Globally Asynchronous Locally Synchronous Wrapper Circuit based on Clock Gating

Esmail Amini Mehrdad Najibi Hossein Pedram
IT and Computer Engineering Department, Amirkabir University of Technology, Tehran, IRAN.
{ es_amini, najibi, pedram@ce.aut.ac.ir }

Abstract

In this paper we propose an asynchronous wrapper with new asynchronous communication port controllers and reliable clock generation scheme for locally synchronous modules. This is achieved by utilizing clock gating idea within GALS wrappers which makes the use of reliable and robust off-chip clock generator possible for locally synchronous modules. In addition to clock robustness, the clock generator part becomes totally synchronous. To validate the proposed solution, we employed the wrapper circuit in Viterbi error detection and correction circuit. The synthesis results show that our GALS approach gains 44%~48% performance improvement in contrast to pausible clock GALS wrappers.

1. Introduction

By technology improvements and increasing complexity of VLSI systems, pure synchronous methodology doesn't look much promising to handle all design requirements. As a well-known example with increasing die size and clock frequency the clock skew problem is getting much troublesome. On the other hand, power consumption is getting even more important design consideration and large clock distribution network is one of the major sources of power consumption [1].

Asynchronous methodology is shown to be a good candidate solution which can solve nearly all the problems arising form the clock. While Asynchronous circuits potentially can have better performance, lower power consumption and more robustness, they can be larger due to handshaking overheads and they also suffer from design complexity and lack of automated CAD tools.

Globally Asynchronous Locally Synchronous or in abbreviate GALS methodology [2] is an intermediate solution that combines benefits of both synchronous and asynchronous methodologies.

The general architecture of pausible clock based GALS module is shown in figure 1. In GALS Systems, synchronous modules are surrounded by asynchronous wrappers. In other words, GALS modules communicate with each other inside an asynchronous medium. A GALS module consists of a Locally Synchronous (LS) module, a clock generator and asynchronous port controllers. Asynchronous port controllers intervene between synchronous and asynchronous environments. Local clock generator is used to generate LS module clock signal.

Most contemporary GALS methodologies use on-chip clock generators to feed locally synchronous (LS) modules. Most of the proposed on-chip clock generators for GALS systems are based on pausible clocking which is based on on-chip ring oscillators [2][3][4][5][6]. Since on-chip ring oscillators is quite dependent to process variations, the generated clock signal will be unstable [7][8]. Therefore we should consider a marginal penalty in clock frequency and in result lower performance will be expected.

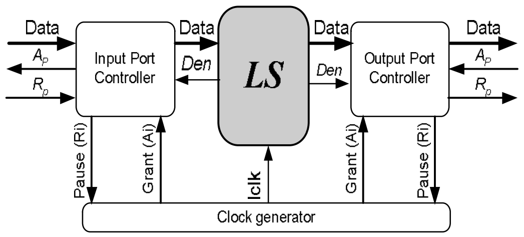


Figure 1- Pausible clock based GALS wrapper

When LS module needs data, it must inform the related port controller by toggling *Den* signal at the falling edge of the current clock pulse. Then port controller generates a request signal (R_i) for clock generator circuit which is responsible for synchronizing the LS module with incoming asynchronous data and stops the local clock. Afterward clock generator replies by an acknowledge signal (A_i) and stretches the low phase of the clock if needed. This prevents metastablity during asynchronous communication. The *Den* signal is activated on the falling edge of clock because using left edge of the clock leaves the controller of the LS module unchanged [8]. After data communication, the clock will be released. These sequences of events are shown in figure 2.

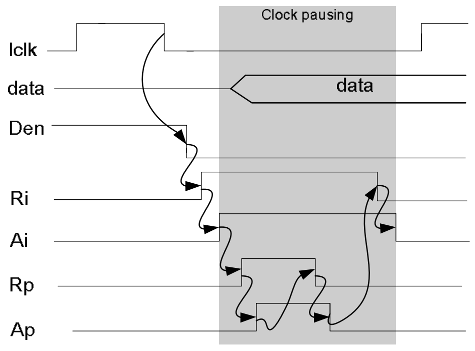


Figure 2- Timing diagram of clock pausing

The STG diagrams of pausible clock based GALS port controllers could be found in [4][8].
The circuit in figure 3 is a typical implementation of pausible clock generator [4].

