SYNCHRONIZATION IN DIGITAL SYSTEM DESIGN

RONIT YADAV

SACHIN SOLANKI

SAHIB ARORA

ABSTRACT

In digital system design, the methods of synchronous & asynchronous designing have their own merits and de-merits. Designers adopt the appropriate method based on the requirements like power efficiency, modularity, performance, clock skew, concurrency, metastability, testability, design tools support, automatic adaptation to physical properties and many more. The “need for testability” is the driving factor to select the method of synchronous design. This helps to achieve comparatively better testability than the design based on asynchronous design. The synchronous design method was applied for a bus control unit to control heater switches from both the serial and I/O ports.

Introduction

Synchronization involves determining or enforcing and ordering of events on signals. In digital system design, synchronization ensures that operations occur in the logically correct order, and is a critical factor in ensuring the correct and reliable system operation. As the physical size of the system increases, or as the speed of operation increases, or issue of testability arises then synchronization plays an important role in the system design. Digital abstraction depends on all signals in a system having a valid logic state. Hence digital abstraction depends on reliable synchronization of external events. Synchronization involves imposing or recognizing an ordering of events on signal lines. An asynchronous signal, which may change at any time, must be synchronized with a clock before it can be used as an input to synchronous logic . The first section details a few points on the disadvantages of asynchronous design & second section outlines some uses of synchronization. The third section details on the mechanism of synchronization. The fourth section gives the details on synchronization and the fifth section concludes with the process of synchronization as applied to the heater switching unit followed by acknowledgement and references.

USE OF SYNCHRONIZATION

Uses of Synchronization Operations in a digital system can either proceed concurrently, or they must obey a precedence relationship. If two operations obey precedence then the role of synchronization is to ensure that operations follow the correct order. Synchronization is thus a critical part of digital system design. There are three common uses of synchronization

• arbiting between asynchronous requests

• Sampling asynchronous signals with a clock

• Transmitting synchronous signals between two clock domains The most common approach to synchronization is to distribute a clock signal to all modules of the system. With the scaling of feature size in VLSI design, clock speeds are increasing rapidly, but increases in the complexity tend to prevent significant reductions in chip size. As a consequence of this scaling, clock speeds in digital system design are increasing in relation to propagation delays. However because of the wide difference between system physical sizes in relation to clock speeds, the design styles of the communities have developed almost independently. Also the impact of accessibility on testing leads to the observation that asynchronous design or those with unconstrained timing signals are much more difficult to test than synchronous designs that have easily accessible clock generation and distribution circuits.

Abstractions in Synchronization:-

Abstractions are often applied in a hierarchical fashion, where each layer of abstraction relies on the essential features of abstraction levels below and hides unessential details from the higher level. Abstraction applied to digital system design At the base of the design we have the physical representation where we have semiconductor materials, interconnect metallization. Above this is the circuit abstraction, where we have transistors, interconnections. Next is the element like flip flops and gates. Here we describe elements in terms that deals with their operational characteristics.(timing diagrams, Boolean functions etc.,). Next is module where we group elements to form more complex entities like memories, register files, arithmetic logic units etc. While abstractions are very useful and in fact absolutely necessary, they should be applied with care. The essence of an abstraction is that we are ignoring some details of the underlying behavior, which we hope are irrelevant to the operation. It should always be verified that in fact the characteristics being ignored are in fact irrelevant and with considerable margin, or else the final system may turn out to be inoperative or 143 unreliable. This is especially true of synchronization, which is one of the most frequent causes of unreliable operation of the system.

➢ Synchronization:-

The role of synchronization is to coordinate the operation of a digital system. we review two traditional approaches to synchronization in digital system design: synchronous and asynchronous interconnection. we briefly describe how synchronization is accomplished in digital communication systems. This will suggest, as discussed further in Section, opportunities to use digital communication techniques in digital system design. A. Synchronous Interconnection As shown each element (or perhaps module) is provided a clock, as well as one or more signals that were generated with transitions slaved to the clock. The common clock controls the order of operations, ensuring correct and reliable operation of the system. We will examine some fundamental limitations in the operation of synchronous interconnection, making idealistic assumptions about the ability to control the clock phases in the system and neglecting

