Int. J. Chem. Sci.: 14(S3), 2016, 733-740
ISSN 0972-768X
www.sadgurupublications.com

DYNAMIC CLOCK ALIGNMENT USING DELAY LOCKED LOOP

G. SNIGDH and S. SIVANANTHAM*

Department of Micro and Nanoelectronics, School of Electronics Engineering, VIT University, VELLORE – 632014 (T.N.) INDIA

ABSTRACT

In system-on-chip (SoC) design, a buffered clock distribution network is typically used to drive the large clock load. Chip design involves a clock alignment step, which equalizes the delay from the clock source to each and every clock target (flip flops, latches, or other memory elements). Accurate clock alignment is important, because unwanted differences or uncertainties in clock network delays may degrade performance or cause functional errors. Clock distribution and alignment has become an increasingly challenging problem in very large scale integration (VLSI) design, consuming an increasing portion of resources such as wiring area, power, and design time. The clock skew problem is more prominent in the case of an SoC (System-on-Chip) device where many blocks need to communicate each other and have different internal clock tree delays depending on their clock tree depth. The objective of the thesis is to address the problem of clock skew between two different modules in modern day microprocessors or any high speed digital design, which is caused by different clock tree insertion delays and due to process, voltage and temperature (PVT) variations. This paper presents an automatic clock skew control scheme in order to mitigate the misalignment of the clocks in the different regions of SoC. The stated approach requires Delay Lock Loop (DLL) to add or subtract the delay to keep the clocks continuously aligned to a common reference clock delay. For Simulation results of the design Cadence compilercverilog and simvision have been used.

Key words: Delay locked loop, System on chip, Clock skew, Physical design.

INTRODUCTION

A system on a chip (SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions-all on a single chip substrate. Clock skew is a phenomenon in synchronous circuits in which the same sourced clock signal arrives at different components (generally latches or flip-flops) at different times. This can be caused by many different things, such as wire-interconnect length, temperature variations, variation

*Author for correspondence; E-mail: ssivanantham@vit.ac.in

in intermediate devices, capacitive coupling, material imperfections, and differences in input capacitance on the clock inputs of devices using the clock[5,8]. There are two types of clock skew. They are negative skew and positive skew. Positive skew occurs when the transmitting register receives the clock tick earlier than the receiving register. Negative skew is the receiving register gets the clock tick earlier than the sending register. Zero clock skew refers to the arrival of the clock tick simultaneously at transmitting and receiving register. Theoretically Zero clock skew is possible but in practical situations it is not possible. Even if zero clock skew is achieved by chance, the power consumption at that instance is very large and might cause damage to the circuit. Hence for practical purposes there should be a skew value but it should be minimal and negligible.

This paper presents a clock alignment scheme in a system-on-chip (SOC) in which there are different delays in each block depending on the clock tree depth. This is done by implementing a digital clock management circuit (DCM) for each block in an SOC by giving the same reference clock to all the DCM's in the SOC, the clock delays are added or subtracted by the DLL which is present in thee DCM and inside each block the clocks are aligned by using clock tree method. The rest of the paper is as follows: Section(II) consists of Background, Section (III) comprises of proposed work, Section(IV) has results, Section(V) has conclusion and future work.

Background

In general, it is relatively easy to match either the clock buffer delays or the RC delays by themselves separately. However, since the wire resistance and capacitance varies differently from the gate trans conductance and the parasitic diode capacitance (under various processing technologies and operating conditions), matching both components together is not an easy task. Furthermore, since the RC delay values are totally dependant on the physical layout of the device, an IC designer can only guarantee the minimum clock skew require- ment by tuning the RC delay along the clock tree after the near completion of the physical design (layout) stage. In fact, in spite of all the tuning work, the minimum clock skew is best guaranteed for a narrow operation range[1-3].