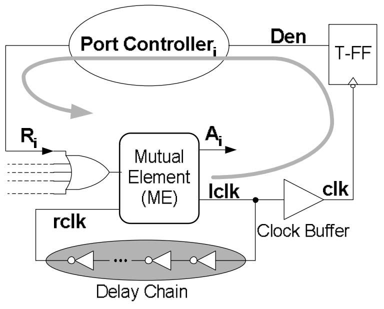


Figure 3- Pausible clock generation

The OR gate in figure 3 has N input for N port controllers inside wrapper. The Mutual Exclusion (ME) element in figure 3, arbitrates between R_i and *rclk* signals. If incoming data arrives too close to sampling clock edge, either the clock edge or data transfer is shifted to a later point in time [4]. It must be also taken into consideration that when the clock generator stops the clock, there may be still clock edges in the buffer tree and therefore, the delay of the chain must be longer than the delay of clock buffer and tree [8]. Also the delay of the port controller (T_{pc}) and delay of the OR gate (T_{or}) has much effect on LS modules frequency.
At this article a new method for generating local clock pulse is introduced that is based on clock gating. The clock gating is also used as a technique to reduce the power consumption in totally synchronous systems. However in totally synchronous systems, gating the clock would produce problems itself because it can worsen the clock skew problem. Because of the simplifying skew nature of GALS methodology, clock gating is suitable in designing GALS systems. We applied clock gating idea in GALS wrapper circuit and compared it with pausible clock method introduced in [4]. There are few works done related to the clock gating the only notable work on using of clock gating method in GALS systems without any details and results has mentioned in [9].
The general architecture of the GALS wrapper circuit based on clock gating is shown in figure 4. In proposed wrapper circuit, each locally synchronous module has a local clock which is obtained by gating separated external clock (*eclk*) signal regarding to requests which comes from port controllers. When locally synchronous module enters data communication phase, it informs the related port controller that it needs data communication. Afterward port controller will generate a gate request for clock generator and the external clock will be gated. After handshake completion, the clock will be released.
Gated clock based GALS scheme has no major impact on the interface of LS module as it is clear by comparing Figure 1 and Figure 4. Thus the same LS modules can be used in both pausible and gated clock based GALS systems.

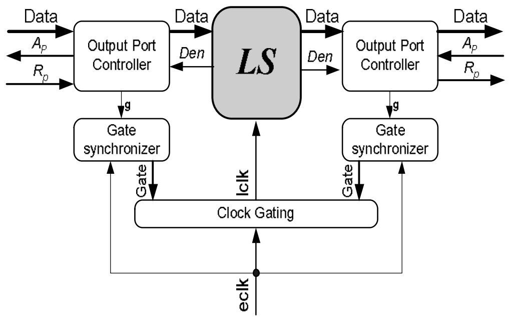


Figure 4- Gated clock based GALS wrapper

Because no handshaking is required between asynchronous port controllers and local clock generation circuit, the port controllers are simpler than pausible clock based port controllers. While simpler port controller will able us to utilize faster LS modules, it requires more considerations; Since there is no acknowledge signal which indicates proper gating of the clock, it is needed for the synthesis tool to guarantee that the gate request always reaches the clock generator with a reliable delay margin. It can be done by enforcing some timing constraints on the path from Den to the gate signal within each GALS module.
In section 2 we introduced wrapper circuit in more details. Viterbi Error Detection and Correction GALS implementation as our benchmark is introduced in section 3. In section 4 we have shown the results of

implemented circuits and final conclusion of the outcomes of this research is provided in Section 5.

2. Wrapper implementation

As stated before, each galsified block consists of a synchronous module, a clock generator and port controllers to communicate with asynchronous environment. Various parts of GALS wrapper shown in figure 4 is introduced in the following subsections.