interconnect delays. we will show how pipeline registers can be used to extend the performance of synchronous interconnect. we will make more realistic estimates of performance considering the effects of the inevitable variations in clock phase and interconnect delays. I) Principle of Synchronous Interconnection: The fundamental principle of synchronous interconnection is illustrated. In this a computational block C1 is connected to a synchronizing register R1 at its input. This register is clucked using the positive transitions of a periodic clock signal, where the assumption is that the output signal of the register changes synchronously with the positive transition of the clock. The computational block performs the same computation repeatedly on new input signals applied at each clock transition. The purpose of R1 is to control the time at which the computational block starts to perform its work, in order to synchronize it to other computational blocks in the system. This is an edge-triggered logic model, which we employ for its relative simplicity. There is also a more complicated level-sensitive model that leads to virtually identical conclusions. The performance measures of interest are the throughput (rate at which the computation is repeated) and computational latency (delay from the time a new input is applied until the result is available). Focusing on the latter, inevitably the computational latency is not entirely predictable. It is likely that the output signal will change more than once before it finally settles to its final correct value. For example, if the output signal actually consists of M > 1 Boolean signals in parallel, as is often the case. some of those Boolean signals may transition before others, or some may transition more than once before reaching a steady-state value. This behavior is an inevitable consequence of real circuit implementation of the computational block, and presents a considerable problem in the synchronization to other computational blocks. Assume the computational block has a minimum time before any outputs transition, called the propagation time t,,, and a maximum time before all the outputs reach their final and correct values, called the settling time Since settling time is the maximum time before the result is guaranteed to be available, it is also the computational latency. It is assumed that the propagation and settling times can be characterized and ensured over expected processing variations in the circuit fabrication. The synchronous interconnect isolates the system behavior from these realities of circuit implementation by setting the clock period T so that there is a certainry period during which the output signal is guaranteed to be correct? and then samples the output signal again (using another register R2) during this certainty period as shown. Of course, the phase of the clock for R2 must be adjusted to fall in this certainty period. This interconnect is called synchronous because the proper phase of clock to use at R2 must be known in advance-there cannot be any substantial uncertainty or time variation in this phase. With synchronous interconnect. the undesired behavior (multiple signal transitions and uncertain completion time) is hidden at the output of R2. We can then abstract the operation of the computational block, as viewed from the output of R2, as an element that completes its computation precisely at the positive transitions of the second clock, and this makes it easy to synchronize this block with others since the uncertainty period has been eliminated.

Conclusions:-

In this paper we have attempted to place the comparison of digital system synchronization on a firm theoretical foundation, and compare the fundamental limitations of the synchronous, asynchronous, and Inesochronous approaches. A firm conclusion is that interconnect delays

place a fundamcntal imitation on the communication throughput for asynchronous interconnect (equal to the reciprocal of two or four delays), this limitation does not exist for mesochronous interconnect. Further, mesochronous interconnect can actually achieve pipelining in the interconnect (as illustrated) without additional pipeline registers, whereas asynchronous cannot. Further, asynchronous requires extra interconnect wires and completion signal generation. Thus, as clock speeds increase and interconnect delays become more important, mesochronous interconnect shows a great deal of promise. However. the advantages of any synchronization technique cannot be fully exploited without modifications at the architectural level.

References:-

[1] David G. Messerschmitt, “Synchronization in digital system design”, in IEEE journal on selected areas in communication. Vol. 8. No. 8. October 1990.

[2] Report on “Asynchronous design” ASYNC2000.

[3] ACid-WG trip report on ISSCC97 panel discussion.

[4] Miron Abramovici, Melvin A. Breuer, Arthur .D. Friedman, Chapter 9, “Digital System Testing and Testable Design”, Computer Science Press.

[5] William J. Dally and John W. Poulton “Digital systems engineering”, Cambridge University Press-1998.

[6] Ran Ginosar, “Fourteen Ways To Fool Your Synchronizer”, ASYNC 2003.

[7] M. Hardmian, L. A. Homak, T. E. Little, S. K. Tewkshury, and P. Franzon, Fundamental interconnection issues, AT&T Tech. J., vel. 66, p. 134. July 1987.

[8] D. G. Messerschmitt, Digital communication in VLSI design, In Proc. 23rd Asilomar Conf. Signals, Sysr. Con~put.. Oct. 1989.

[9] M. Hatamlan and G. L Cash, Parallel bit-level pipelined VLSI designs for high-speed signal processing, Proc. IEEE. kol. 75, p. 1192, Sept. 1987.

[10] M. Hatamian. Understanding clock skew in synchronous systems, in Concurrent Compuralions. S. C. Schwartz, Ed. New York.

[I1] T. Meng, R. W. Brodersen, and D. G. Messerschmitt, AsynchronPlenum, 1988. ous logic aynthesls for signal processing from high-level speclfications, IEEE ICCAD 87Dig. Tech. Papers, Nov. 1987.

[12] T. Meng, G. Jacobs, R. Brodersen, and D. Messenchmitt. lmplementation of high sampling rate adaptive filten using asynchronous deslgn techniques, In Proc. IEEE Workshop VLSI Signal Processing, Nov. 1988. 1121 T. Meng and D. G. Messerschmitt. Automatic synthesis of asynpur. Aide