100MHz. The RTL view of dual clock implementation is shown in Fig. 7. The 50MHz frequency (*clk1*) is connected to *data_f2s_reg* and the 100MHz frequency (*clk2*) is connected to *data_s2f_reg* illustrated in Fig. 7. After the synthesis, the RTL view is shown in Fig. 7. The *sram_ctrl_unit* is our main memory controller. Using pll_50 for 50MHz frequency is passed by *clk_w* and *pll_100* for 100 frequency is passed by *clk_100*. Seven segments are used to display our write and read data. Write data is passed through *data_f2s* path and after read data is passed through *data_s2f_r* path.

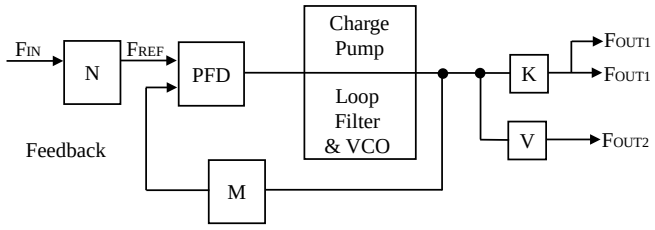

Fig.6. Block diagram of ALTPLL [2]

A. Clock Skew Reduction Method

Clock skew is the difference in clock signal arrival times throughout the chip. Timing must satisfy register setup and hold time requirements, which is a fundamental design principle. These estimates include both data propagation delay and clock skew. Timing violations or even functional failures can occur when successively neighboring registers are clocked on the same edge of a high-skew clock. In this paper, two clock skew reduction methods are applied such as alternate edge clocking and alternate phase clocking for reducing clock skew.

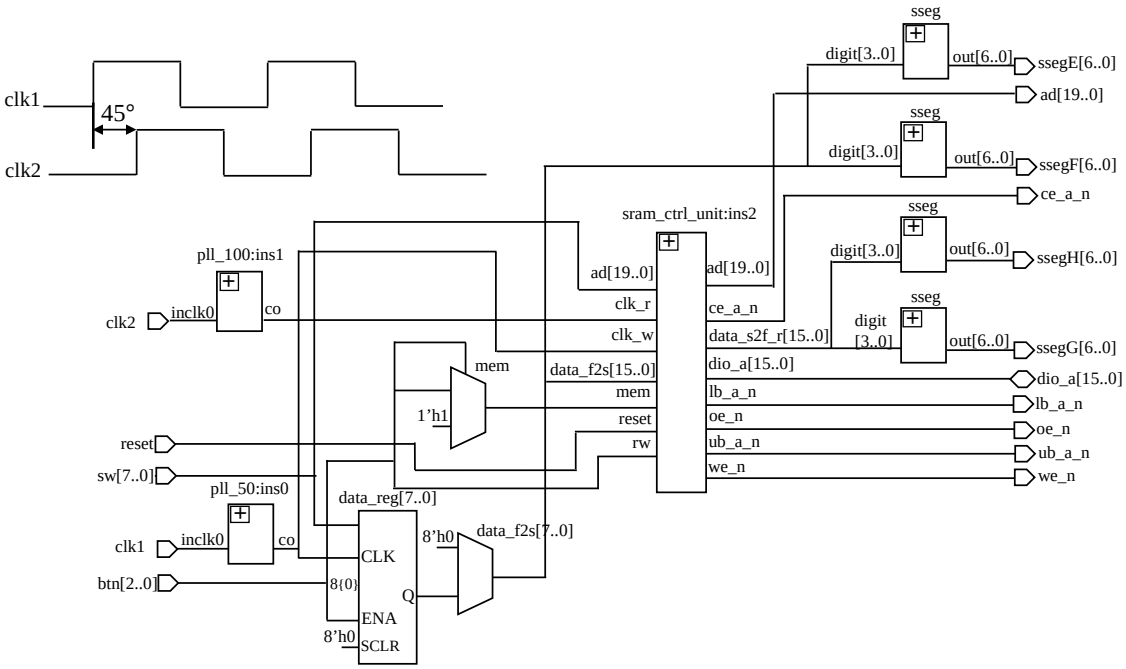

Fig.7. RTL view of dual clock implementation

For alternate edge clocking, positive edge triggering (clk1) is connected to data_f2s_reg and the one-half clock cycle path (*clk2*) is connected to *data_s2f_reg* illustrated in Fig. 7.

