

# Multi-GHz Systems Clocking

## Invited Paper

Vojin G. Oklobdzija, Fellow IEEE

Department of Electrical Engineering, University of California, Davis

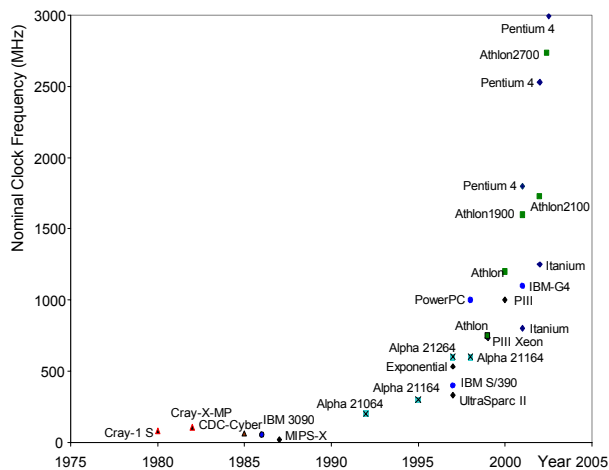
[vojin@ece.ucdavis.edu](mailto:vojin@ece.ucdavis.edu)

<http://www.ece.ucdavis.edu/acsel>

**Abstract:** Clocking, Synchronous Systems, Asynchronous Systems, Clock Uncertainties, Clocked Storage Elements, Finite-State Machine, Clock Distribution.

## 1. Introduction

The microprocessor clock speed has been rising rapidly; doubling every three years, Fig.1. The highest microprocessor clock frequency today is slightly above 3GHz while changing rapidly upward. It is expected that 10GHz clock rate will be reached in the next 5 years and by the year 2010 the processors will be running at frequencies beyond 10GHz. At that clock rate several challenges may force us to re-examine standard approaches to clocking.



**Fig. 1.** Clock frequency over the years for various representative machines and processors

As the clock speed increases, the number of logic levels in the *critical path* diminishes. In today's high-speed processors, instructions are executed in one-cycle, which is driven by a single-phase clock. In addition the pipeline depth is increasing to 15 or 20 stages in order to accommodate the clock frequency increase. Today 10 or less levels of logic in the critical path are not uncommon. Never the less, the amount of logic between the two stages is decreasing further. Thus any overhead associated with the clock system and clocking mechanism that is directly and adversely

affecting the machine performance is critically important.

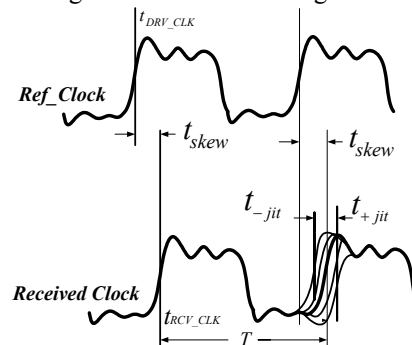
At today's frequencies ability to absorb clock skew and to use faster Clocked Storage Element (CSE), results in direct and significant performance improvement. Those improvements are very difficult to obtain through architectural techniques or micro-architecture level. As the clock frequency reaches 5-10GHz traditional clocking techniques will be stretching to their limits. New ideas and new ways of designing digital systems are required.

### 1.1 Clock Distribution

The two most important timing parameters affecting the clock signal are: *Clock Skew* and *Clock Jitter*:

*Clock Skew* is a spatial variation of the clock signal as distributed through the system. It is caused by the various RC characteristics of the clock paths to the various points in the system, as well as different loading of the clock signal at different points on the chip. Further we can distinguish *global clock skew* and *local clock skew*. Both of them are equally important in high-performance system design.

*Clock Jitter* is a temporal variation of the clock signal with regard to the reference transition (reference edge) of the clock signal as illustrated in Fig. 2.



**Fig. 2.** Clock Parameters: Period, Width, Clock Skew and Clock Jitter

Clock jitter represents edge-to-edge variation of the clock signal in time. As such, clock jitter can also be classified as: long-term jitter and edge-to-edge clock jitter, which defines clock signal variation between two consecutive clock edges. In the course of high-speed

logic design we are more concerned about edge-to-edge clock jitter because it is this phenomenon that affects the time available for the logic operation.