Previously, the clock skew problem in an IC device was fixed by balancing the clock tree (matching clock skew) using one or both of two main techniques. The first technique is to balance the RC delays among different clock branches. The RC delay values are totally dependent on the physical layout of the device, an IC designer can only guarantee the minimum clock skew requirement by tuning the RC delay along the clock tree after the near completion of the physical design stage. In fact, in spite of all the tuning work, the minimum clock skew is best guaranteed for a narrow operation range. The first method is the balancing of RC delays. Resistive-capacitive delay, or RC delay, hinders the further

increasing of speed in microelectronic integrated circuits. When the feature size becomes smaller and smaller to increase the clock speed, the RC delay plays an increasingly important role.The second common technique is used to balance the clock buffer (clock insertion delay). This technique is generally used to synchronize the I/O signals among various macros (cores) with or without phase synchronizing the internal clocks within each macro (core). The pitfall of the second method is that it completely ignores the wire RC delay. If there is a substantial RC delay on the clock wire, then they cannot be synchronized. Thus, this is not a suitable for SoC design technique[4,7].

Proposed work

This paper describes an automatic clock alignment mechanism or skew control mechanism. We use the Digital clock management (DCM) to control the clocks or to align the clocks of the different blocks in an System-on-chip (SOC).A phase detector is used which compares the signal with reference and tells whether it is an early or a late signal and using a DLL we create that much delay to the signal and align both the clocks. This alignment is done not in a single step but in a series of steps and the clock increment is done gradually and not at once to avoid any loss of data. Also the clock is not aligned based on a single pulse of early or late signal, but after a series of pulses however here we take it as after every 128 pulses of early and late signals, increment or decrement of clock is done. So inside a system-on-chip(SOC) for every block there is a corresponding digital clock management circuit (DCM) to which a reference signal is given which is common to all and based on this reference signal the clocks get aligned.

The overall design can be simplified with the help of the block diagram that consists of different segments where each of these segment have their own individual functionalities. The various segments in the block diagram are Bang Bang phase detector, sampler, divided-by-8 logic, majority logic, 8-bit accumulator, decision making and code generation, delay locked loop (DLL). The block diagram of DCM (Digital Clock Alignment) is:

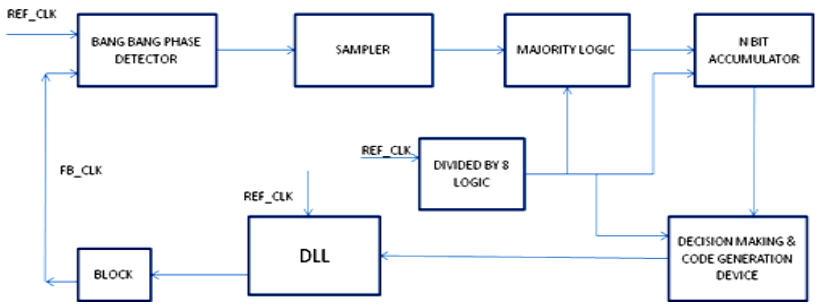

Fig. 1: Block diagram of digital clock management (DCM)