For alternate phase clocking, the 50MHz frequency (*clk1*) is connected to *data_f2s_reg* and the 45° phase shift & 90° phase shift of 50MHz frequency (*clk2*) is connected to *data_s2f_reg* illustrated in Fig. 7.

3. Result and Discussion

3.1. Performance Analysis of Static Timing Analysis

Various types of static timing analysis parameters are shown below for four types of SRAM controllers. Intel Quartus Prime Timing Analyzer demonstrates how to specify timing constraints and perform static timing analysis. The Timing Analyzer validates the timing performance of all logic in our design using industry-standard constraint, analysis, and reporting methodology. This work has been simulated on the Altera DE2-115 FPGA board with its SRAM of *ISSIIS61WV102416BLL* model.

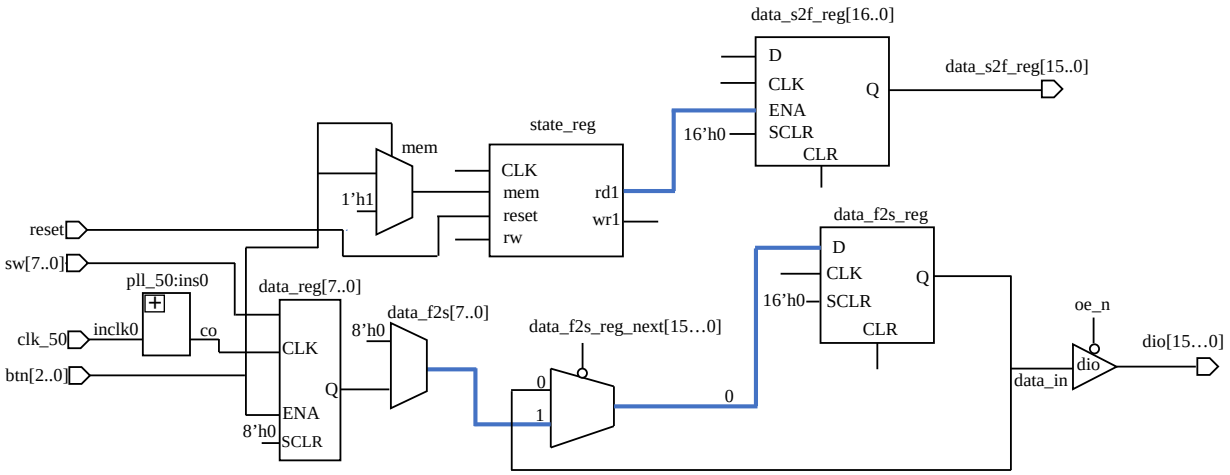

Fig.8. RTL view of 4th SRAM Controller for state_reg: rd1 to data_s2f_reg[7] and data_reg[5] to data_f2s_reg[5]

A. Single Clock Implementation

We are presenting node to node static timing information for both *state_reg.rd1* to *data_s2f_reg[7]* and *data_reg[5]* to *data_f2s_reg[5]* shown in Fig. 8.

Setup Summary for Single Clock Implementation: Different types of timing paths have mentioned in Table 2. The path *state_reg.rd1* to *data_s2f_reg[7]* is as shown in Fig. 9. Here the start point is a *state_reg.rd1* input port and the end point is a *data_s2f_reg[7]* output port. Launch edge is the edge where the data from the source register *state_reg.rd1* is launched and latch edge is the edge at the destination register *data_s2f_reg[7]* that latches the data with respect to the launch edge, selected by the timing analyzer, typically 1 clock cycle, (i.e., rising to rising edge). Therefore, the setup relationship or time is 20ns for 50MHz. The generated value by the timing analyzer of Quartus Prime tool is illustrated in Fig. 9. Setup slack ($t_{setupslack}$) is the difference between the data arrival time (t_{da}) and the data required time (t_{dr}).

$$t_{setupslack} = t_{da} - t_{dr} \tag{4}$$

Setup slack must be positive otherwise data violation creates and circuit does not work properly. Positive setup slack suggests that the design is on schedule, although it can be improved. Zero setup slack shows that the design is operating at the specific frequency.