Typically the clock signal has to be distributed to several hundreds of thousands of the clocked storage elements. Therefore, the clock signal has the largest fan-out of any node in the design, which requires several levels of amplification. As a consequence, the clock system by itself can use up to 40-50% of the power of the entire VLSI chip [1,9]. In addition it must be assured that every clocked storage element receives the clock signal precisely at the same moment in time.

## 2. Clocking in Sequential Systems

Traditional view of the Finite State Machine (FSM) is represented by Huffman model consisting of a combinational logic (CL) and clocked storage elements (CSE). In this model, the next state, which is determined by the present state and the input (in case of Mealy machine), is stored into the CSE by the triggering mechanism of the clock (edge or level). Following this model we are used to thinking that the purpose of the CSE is to “hold” or “memorize” the state. This concept is further enforced by the Level Sensitive Scan Design (LSSD) methodology, which uses the storage elements to “scan-out” the state of the machine during the test and debug mode.

We would like to present a slightly different view. The purpose of CSE is to prevent the corruption of the next state as illustrated in Fig. 3.

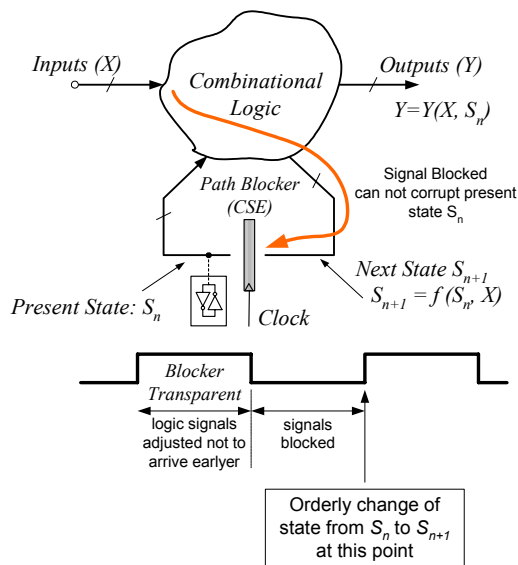


Fig. 3. Different view of FSM (Huffman model)

This model is broader and can represent wave pipelining [2], for example. In case of wave pipelining

signal is blocked from corrupting the present state  $S_n$  by a sheer delay of the wire. It simply cannot arrive in time, therefore no blocking is necessary. However, this model also reveals problems of wave pipelining technique. Ideally all the signals should arrive at the same point in time, which is not possible. Therefore fast-path problem becomes more difficult to control and much stringent requirements are necessary. Since a strict control of critical paths is not possible, after several cycles the system will run a danger of corrupting the state.

The case of skew tolerant domino logic [5,6], shown in Fig. 4, conforms to the model presented in Fig. 3.

