International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Design of Clocked Synchronised Timing Methodology for Advance Digital System

Manjunath C[1], Harish P[2], Vinay Kumar AN[3], Prof. Pavan Kumar E[4]

[1-4]*Department of Electronics and Communication, Sai Vidya Institute of Technology, Bengaluru, India*

Abstract - *In digital and distributed systems the clock inaccuracies cause some serious problems, thus to have efficient communication or data sharing, the clocks must be synchronized or adjusted continuously. As Asynchronous timing method is very significant in the design of large scale digital electronic circuits.*

This paper concentrates or overviews on clock design, clock distribution and timing methodology and describes how synchronization can be handled by minimizing the clock skew, ensuring that flip-flops have positive setup- and hold-time margins, synchronizing the asynchronous inputs with the clock also the asynchronous inputs to fan out to more than one flip-flop should be not allowed.

The clocked synchronous timing methodology involves a single or common clock for all registers, and operation on data by combinational circuits between clock edges and also reducing noise at the sequential elements driven by clock signal.

***Key Words*: Synchronous, Asynchronous input, Clock, Timing methodology**

1. INTRODUCTION

The clock signal is used to determine a time reference for the transfer of data . The clock distribution network distributes the clock signal from a clock source to all sequential elements which require it.

The digital systems consist a data path that stores and transforms binary coded information and a control section that sequences operations within the data path. The data path contains combinational sub circuits that implement operations on the data and registers that store the data. Stored data can be fed back to previous or earlier stages of the data path or fed forward to next stages. The control section drives the control signals that manage operation of the combinational sub circuits and storage of data in the registers. The control section can also use status information about the data values to determine what operations to function and in what sequence[1].

The complexity of timing in digital systems is governed by the division of time into discrete intervals is a key abstraction.

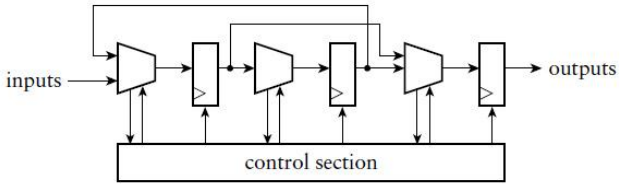


Fig – 1: A overall view of a digital system.

In the Fig 1 shown above all the registers are connected with a common periodic clock signal i.e. registers consisting of flip flops are all clocked synchronously on each rising clock edge. The clocked synchronous timing methodology guide us guaranteed operations that are completed by combinational sub circuits by the time their results are produced, and makes clear formation of large systems from smaller subsystems.

We will make an assumption that all the flip-flops in a given register have the same timing characteristics, or that any differences are neglected. One can thus identify the setup time tsu, hold time th and clock-to-output delay tco of a register as being the same as those characteristics of the continuous flip-flops. The data stored in a register must be secure or stable at the input partially the setup time before a clock edge and slightly the hold time after the clock edge[1]. The past stored data will only be availabale at the output after the clock to output delay following the clock edge.

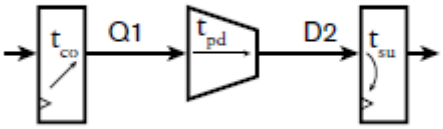


Fig – 2a: A register to-register path.

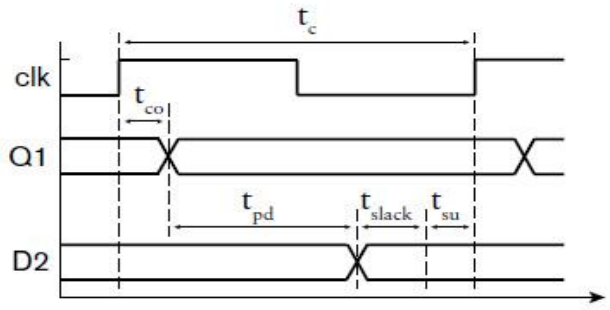


Fig -2b The timing shows that the sum of these intervals must be equal to the clock cycle time (tc).