Negative setup slack indicates that the design does not match the specified timings or frequency requirements. According to equation (4), setup slack is positive and no data violation creates. It is satisfied for 50MHz illustrated in Fig. 9. For 100MHz, setup slack & hold slack is positive that meets all controllers are on schedule and no violation exits.

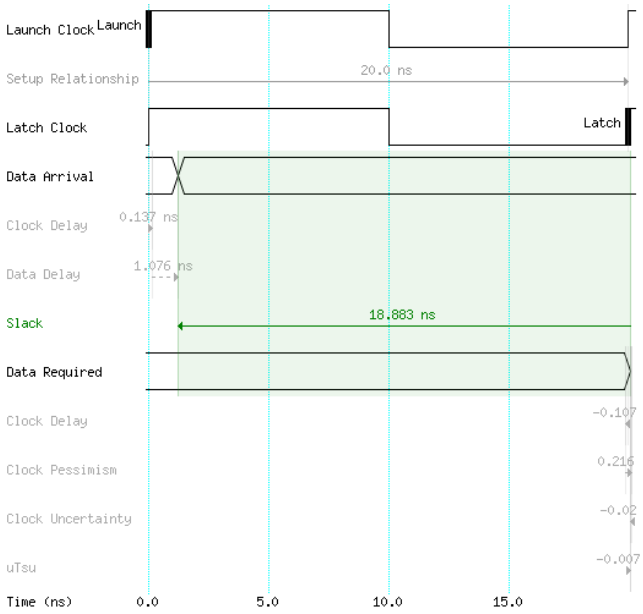

Fig.9. Simplified two nodes (state_reg.rd1 to data_s2f_reg[7]) of setup summary for 4th SRAM controller

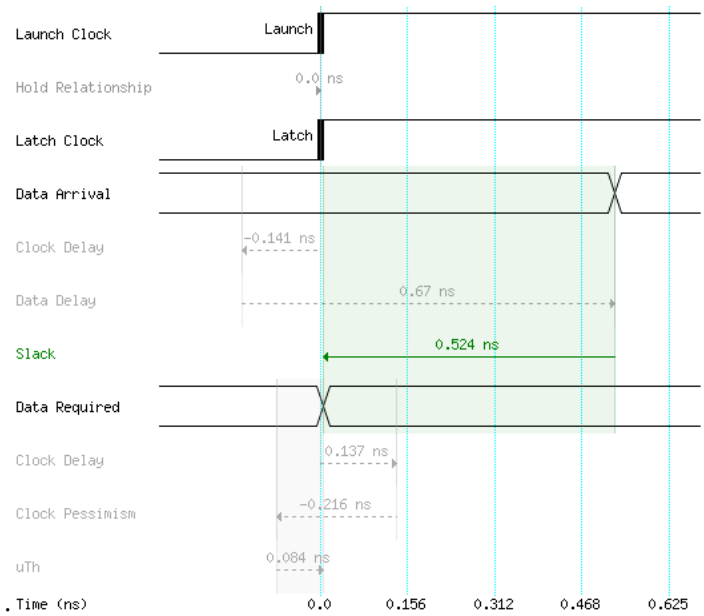

Fig.10. Simplified two nodes (data_reg[5] to data_f2s_reg[5]) of hold summary for 4th SRAM controller

Hold Summary for Single Clock Implementation: The path *data_reg[5]* to *data_f2s_reg[5]* is shown in Fig. 10. Here, the start point is a *data_reg[5]* input port and the end point is a *data_f2s_reg[5]* output port. Clock latency or clock delay is -0.141ns because the time it is necessary for the clock signal to travel between two places (the source (PLL) to the source pin (Clock Pin) of registers). Hold time is the lowest time the data signal must retain stable after the clock edge [4]. Hold slack ($t_{holdslack}$) is the difference between the data arrival time (t_{da}) and the data required time (t_{dr}). According to equation (5), hold slack is positive and meets the timing requirement. In equations (4) and (5), each path is satisfied and meets the timing requirement. In every node-to-node timing path, slack is positive and no violation creates for every controller. Even for 100MHz, slack is positive and no violation creates for every controller. But clock skew is negative for setup time because of the clock source delay or clock latency. Positive clock skew is useful for resolving setup issues, but they can also create the problem for hold issues. Negative clock skew can prevent a hold violation, but it can also result in a setup violation. Yet negative clock skew is not more serious for setup issues. By increasing the frequency, the clock skew is reduced a little less.

$$t_{holdslack} = t_{da} - t_{dr} \tag{5}$$

For each controller, the data delay is nearly the same for both clock frequencies. The slack for 100MHz is greater than to that of 50MHz. The clock skew value in our proposed design is lower than in the other three controllers because number of paths, number of states are reduced, and the slack value is higher than in 1st and 2nd controllers. Internal delay is reduced since 4th controller has fewer temporal routes than the other three controllers.