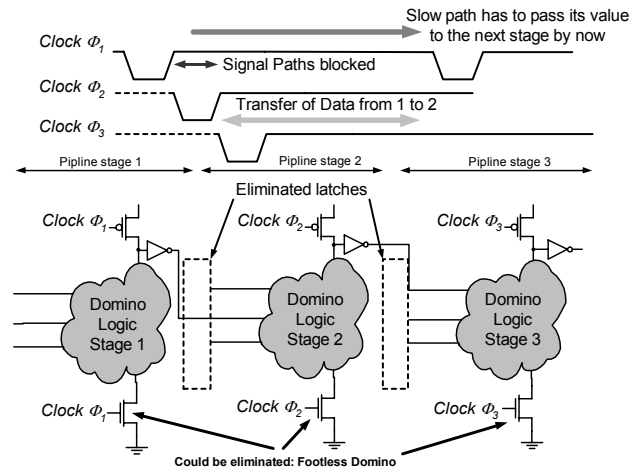


Fig. 4. Skew-tolerant Domino Logic: no explicit latches

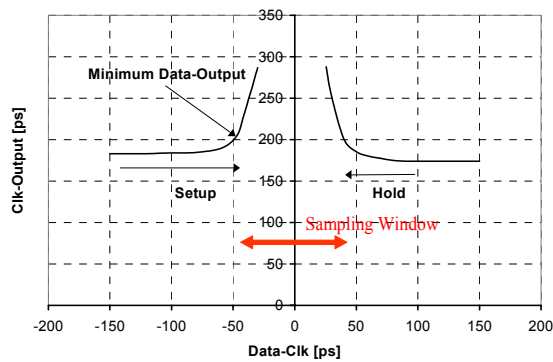
Blocking of signal is accomplished by the pre-charge phase of the clock. For example, while Clock  $\phi_2$  is “low” (pre-charge) data from Stage 1 can not be passed onto the Stage 2. Only after the pre-charge phase elapsed and Clock  $\phi_2$  has returned to “high” value can data from the Stage 1 be passed to Stage 2. This transfer has to be completed while the Clock  $\phi_1$  is “high”. Obviously the speed of this logic is determined by precise matching of the clocks. This is accomplished by having the clock signal travel along the data-path, while the local clocks are generated by delaying the clock for the amount of time needed in the logic stage. In a way this is similar to the clocking used in the early mainframe computers [3].

### 2.1. Clocked Storage Elements

The function of a *clocked storage element*: flip-flop or latch, is to block the signal path, thus preventing it from corrupting the present state. In addition it may be used to capture the state information and preserve it as long as it is needed by the digital system. It is not possible to define a storage element without defining its relationship to the *clock*.

*Timing Parameters*

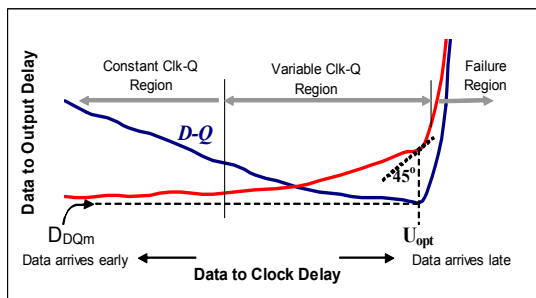
Data and Clock inputs of a clocked storage element need to satisfy basic timing restrictions to ensure correct operation of the flip-flop [4]. Fundamental timing constraints between data and clock inputs are quantified with *setup* and *hold* times, as illustrated in Fig. 5 [8]. Setup and hold times define time intervals during which input has to be stable to ensure correct flip-flop operation. The sum of setup and hold times define the “*sampling window*” of the clocked storage element. The “*sampling window*” is the time period in which clocked storage element is “*sampling*” and data is not allowed to change.



**Fig. 5.** Setup and Hold time behavior as a function of Clock-to-Output delay [8]

*Setup and Hold Time Properties*

Failure of the clocked storage element due to the Setup and Hold time violations is not an abrupt process. This failing behavior is shown in Fig. 6.



**Fig. 6.** Setup and Hold time behavior as a function of Data-to-Output delay

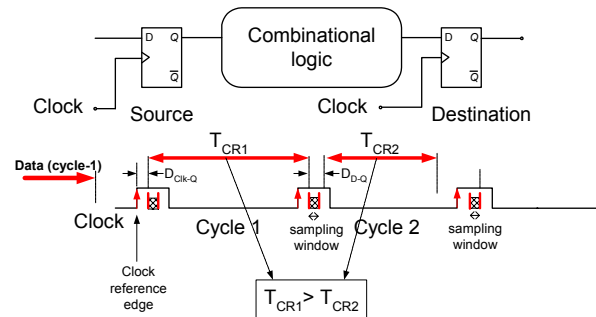
There are two opposing requirements with respect to the change of data signal as the locking event is approaching [8]:

- The change should be kept further from the failing region (Fig. 6) for the purpose of design reliability.
- The change should be as close as possible to the failing region, in order to increase the time available for the logic operation.

In industry, Setup and Hold times are specified as points in time when the Clk-Q ( $t_{CO}$ ) delay raises for an arbitrary number (commonly 5-20%). This reason is not valid. If we pay attention to D-Q ( $t_{DQ}$ ) delay (instead of Clk-Q), we see a different picture. In spite of the increase in Clk-Q delay, there are still benefits of getting closer to the locking event, because D-Q delay (representing the time taken from the cycle) is reduced [16].