The path with the longest propagation delay is called the critical path, Since the critical path controls the entire system performance we need to revise performance issues and identify which combinational sub circuit is on the critical path and tries to reduce its delay. Most systems the critical path will be register to register path in the data path of the system. For instance, there is a path that has a counter the carry chain may be the critical path when the delay on the critical path is reduced below that of another path, that other path becomes the critical path, hence it is very necessary to pay attention to other paths as well to address the performance issues.

The timing considerations originate from the way the clock signal is connected to all the registers in a circuit. For example, in a register to register path, clock signal is directly connected to target register through long wire with substantial delay, as shown in Fig -3a. When a rising clock edge appears at the source register prior to the target register. This phenomenon is called clock skew. If the propagation delay through the combinational sub circuit is less the hold time after the clock edge may not remain stable from the previous cycle value as shown in the Fig -3b.

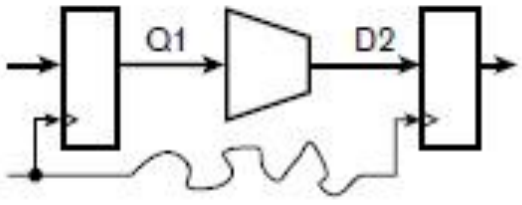


Fig-3a A register to register path with clock skew [1]

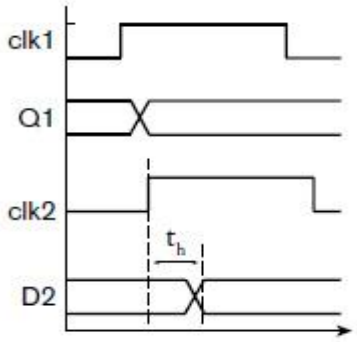


Fig- 3a shows the timing problem arising from clock skew

The timing characteristics or parameters and the constraints we have considered so far have been applied to the control section and the data path which are part of the integrated circuit chip. The excessive voltage and the static discharge that are caused in the chip are protected by the internal buffers and the the outputs have buffers to drive the relatively large capacitances and inductances that occur outside the chip. These buffers and the wiring connecting the integrated circuit chip to the package pins, introduce propagation delays. When the timing behavior of the entire system is analyzed one need to introduce the pins and the wiring delays. The sum of the propagation delays must be less than the system's clock cycle time. This feature will be a difficult constraint to meet for High speed systems, in such cases we typically evade having any combinational input. An input which connects directly to an input register is called a registered input, and an output that is driven directly from an output register is called a registered output.[1]. These High-speed design methodologies frequently require registered inputs, registered outputs or a combination of both.

The designing of a clock synchronous timing methodology system must have three well-defined tasks to overcome the synchronization failures.

1. Minimalize and limit the amount of clock skew in the system.
2. Guarantee that flip-flops have positive setup- and hold-time margins, including an allowance for clock skew.
3. Recognize asynchronous inputs, synchronize them with the clock, and make sure that the synchronizers have an adequately low probability of failure.

1.1 Synchronization Failure

a. Synchronization failure occurs when Flip Flop input changes to clock edge.
b. The flip flop might enter a metastable state i.e a state of logic 0 or logic 1.
c. This might stay in this state an indefinite amount of time.

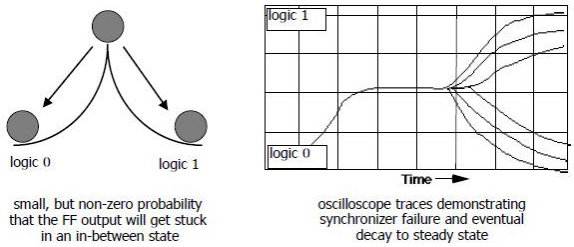


Fig-4: Representation of Synchronization Failure

1.2 Dealing with Synchronization Failure

- The Probability of failure can never be reduced to 0, but it can be able to be reduced.
- Decrease the system clock, this gives the synchronizer more time to decay into a fixed stat.;
- Synchronizer failure becomes an enormous problem for very high speed systems
- Use fastest possible logic technology with in the synchronizer.
- This makes a sharp peak upon which to balance.
- Also by cascading two synchronizers.