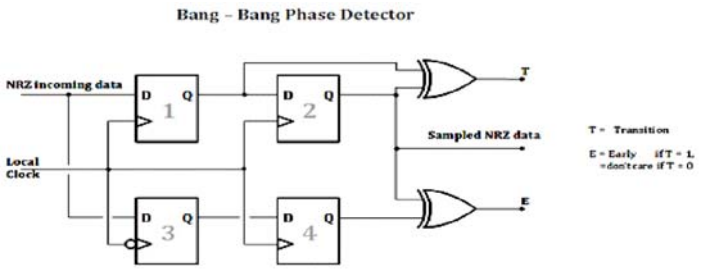

Fig. 2 Block diagram of Bang Bang phase detector

Bang Bang phase detector

A Bang Bang Phase Detector is a special type of phase detector which is used to compare the reference and the feedback clock signals to generate an early signal (or an up signal) or late signal (or a down signal). The bang bang phase detector has two inputs and two outputs. The two inputs are a reference clock and a feedback clock. Also the nyqust criteria should be followed i.e. the sampling rate should be twice the clock rate so we give the feedback clock to a divided by two logic and then give it as an input to bang bang phase detector. The two outputs are a transition_out and an out. When the transition_out is equal to 1 and then if the out is equal to 1, then it is an early signal or else if out is equal to 0 then it is a late signal. The output of this Bang Bang phase detector is given to the sampler as input. It gives a series of ones and zeros as input based on whether it is an early or a late signal.

Sampler

The sampler block is used to accumulate either up or down signals up to eight clock pulses in each 8 bit register one for up and one for down. Output is sent after every eight clock pulses. The input is obtained from the bang bang phase detector. There are two inputs one from the transition_out and other from out. In this Sampler block, there are two 8-bit-shift registers one for up signals and other for down signals. When the transition_out is equal to 1 and out is equal to 1 then 1 is sent into the up register and 0 to down register. When the transition_out is equal to 1 and out is equal to 0 then 0 is sent into the up register and 1 to down register. When the registers receive another value either 1 or 0, the previous value is shifted to right. In this way the values are shifted up to eight clock pulses and after which the output of these two registers is sent to majority logic block.

Divided by 8 logic

The frequency divideror clock divider or scaler or prescaler, is a circuit that takes an input signal of afrequency, f_{in}, and generates an output signal of a frequency:

$$f_{out} = \frac{f_{in}}{n}$$

where n is an integer, here we take the value of nas 8 as we want a divided by 8 logic.

This divided by 8 clock is given as input to majority logic block, 8-bit-accumulator block, decision logic and code generation block as these blocks work at a low frequency and also need output after 8 clock pulses of the normal reference signal as they get input from sampler only after every 8 clock pulses.

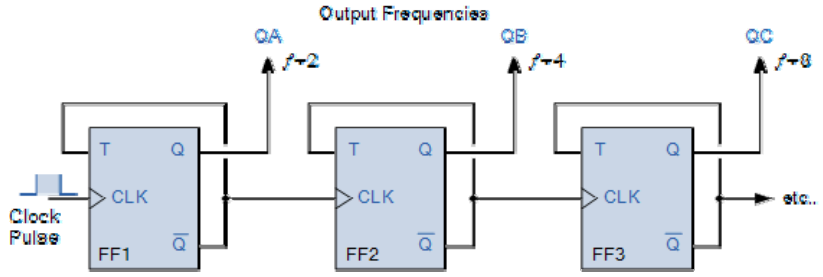

Fig. 3: Block diagram of divided by 8 logic

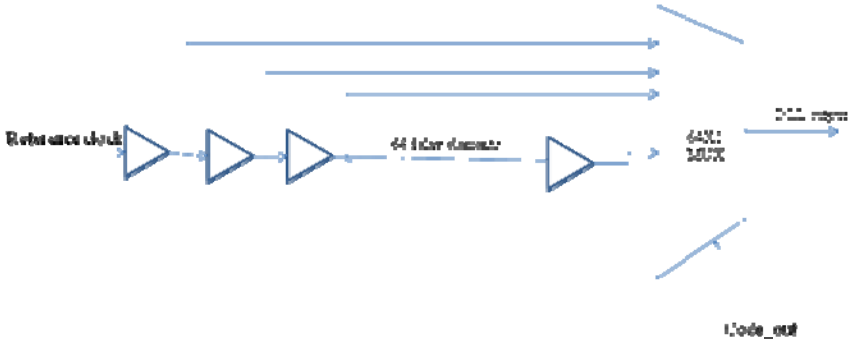

Fig. 4: Block diagram of delay locked loop

Majority logic

The majority logic block is used to the compare the number of ups and number of down in the given input and the difference between them is given as output. The majority logic block counts the number of ones in the up register and the number of ones in the down register and the counts are stored in two different variables and then the difference of up and down is sent as output to the 8-bit-accumulator. The clock input to this block is the divided by 8 logic clock. There are two inputs to the majority logic and one output. The inputs are 8-

bit which contain the no of ups and downs in the each. The counting of these up signals and down signals are done by using two different 8-bit counters. Then by using a 8-bit subtractor the number of down signals is subtracted from the number of up signals and this output is sent to the 8-bit-accumulator.