**2.2 Time Borrowing and Absorption of Clock Uncertainties**

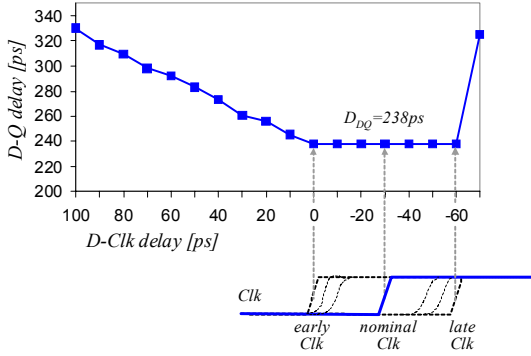
Even if data arrives past the clock edge, the delay contribution of the storage element is still smaller than the amount of delay passed onto the next cycle. This allows for more time for useful logic operation. This is known as: “*time borrowing*” or “*cycle stealing*” [7]. In order to understand the full effects of delayed data arrival we have to consider a pipelined design where the data captured in the first clock cycle is used as input in the next clock cycle as shown in Fig. 7.



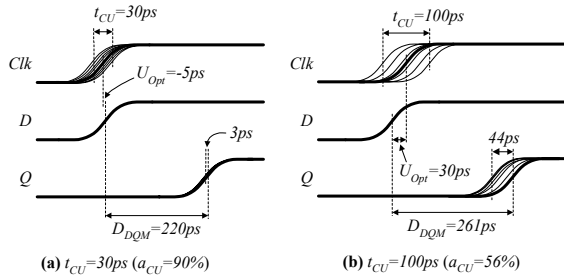
**Fig. 7.** “Time Borrowing” in a pipelined design

The “*sampling window*” is defined as the time period in which clocked storage element is “*sampling*” and data is not allowed to change. The amount of time for which the  $T_{CR1}$  was stretched did not come for free. It was simply taken away (“*stolen*” or “*borrowed*”) leaving less time in the next cycle (Cycle-2) for  $T_{CR2}$ . As a result of late data arrival in the *Cycle 1* there is less time available in the *Cycle 2*. Thus a boundary between pipeline stages is somewhat flexible. This feature not only helps accommodate a certain amount of imbalance between the critical paths in various pipeline stages, but it helps in absorbing the clock uncertainties: *skew* and *jitter*.

Thus, “*time borrowing*” is one of the most important characteristics of today’s high-speed digital systems. Absorption of the clock jitter is shown in Fig. 8 [7] and the effect on data arrival in the following cycle is illustrated in Fig.9. We observe how moderate amounts of clock uncertainties can be effectively absorbed, while the absorption property diminishes as the clock uncertainties become excessive.



**Fig. 8.** Clock Skew absorption property: Data-to-Output Delay versus Clock Arrival Time



**Fig. 9.** Effects of clock uncertainties to Data arrival in the next cycle [17]

The benefits of the “flat” Data-to-Output characteristic are seen in Figures 8 and 9. We create the “flat” characteristic by expanding the transparency window of the CSE. Widening of the transparency window is equivalent to increasing the separation between the two reference events in time: one that opens and other one that closes the CSE. In effect, the storage element behaves as a transparent latch for the short amount of time after the active clock edge [10]. Widening the transparency window can be achieved by intentionally creating wider capturing pulse of Flip-Flop and/or Pulsed Latch [11], or overlapping Master and Slave clocks in the Master-Slave Latch. A consequence of increasing the transparency window is that the failure region of Data-to-Output characteristic is moved away from the nominal clock edge. This results in the negative Setup Time but at the expense of increasing the Hold Time of the storage element. Large Hold Time makes fast path requirement harder to meet. Thus, the design for clock uncertainty absorption is often traded for longer Hold Time. In many cases, however, these two requirements are not contradictory, since different type of storage elements are used in fast and slow paths. The maximal clock skew that a system can tolerate is determined by clock storage elements. If the clock-to-output delay of a clocked storage element is shorter than the hold time required and there is no