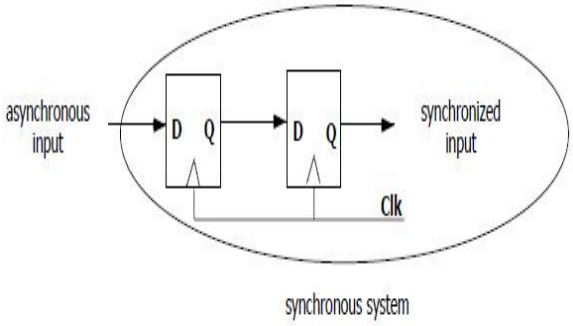


Fig-5: A synchronizer for an asynchronous input.

2. DESIGN METHODOLOGIES

2.1 Asynchronous Inputs

The clocked synchronous timing methodology requires us to verify that during an interval around each clock edge inputs to registers are stable. [1]

Make sure that we meet this constraint for those signals that are generated within the circuit. Most circuits must handle some inputs that are generated externally, either by transducers whose outputs represent real-world quantities or events, or by separate systems that don't share a typical clock. We call such signals asynchronous inputs [1].

One cannot guarantee that they meet timing constraints for register inputs because no one has control over no of times at which they modify value. There are several digital systems that accept asynchronous inputs from outside or external devices and it's vital that these asynchronous signals should be synchronized with the system clock. There is also possibility where the input or the incoming signal features a brief time duration by comparison with the sampling period of the system clock. A particular example of such an incidence is shown in Fig- 6 and it's clear that the asynchronous input misses the sampling fringe of the clock, during this instance its trailing edge.

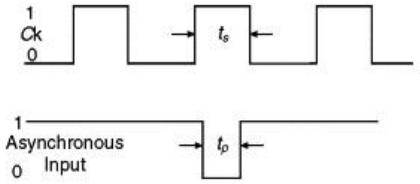


Fig-6: Timing diagram for asynchronous input that was never sampled.

This problem can be solved by the addition of a catcher cell which is in effect of an SR latch whose set signal is the asynchronous input with the reset signal coming from the complementary output of Flip Flop 1. The System flip fop FF1 is connected to the catcher cell that consists of the gates, as shown in the Fig7.

We can assume that Q is equal to 0(Q=0) in the same way D= 0 and A = 1, the logic levels for this condition are specified at certain points of the circuit. On the onset of the asynchronous signal, A makes 1 to 0 transition, and D makes a 0 to 1 transition. 1 and 0 are the respective outputs of the latch. Before the edge of the clock pulse the catcher cell input a makes a returns from 0 to1.

Flip Flop1 is now fixed the edge of the clock pulse and therefore, the complementary output Q^- is fed back to the lower input of the S^-R^- latch, resetting the catcher cell to its original condition before the arrival of the subsequent asynchronous input [2]. The catcher cell has been accustomed synchronize the asynchronous signal to the system clock. Instead of using the straightforward catching cell shown in Figure 7 , an edge-triggered D Flip Flop may be used because the synchronizer as shown in Figure 8. The asynchronous input should be routed to variety of various points of the system and each single asynchronous signal requires its own synchronizer and, the synchronization should happen at only one point and any delays must be corresponded carefully. Moreover, it's always suitable to come first any combinational logic with the synchronization process due to differing combinational delays.