Table 2. Single Clock Implementation of Four SRAM Controllers

From node	To Node	50MHz						100MHz					
		Setup summary			Hold Summary			Setup summary			Hold Summary		
		Slack	Clock Skew	Data Delay	Slack	Clock Skew	Data Delay	Slack	Clock Skew	Data Delay	Slack	Clock Skew	Data Delay
1st SRAM Controller													
data_reg[0]	data_f2s_reg[0]	19.292	-0.038	0.657	0.450	0.043	0.577	9.027	-0.077	0.883	0.654	0.004	0.742
state_reg.rd2	data_s2f_reg[0]	18.510	-0.007	1.470	1.194	0.076	1.35	8.598	-.017	1.406	1.117	0.098	1.299
2nd SRAM Controller													
data_reg[0]	data_f2s_reg[0]	19.463	-0.042	0.482	0.289	0.042	0.415	9.457	-0.040	0.490	0.263	0.044	0.411
state_reg.rd1	data_s2f_reg[0]	18.706	-0.010	1.271	1.032	0.074	1.190	8.961	-0.035	0.991	0.813	0.049	0.946
3rd SRAM Controller													
data_reg[0]	data_f2s_reg[0]	19.226	-0.028	0.733	0.497	0.055	0.613	9.223	-0.028	0.736	0.495	0.055	0.634
state_reg.rd1	data_s2f_reg[0]	18.706	-0.010	1.271	1.032	0.074	1.146	9.261	-0.042	0.684	0.557	0.042	0.683
4th SRAM Controller													
data_reg[0]	data_f2s_reg[0]	19.146	-0.021	1.076	0.575	0.062	0.721	9.134	-0.021	0.832	0.577	0.062	0.723
state_reg.rd1	data_s2f_reg[0]	18.883	-0.028	0.793	0.882	0.056	1.022	8.883	-0.028	1.076	0.882	0.056	1.022

Metastability is an unstable equilibrium phenomenon in which the sequential element is unable to resolve the state of the input signal, causing the output to remain unresolved for an infinite period of time [8]. No metastability is found in all controllers.

B. Dual Clock Implementation

For static timing analysis, we mainly change the read state where *data_s2f_reg, oe_reg* are changed accordingly to 100MHz. By using both frequencies together, the slack value is positive for each controller and no violation creates mentioned in Table 3. But the clock skew is increased because of the clock source latency. The *data_s2f_reg* register receives the clock tick before the transmitting *data_f2s_reg* register, this occurs for different frequencies. For each controller, the data delay is nearly the same for all controllers. In comparison, the values of slack, clock skew, and data delay for dual clock execution reported in Table 3, the 3rd controller is better than 1st, 2nd, and 4th controller. The comparison between single clock and dual implementation is shown that total power consumption is reduced for single clock (170.29mW) and is increased for dual clock (188.90mW) mentioned in [3]. After increasing the clock frequency to 100 MHz, the value of Setup slack is better than 50MHz without any violations. For single clock implementation Hold slack is 0.882ns that is reduced almost half 0.430ns for dual clock implementation. Data delay is also important part for analysis that is increased for dual clock because both clock frequencies are not same.

One clock frequency is faster than other. But Slack is always positive and no violation creates. No metastability is found in all controllers.