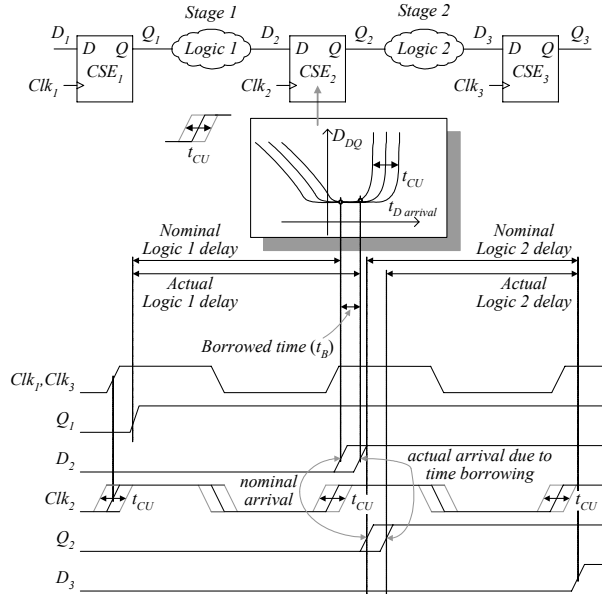
logic in between two storage elements, a race condition can occur. A *minimum delay restriction* on the clock-to-output delay is given by:

$$t_{CLK-Q} \geq t_{hold} + t_{skew}$$

If this relation is satisfied, the system is immune to hold time violations.

The clock uncertainty absorption property shows how the propagation delay of a CSE is changing if the arrival of the reference clock is uncertain. Applying the clock uncertainty to a CSE is equivalent to holding reference clock arrival fixed and allowing data arrival to change.

More generally, uncertainty absorption should be treated as degradation of Data-to-Output delay for *uncertain Data-to-Clock delay*. As such, it can be used to describe the timing of the CSE if used in time borrowing, in exactly the same way if used for clock uncertainty absorption. Therefore, a “soft clock edge” designates a storage element whose output follows both early and late arrivals of the input, allowing slower stages to borrow time from the faster subsequent stages.



**Fig. 10.** Time borrowing with uncertainty-absorbing clocked storage elements [17]

The time borrowing capability and the clock uncertainty absorption are not mutually exclusive. They can be traded-off for each other. Fig. 10 illustrates a case where a wide transparency window, denoted as a flat Data-to-Output characteristic, is used to absorb both, the clock uncertainties  $t_{CU}$  and to borrow time  $t_B$  from the surrounding stages. Combinational logic of stage 1 takes more time than nominally assigned, and it borrows a portion of the cycle time from stage 2. In general, the storage element

may not be completely transparent (i.e. Data-to-Output characteristics is not completely flat). The combination of clock uncertainty  $t_{CU}$  and time borrowing  $t_B$  causes an increase in the Data-to-Output delay of the Flip-Flop  $\Delta D_{DQ}$ .

It should be noted that the practical values of the total borrowed time are about the width of the transparency window of the storage element and in any event shorter than the Hold Time.

### 3. Asynchronous Systems

As the clock frequency increases, synchronous systems are facing serious problems such as the lack of ability to precisely control the clock, non-scaling clock uncertainties, wire delays and the simple fact that the signal may need one or more clock cycles to reach its destination. Thus, asynchronous system design has been revisited.

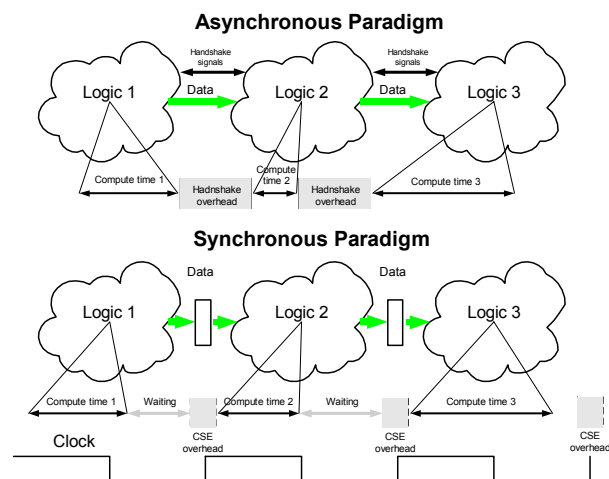


Fig. 11. Data transfer in Asynchronous System versus Synchronous [15]