8-Bit-accumulator

The accumulator used can be of any bit size, but here for our design we are using an 8 bit accumulator. The accumulator is used here to count up to +127 up signals or -128 down signals. Once the desired count of up or down signals is reached then only there is a change in the output value sent, the output value sent is a 1 for an up signal and a 0 for a down signal. This block is used because in the design we should not increment or decrement the clock signal for every clock cycle but should but done only if continuously a signal comes early or late, then only the clock is to be incremented or decremented because the clock is not usually stable and has other jitter components also due to which there might be undesired variations which should not be considered. The 8-bit-accumulator has one input which is an 8 bit input. The output of this 8-bit-accumulator is a single bit value representing 127 up signals or down signals which is given to a decision making and code generation block.

Decision making and code generation

This decision making and code generation block is used to create the delay based on the input from the 8-bit-accumulator which is given to the selection line of a 64X1 mux of DLL to select the delay required for the clock alignment. The input of the decision making and code generation block is from the 8-bit-accumulator, which consist of either 1 or 0 representing an up or down signal and these inputs are either added or subtracted to the output of this block. The output of this block is 6-bit output as the DLL of mux has 64 delay elements and based on this output the delay is selected from the different types of delays generated by mux. The output of this decision making and code generation block is given to the DLL.

Delay locked loop

DLL is a feedback system that aligns the feedback clock to the reference clock. This is done by delaying the input feedback clock after passing it through a delay line and controlling the delay using the control mechanism. Once the input feedback clock is delayed, a phase detector (PD) compares the phases of the two inputs. Based on PD output value, the delay is adjusted (increased or decreased) until the two phases are aligned. A DLL is widely used as a timing circuit in many systems for the purpose of clock generation, signal synchronization and other purposes. Analog DLL's are mainly used for clock distribution purposes. In our design we go for digital DLL. Here in our design, we use a DLL which is modeled by 64 delay elements. The outputs of all these 64 delay elements is given to a 64X1

mux and the selection line of this mux gets input (6-bit input to select from the 64 delay elements) from the decision making and code generation block. The delay of each element is clock period/64*signal. The delay increases for every delay element increases by clock period/64. The output of this 64X1 mux is given to the block in which the clocks are aligned by using clock tree and then the output of these block is given as feedback to the Bang Bang phase detector from where the whole process repeats again.

RESULTS AND DISCUSSION

The design of the DCM block when analysed in a waveform analysersimvision wherein the delay between the feedback and reference clock is 2ns, the clock gets aligned after about 5000ns.