Table 3. Dual Clock Implementation of Four SRAM Controllers

From node	To node	Setup summary			Hold Summary		
		Slack	Clock skew	Data Delay	Slack	Clock skew	Data Delay
1st SRAM Controller							
state_reg.rd1	state_reg.oe_reg	8.678	-0.285	0.894	0.258	0.230	0.722
state_reg.rd2	data_s2f_reg[0]	8.352	-0.268	1.237	0.676	0.247	1.157
2nd SRAM Controller							
state_reg.idle	state_reg.oe_reg	8.774	-0.285	0.798	0.227	0.229	0.690
state_reg.rd1	data_s2f_reg[0]	8.583	-0.280	0.994	0.477	0.235	0.946
3rd SRAM Controller							
state_reg.rd1	data_s2f_reg[0]	8.907	-0.285	0.665	0.205	0.230	0.669
4th SRAM Controller							
state_reg.rd1	data_s2f_reg[0]	8.641	-0.275	0.941	0.430	0.240	0.904

3.2. Summary Result of Alternate Edge Clocking

Attempting to eradicate clock skew for setup summary after evaluating single clock and dual clock execution. To do alternate edge clocking, we sequentially clock subsequent flops on the clock's opposite edges. For reducing clock skew, this approach provides a one-half clock cycle path-clock skew margin [9]. This technique successfully works for all controllers such as seen in the following Table 4. For alternate edge clocking, clock skew is reduced -0.015ns, -0.016ns, -0.015ns and - 0.002ns from -0.041ns, -0.042ns, -0.041ns and -0.028ns according to 1st, 2nd, 3rd and 4th SRAM controllers. Slack, on the other hand, is half to account for the clock's opposing edges. Otherwise, for all controllers, slack values are positive, and no violation exists. Data delay is not much changed for using this technique.

Table 4. Setup Summary for Four SRAM Controllers for Alternate Edge Clocking

From node	To node	50MHz				Alternate Edge Clocking			
		Design-1	Design-2	Design-3	Design-4	Design-1	Design-2	Design-3	Design-4
data_reg[0]	data_f2s_reg[0]	-0.038	-0.042	-0.042	-0.028	-0.024	-0.026	-0.027	-0.026
state_reg.wr1	data_f2s_reg[0]	-0.041				-0.027			
state_reg.idle	data_f2s_reg[0]		-0.040				-0.027		

3.3. Summary Result of Alternate Phase Clocking

Alternate phase clocking is one of the well-known methods for avoiding clock skew [9]. One clock is 50MHz and the other is 45° of 50MHz or 90° of 50MHz. After using alternate phase clocking, the clock skew is not reduced. Because the receiving register receives the clock tick before the transmitting register. Again, the sending register receives the clock tick before the receiving register. As a result, register to register delay is increased. Although the 45° phase shift is changed -0.045ns to -0.094ns increased the value of clock skew for 2nd controller mentioned in Table 5. Again, for the 90° phase shift is changed -0.042 to -0.092 increased the value of clock skew for 2nd controller. 2nd controller is increased double value of clock skew. Other 3 controllers are also increased the value of clock skew. As a result, this technique is not applicable for all SRAM controllers.

Table 5. The value of clock skew for Setup Summary for Four SRAM Controllers for Alternate Edge Clocking

From node	To node	50MHz				Alternate phase clocking							
						45° phase shift				90° phase shift			
		Design-1	Design-2	Design-3	Design-4	Design-1	Design-2	Design-3	Design-4	Design-1	Design-2	Design-3	Design-4
data_reg[0]	data_f2s_reg[0]	-0.038	-0.042	-0.021	-0.028	-0.038	-0.094	-0.035	-0.026	-0.074	-0.092	-0.091	-0.035
state_reg.wr1	data_f2s_reg[0]	-0.041				-0.041				-0.099			
state_reg.idle	data_f2s_reg[0]		-0.040				-0.093				-0.090		

The comparison between alternate edge clocking and alternate phase clocking is reported that clock skew is reduced for alternate edge clocking but alternate phase clocking is not helpful for these controllers. However, clock skew is high for alternate phase clocking. But Slack is always positive and no violation creates. No metastability is found in all controllers.

4. Conclusions

This paper provides a basic overview of static timing analysis for the SRAM controllers. Four types of SRAM Controllers are discussed including their design and implementation on the Altera DE2 board. By using Quartus Prime

tools, their resource utilization and power consumption are discussed in [3]. We utilize multiple clock frequencies and again separate frequencies to check all controllers for timing stability. Static timing analysis parameters (such as setup & hold timing, setup & hold violation, slack, clock skew, etc) are checked for all controllers.