The overhead imposed on the synchronous system by the clock uncertainties and Clocked Storage Element properties is simply traded for the overhead imposed by the handshake signaling in the asynchronous system, as shown in Fig. 11. Thus, the question really is which one of the two can be designed so that it imposes lesser penalties on the data transfer as the speed of the logic keep rapidly increasing. As of today it makes logical sense to use synchronous design in local domains, which can be clocked synchronously without considerable difficulties. Data transfer lasting several clock cycles could be accomplished using asynchronous communication. This opinion is supported by the fact that at 10GHz or more it would take several clock cycles to cross from one chip edge to another, as well as the fact that an entire processor in 1

billion transistor chip would occupy only a small portion of the chip.

### 4. Globally Asynchronous Locally Synchronous Systems

Following the industry projections VLSI chips will contain 1 billion transistors before the year 2010. However, the number of transistors used to build the logic of a single processor has not been increasing at the same rate. On the contrary that number is staying relatively constant. The Table 1. shows some of the transistor numbers for a sample of typical super-scalar RISC architecture processors around the year 2000.

Table 1: Transistor count in typical RISC processors

Feature	Digital 21164	MIPS 10000	Power PC620	HP 8000	Sun US
Freq. [MHz]	500	200	200	180	250
Pipeline Stg.	7	5-7	5	7-9	6-9
Issue Rate	4	4	4	4	4
Out-of-Ord.	6 loads	32	16	56	none
Reg-Ren./flp	none/8	32/32	8/8	56	none
Total Trans.	9.3M	5.9M	6.9M	3.9M	3.8M
<b>Logic Trans.</b>	<b>1.8M</b>	<b>2.3M</b>	<b>2.2M</b>	<b>3.9M</b>	<b>2.0M</b>

Fig. 12 provides an illustration of a 1billion transistor chip. If we project what would the speed of the chip be at that time and compare it with the projection for the decreasing speed of the interconnect, as the technology scales down, it becomes obvious that synchronous design on the entire chip may not be possible. Several clock cycles would be necessary for the signal to cross from one side of the chip to the other. From the design point, it is also obvious that the future 1 billion transistor chip will contain multiple cores in either: multi-processor or system-on-chip arrangement.

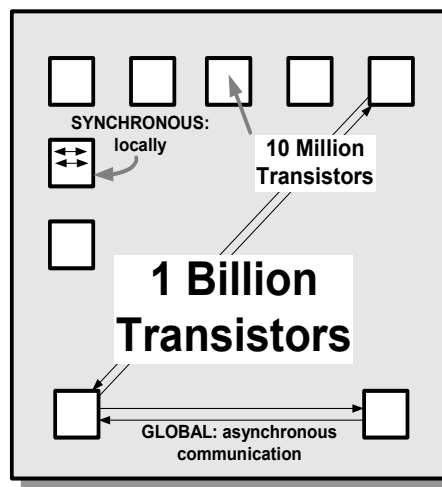


Fig. 12. Projection of a 1 billion transistor VLSI chip

Therefore Globally Asynchronous and Locally Synchronous clocking is considered as a promising technique for the System on a Chip (SOC) design.

In such a system a number of independently synchronized modules communicate between each other using asynchronous communication mechanism. It is projected that interconnect effects are to be manageable within a local synchronous module; therefore, a synchronous design would continue to be a viable option for the processor core.

The main feature of these systems is the absence of a global timing reference. Synchronous modules use distinct local clocks, or clock domains, even running at different frequencies.

This methodology is viable when various Intellectual Property (IP) blocks are integrated in a single chip in a System on a Chip (SOC) environment, since proven IP blocks can be reused without any modifications while relying on asynchronous interface between blocks. Such design is able to keep the benefit of synchronous design while avoiding problems caused by global wiring, especially a global clock signal.

Most conventional microprocessor IPs are synchronous in their construction. On the other hand, fully asynchronous designs are built using self-timed circuits without having any global timing reference.

Basically, there is no novelty in the Globally Asynchronous, Locally Synchronous design, given that such concept has been long used in the mainframe computer systems and this represents its logical migration on the VLSI chip.