2-1. Clock generation

The gated clock based clock generation circuit in GALS wrapper is as simple as usual clock gating as shown in figure 5. There is no asynchronous element in clock generator circuit. This simplifies implementing clock generator circuit.

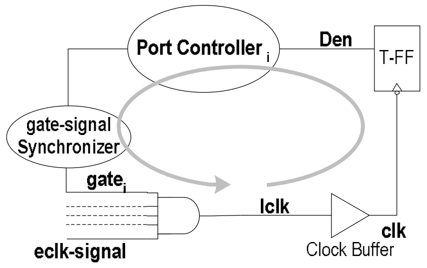


Figure 5- Clock gating in gated clock GALS wrapper

During the activity phase of LS module all of the gate-signals are high. When LS module enters data communication phase, one or more of the gate-signals go low and clock will be gated.
The period of the local clock is a multiple of external clock signal period because the gating and releasing of the clock is done when external clock signal is in transition.
As pausible clock based GALS scheme, the overall probability of stopping the clock relates to both the number of port controllers (*gate* signals) and *gate* signals overlapping. Also the propagation delay from *eclk-signal* to *gate* signal has to be smaller than the low phase of *eclk-signal*. This path consists of clock tree buffers, a T-flip flop, the port controller, gate signal synchronizer circuit and an AND gate as shown in figure 5. The AND gate has N+1 input where N is the number of port controllers inside wrapper. So by growing N, it takes more time to produce *gate* signal.
The larger the clock tree latency, the less time is available for generating *gate* signal. This scenario exists in pausible clock generation too. So for comparing two methods we will ignore the clock tree latency.

2-2. Port controllers

Each port controller is activated by Den signal which is generated by LS module. When LS module needs data for next clock cycle,it activates Den signal at the coming negative edge of the current clock pulse like pausible clock based GALS scheme.
After Den signal activation, the port controller starts its work. At the first step, external clock will be gated to prevent metastablity during asynchronous data exchange. The flow of signal events to gate the eclk-signal is shown in figure 6.

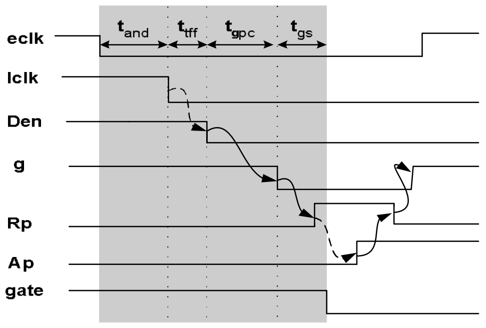


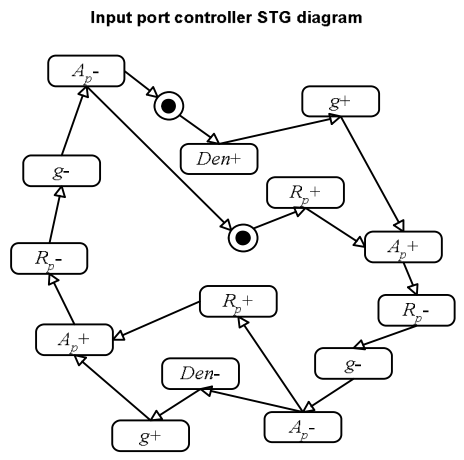
Figure 6- Timing diagram of external clock gating

In figure 6 the meaning of indicated times are:
- $-t_{and}$: delay of the AND gate in figure 5.
- $-t_{tff}$: the delay of T-flip flop to toggle Den-signal.
- $-t_{pc}$: delay of port controller
- $-t_{gs}$: delay of gate synchronizer circuit

The *gate* signal must be generated before the rising edge of *eclk-signal* to prevent metastablity problem during asynchronous communication and ensure that incorrect data not being processed by LS module. Thus the gray rectangle in figure 6 should not exceed the incoming rise edge of eclk-signal. In the other words we must guarantee $t_{and}+t_{tff}+t_{gpc}+t_{gs}<t_{eclk}/2$.
The STG diagram of the asynchronous port controller in clock gating wrapper circuit is mentioned in figure7. Petrify [10] is used for synthesis of the port controllers and the synthesis results for port controllers for tsmc018 library cells are shown in figure 8.