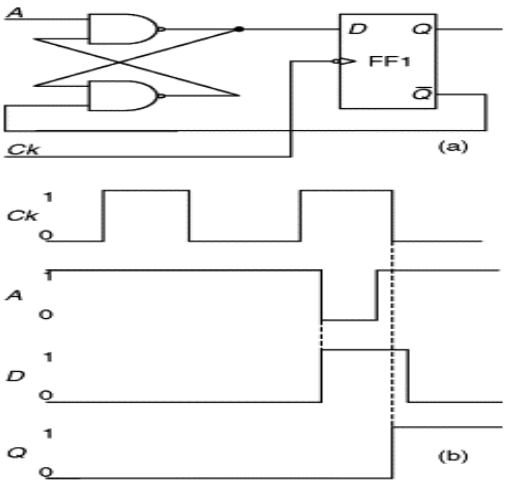


Fig-7: Catching an asynchronous signal (a) the circuit (b) the timing diagrams

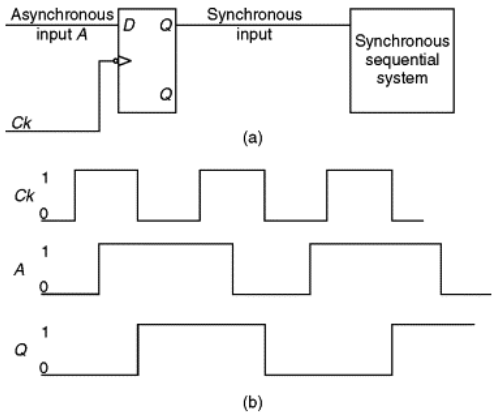


Fig 8: Synchronizer circuit (a) logic diagram (b) timing diagram

Synchronization failure may arise irregularly and disturb the operation of the system.

Handling of Asynchronous inputs

- Asynchronous inputs to fan-out to over one flip-flop should be not allowed.
- Synchronize as soon as possible and then treat as synchronous signal.

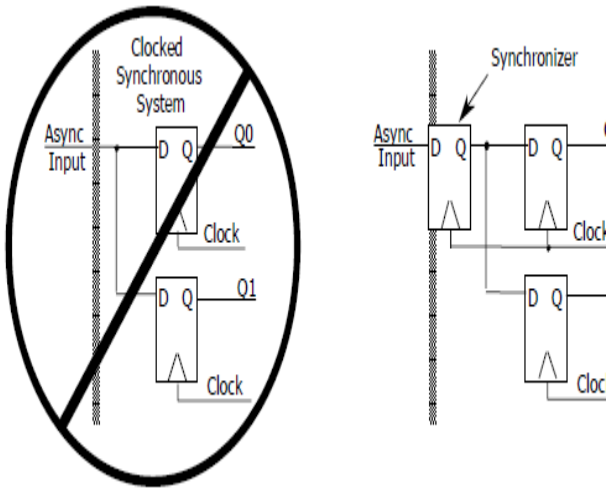


Fig-9: Representation of handling asynchronous inputs.

2.2 Verification of Sequential Circuits

As the design of clocked sequential circuits and the timing constraints are discussed, we need to look at the functional verification and timing verification i.e. functional verification using Verilog models.

The fact is that the operations take one or more clock cycles to finish when verifying a sequential circuit also make sure that by guarantee the procedural block that checks the output is synchronised with the stimulus block, and knows what number of clock cycles after application of test case to wait before checking the output.

If all operations complete within the same number of cycles, and just one operation takes place at a time, this is often relatively straightforward. On the opposite hand, if operations take varying numbers of cycles to finish, the checker must check both that the operation completes at the right time which the right result's produced.

If multiple operations can happen concurrently, as an example, if the data path may be a pipeline, the checker must

make sure that all operations that start also complete, which no spurious results are produced.

Verification of sequential circuits using verliog code and simulation are provided in simulation using Xilinx 3 section.

2.3 Asynchronous timing methodology

Though the clocked synchronous methodology provides significant simplifications, there are some applications where it breaks down. Two important assumptions are that the clock signal is distributed globally i.e. across the complete system with minimal skew, while the propagation delay between registers is a smaller amount than a clock cycle. In large high-speed systems, these assumptions are very difficult to take care of. For instance, in a very large unified circuit operating with a clock frequency of several GHz, the time consumed for a change of signal value to propagate along a wire that stretches across the chip could also be an oversized proportion of a clock cycle, or perhaps over a clock cycle.

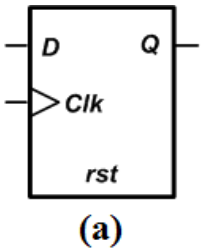
One of the solution is. We can reconsider by assuming single global clock signal for the entire chip or system. The system is divided into several regions or segments each with its individual or own clock.

Where signals connect from one region to a different, they're treated as asynchronous inputs. The timing for the system is claimed to be Globally Asynchronous Locally Synchronous GALS [5]. The constraints on clock distribution and timing within each region is simpler to manage. The downside is that inter-region connections must be synchronized, thus adding delay to communication. The challenge for the system architect is to search out a partitioning for the system that minimizes the number of communication between regions.