## 5. Conclusion

Clocking for high performance and low-power systems represent is a challenge given the rapid increase in clock frequency which has already reached multiple GHz rates. We expect that current clocking techniques will hold up to 10 GHz. Afterwards the pipeline boundaries will start to vanish while more exotic clocking techniques will find their use. Synchronous design will be possible only in limited domains on the chip. A mix of Synchronous and Asynchronous design may emerge even in digital logic. This may represent the next design challenge in complex chip design.

## References

- [1] Gronowski P.E, et al, "High-performance microprocessor design" Solid-State Circuits, IEEE Journal of, Volume: 33 Issue: 5, May 1998.
- [2] Wayne P. Burlison, Maciej Ciesielski, Fabian Klass, Wentai Liu, "Wave-Pipelining: A Tutorial and Research Survey", IEEE Transaction of Very Large Scale Integration (VLSI) Systems, Vol. 6, No. 3, September 1998.
- [3] Cotten L. W, "Circuit Implementation of High-Speed Pipeline Systems", AFIPS Proceedings, Fall Joint Comput. Conf., pp. 489-504, 1965.
- [4] Unger S.H, Tan CJ, (1986) "Clocking Schemes for High-Speed Digital Systems", IEEE Transactions on Computers, Vol. C-35, No 10, October 1986.
- [5] D. Harris, M. A. Horowitz, "Skew-Tolerant Domino Circuits", IEEE Journal of Solid-State Circuits, Vol. 32, No. 11, November 1997.
- [6] D. Harris, et al, "Opportunistic Time-Borrowing Domino Logic, US Patent No. 5,517,136, Issued May 14, 1996.
- [7] Partovi, H. et al, "Flow-through latch and edge-triggered flip-flop hybrid elements", 1996 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, ISSCC, San Francisco, February 8-10.
- [8] V.Stojanovic and V.G. Oklobdzija, "Comparative Analysis of Master-Slave Latches and Flip-Flops for High-Performance and Low-Power VLSI Systems," IEEE Journal of Solid-State Circuits, Vol.34, No.4, April 1999.
- [9] T. Kuroda and T. Sakurai "Overview of Low-Power VLSI Circuit Techniques", IEICE Trans. Electronics, E78-C, No 4, April 1995, pp.334-344, INVITED PAPER, Special Issue on Low-Voltage Low-Power Integrated Circuits.
- [10] F. Klass et al, "A New Family of Semidynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors", IEEE Journal of Solid-State Circuits, vol. 34, no. 5, pp. 712-716, May 1999.
- [11] Tschanz James, Siva Narendra, Zhanping Chen, Shekhar Borkar, Manoj Sachdev, Vivek De, "Comparative Delay and Energy of Single Edge-Triggered & Dual Edge-Triggered Pulsed Flip-Flops for High-Performance Microprocessors, Proceedings of the 2001 International Symposium on Low Power Electronics and Design, Huntington Beach, California, August 6-7, 2001.
- [12] Hauck Scott, "Asynchronous design methodologies: an overview", Proceedings of the IEEE, Volume: 83 Issue: 1, Jan. 1995.
- [13] E. Sutherland, "Micropipelines," Communications of the ACM, June 1989.
- [14] Hemani, A., Meincke, T.; Kumar, S.; Postula, A., Olsson, T., Nilsson, P., Oberg, J. Ellervee, P., Lundqvist, D., "Lowering power consumption in clock by using globally asynchronous locally synchronous design style", Proceedings of the 36th Design Automation Conference, 21-25 June 1999.
- [15] V. G. Oklobdzija, Jens Sparso, "Future Directions in Clocking Multi-GHz Systems" Invited presentation, International Symposium on Low-Power Electronics and Design, Monterey, California, August 12-14, 2002.
- [16] Oklobdzija V.G, (ed.) "High-Performance System Design: Circuits and Logic", Book, IEEE Press, July 1999.
- [17] Oklobdzija V.G, Stojanovic V, Markovic D, Nedovic N, "Digital System Clocking: High-Performance and Low-Power Aspects, J. Wiley, January 2003.