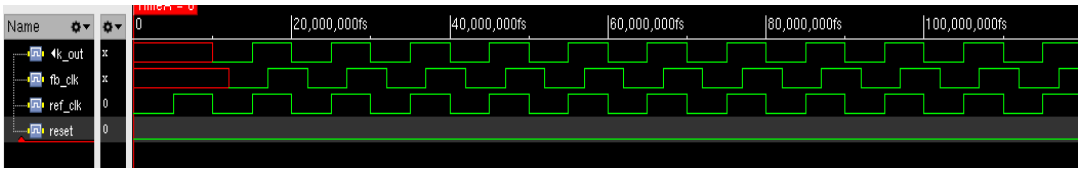

Fig. 5: DCM-1

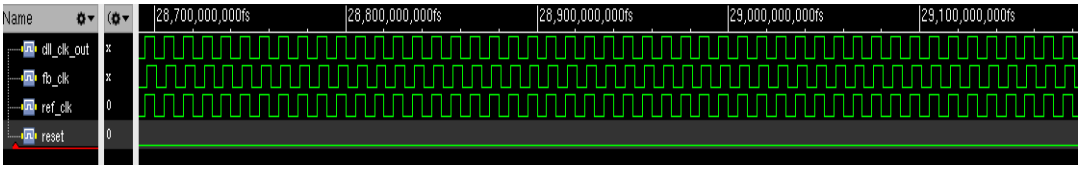

Fig. 6: DCM-2

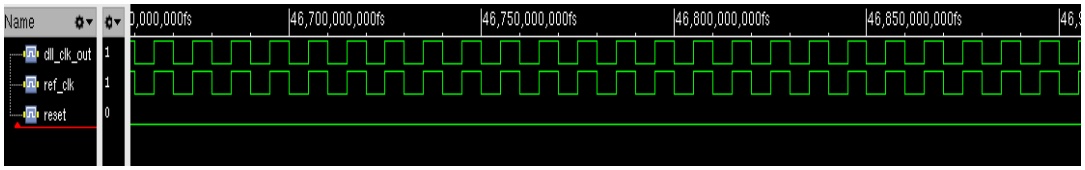

Fig. 7: DCM-3

Initially in Fig. 5: DCM -1, the dealy between the reference clock and the feedback clock is 2ns and gradually the delay is reduced and the clocks get aligned at about 5000ns.

CONCLUSION

By using this DCM block, the clocks will successfully be aligned dynamically and there will be no loss of data as clocls are synchronised. The power consumption in the

design by the use of this block is reduced by 30%. The DCM block designed in this work is useful in aligning the clocks when there are different voltage levels in a design, like the memory controller and DDR-5 where in the voltage levels are different and hence can be aligned using this DCM block.

REFERENCES

1. H. Terng-Yin, B.-J. Shieh and C.-Y. Lee, An All-Digital Phase-Locked Loop (ADPLL)-Based Clock Recovery Circuit, IEEE J. Solid-State Circuits, **34**, 1063-1073 (1999).
2. Stojčev, Mile and Goran Jovanović, Clock Aligner Based on Delay Locked Loop with Double Edge Synchronization, Microelectronics Reliability, **48**, 158-166 (2008).
3. Lee, Hyun, Han Quang Nguyen and D. W. Potter, Design Self-Synchronized Clock Distribution Networks in anSoC ASIC using DLL with Remote Clock Feedback, ASIC/SOC Conference, Proceedings, 13th Annual IEEE International, IEEE (2000).
4. S. Sivanantham, R. Adarsh, S. Bhargav and K. Jagannadha Naidu, Partial Reconfigurable Implementation of IEEE802.11g OFDM, Indian J. Sci. Technol., **8(36)** (2015).
5. S. Sivanantham, K. Jagannadha Naidu, S. Balamurugan and D. B. Phaneendra, Low Power Floating Point Computation Sharing Multiplier for Signal Processing Applications, Int. J. Engg. Technol., **5(2)**, 979-985 (2013).
6. A. Mishra, S. Sivanantham and K. Sivasankaran, Sine and Cosine Generator using CORDIC Algorithm Implemented in ASIC (2015) IC-GET 2015 - Proceedings of Online International Conference on Green Engineering and Technologies (2015).
7. A. Tiwari, S. Phapal, D. Kadam, S. Sivanantham and K. Sivasankaran, FPGA Implementation of FFT Blocks for OFDM, IC-GET 2015 - Proceedings of Online International Conference on Green Engineering and Technologies (2015).
8. S. Sivanantham, G. Gopakumar, A. Pandey and M. J. Paikada, Adaptive Test Clock Scheme for Low Transition LFSR and External Scan Based Testing, International Conference on Computer Communication and Informatics, ICCCI (2013).

Accepted : 11.10.2016