The amount of difficulty in distributing high-speed clock signals and managing timing is even greater within the terms of a entire printed circuit consisting of several integrated circuits. It's simply not practical to distribute a high-speed clock across an outsized system. Rather a slower clock is commonly used externally to high-speed chips, and operations between chips are synchronized to that external clock. The internal clocks operate at a frequency that's a multiple of the external clock, with synchronization of clock edges. The separate boards in an exceedingly high-speed system typically aren't synchronized, but have independent clocks. Data transmitted from one board to a different is treated as an asynchronous input by the receiving board. A separate issue with the clocked approach is that clocked circuits consume significant amounts of power. Whether or not a flip-flop doesn't change its stored value, changing the clock input between 0 and 1 involves switching transistors on and off, thus consuming extra power [1].

In applications with very low power budgets, like battery powered mobile devices, this waste of power is unacceptable. Asynchronous circuits are an option since logic levels only change when data values change. The circuit becomes quiescent if there is no new data to work upon. A few low-power products using asynchronous circuits are successful. Low-power applications could also be a more significant motivation for asynchronous design than the potential performance gains.

3. SIMULATION USING XLINIX



(a)

Fig- 9a: above shows the block diagram for positive edge and negative edge D Flip Flop.

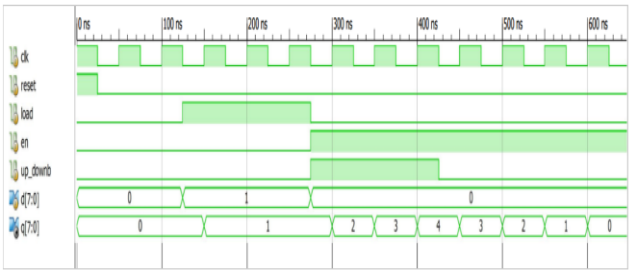
Verilog Code for DFF:

```
1   module D_FF
2       (
3           input wire clk, reset,
4           input wire d,
5           output reg q
6       );


7       always @(posedge clk, posedge reset)
8           if (reset)
9               q <= 1'b0;
10          else
11              q <= d;

12  endmodule
```

Testbench code is as follows:

```
module bidirect_cnt8
    (
        input wire clk, reset,
        input wire en, load, up_downb,
        input wire [7:0] d,
        output reg [7:0] q
    );

    reg [7:0] q_next;

// The storage elements
    always @(posedge clk, posedge reset)
        if (reset)
            q <= 8'h00;
        else
            q <= q_next;

//The next state logic
    always @*
        if(load)
            q_next = d;
        else if (~en)
            q_next = q;
        else if (up_downb)
            q_next = q + 8'h01;
        else
            q_next = q - 8'h01;


endmodule
```

The simulation is as follows:



4. CONCLUSIONS

From timing constraints to the variation in clock timing such as delay and clock distribution and timing inside each region are discussed in this paper and the methods to overcome this such as to operate efficiently clock skew should minimized. CAD tools typically implement clock distribution to attenuate skew. Registered inputs and outputs decrease combinational delays in interchip register to register paths, and thus help in meeting timing constraints.

ACKNOWLEDGEMENT

We would like to thank and express our gratitude to our college Sai Vidya Institute of Technology and Visveshwaraya Technological University (VTU) for giving us this opportunity to enhance our knowledge through this work.

REFERENCES

[1] Ashenden, P. (2008). Digital design. Amsterdam: Morgan Kaufmann Publishers.

[2] Holdsworth Digital logic design. (2002).

[3] Components and Design Techniques for Digital Systems. http://cseweb.ucsd.edu/classes/sp10/cse140L/lectures/lab_wk4.pdf

[4] Joo, D., Kang, M., & Kim, T. (2012). Design Methodologies for Reliable Clock Networks. Journal Of Computing Science And Engineering, 6(4), 257-266. doi: 10.5626/jcse.2012.6.4.257 https://www.researchgate.net/publication/313118020_Design_Methodologies_for_Reliable_Clock_Networks

[5] GALS System Design: Side Channel Attack Secure Cryptographic Accelerators. (n.d.). Retrieved May 17, 2020.from https://iispeople.ee.ethz.ch/%7Ekgf/acacia/acacia.html