We check the value of slack for every controller so that any violation is not created for data setup and hold time through timing analyzer tools. With that clock skew and data delay are described for every design. We implement the clock skew reduction method to reduce clock skew. Alternate edge clocking and alternate phase clocking are implemented four SRAM controllers and compared them. Alternate edge clocking is applicable for four SRAM controllers. No metastability found four SRAM controllers.

References

[1] A. Kumar, S. L. Tripathi, S. Dhariwal, "Static timing analysis of sequential circuit with GUI", *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, IEEE, 2020, pp. 312-315

[2] P. P. Chu, FPGA prototyping by Verilog examples: Xilinx Spartan-3 version, John Wiley & Sons, 2011.

[3] J. Sultana, S. M. S. Alam, "Performance analysis and implementation of sram controller on altera de2 board", *In: 2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, 2021, pp. 1–4. doi:10.1109/ICECIT54077.2021.9641430.

[4] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, E. G. Friedman, "Exploiting setup–hold-time interdependence in static timing analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* vol.26 (6) (2007) pp. 1114–1125.

[5] R. Abdollahi, K. Hadidi, A. Khoei, A simple and reliable system to detect and correct setup/hold time violations in digital circuits, *IEEE Transactions on Circuits and Systems I*: Regular Papers vol. 63 (10) (2016) pp.1682–1689.

[6] IS65WV102416BLL SRAM datasheet pdf, www.issi.com, accessed: 2022-5-16.

[7] ALTPLL (phase-locked loop) IP core user guide, https://www.intel.com/programmable/technical-pdfs/683732.pdf

[8] M. Thakur, B. B. Soni, P. Gaur, P. Yadav, Analysis of metastability performance in digital circuits on flip-flop, *In 2014 International Conference on Communication and Network Technologies*, IEEE, 2014, pp. 265–269.

[9] Chapter2 clocks resets-04, https://rb.gy/hvnryq.

[10] J. Bhasker, Rakesh Chadha**, "**Static Timing Analysis for Nanometer Designs a Practical Approach" Springer, 2009th edition, 2011.

[11] C. XIANG, "Design and implementation of Asynchronous SRAM," 2009.

[12] Lei Li, Jianhao Hu, Chun He and Wanting Zhou, "Statistical clock skew modeling and analysis for resonant clock distribution networks," International Conference on Communications, Circuits and Systems 2009, pp. 1024-1028, doi: 10.1109/ICCCAS.2009.5250335, 2009.

[13] Altera DE2-115 User Manual, available on https://www.intel.com. [Accessed: 12-Mar-2022].

[14] "Minimum pulse width," Blogspot.com. [Online]. Available: https://vlsiuniverse.blogspot.com/2016/09/min-pulse-width-check.html. [Accessed: 07-Mar-2021].

[15] J. Mistry, "VLSI Basic," Blogspot.com. [Online]. Available: https://vlsibasic.blogspot.com/2014/10/clock-skew.html. [Accessed: 07-Mar-2022].

Authors' Profiles

Jabin Sultana received her B.Sc(Engg). in Electronics & Communication Engineering from Khulna University, Khulna, Bangladesh, in 2022. Her main research interests are digital circuits design, system on-chip design & very large-scale integration(VLSI). I am currently working as an ASIC physical design engineer at Primesilicon Technologies (BD) Ltd.

S. M. Shamsul Alam received the B.Sc. (Engg.) degree in Electronics and Communication Engineering from Khulna University, Khulna, Bangladesh in 2004 and M. Engg. degree from the Department of Information and Communication Engineering Chosun University, Gwnagju, Korea, under the Global IT, NIPA scholarship program in 2013. From 2011 to 2013, he was working as a Research Assistant with the Department of Information and Communication Engineering, Chosun University, Gwangju, Korea. Currently, he is with the Electronics and Communication Engineering (ECE) Discipline, Khulna University, Khulna, Bangladesh and is serving as a Faculty Member in the ECE Discipline. His research interests include Chip Design and Application Specific Processor Design for communication systems..

How to cite this paper: Jabin Sultana, S. M. Shamsul Alam, "Static Timing Analysis of Different SRAM Controllers", International Journal of Intelligent Systems and Applications(IJISA), Vol.15, No.3, pp.33-43, 2023. DOI:10.5815/ijisa.2023.03.03