2-3. Gate synchronizer

Gating the eclk-signal must not be done later than the next positive clock edge. As stated before, this can be guaranteed by enforcing timing constraints on GALS modules during synthesis process.
Releasing the clock in the high phase of external clock will decrease the clock period (figure 9) so it must be avoided. To avoid this, the gate synchronizer circuit is used to generate the final gate-signal from port controller's output (g-signal) and eclk (figure 10). This prevents any spike or clock period reduction.
In the circuit in figure 10, gating will be done as soon as g-signal goes high. It is done by lowering the gate-signal so the external clock will be gated.



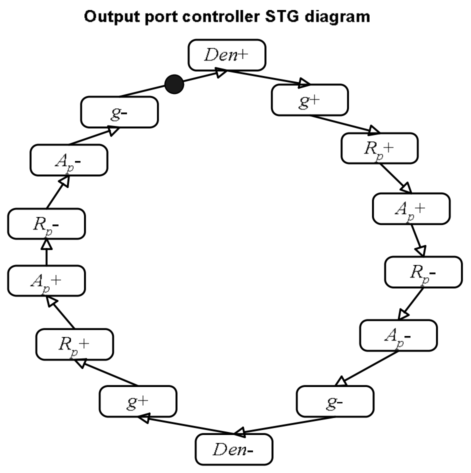


Figure 7- STG diagram of the port controllers

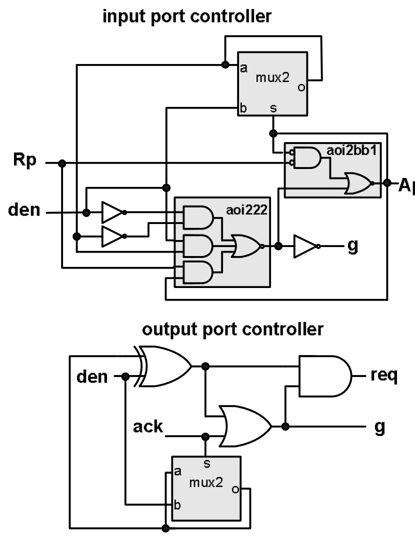


Figure 8- Gated clock based Port controllers

The time for the gate signal to stop the eclk signal is depicted in figure 10, by solid Line (T3—T5—T8).

The clock will be released when both g-signal and eclk-signal are in low state. It takes the normal time to release the eclk dedicated by dashed line in figure10 (T2—T6—T7).

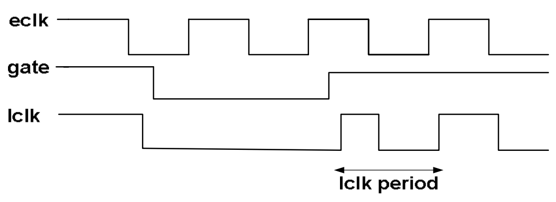


Figure 9- Invalid time for clock releasing

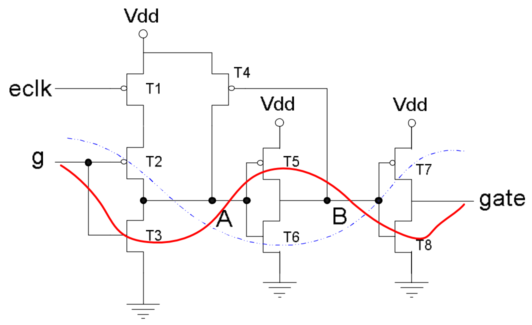


Figure 10- Gate-signal synchronizer

The releasing scenario is as following:
Suppose that the external clock is gated. Thus the value of A-signal is low. Depending on eclk-signal value, one of the following situations can occur:

- *eclk-signal* is high: in this situation A is not connected directly to Vdd or GND until the *eclk-signal* changes its value; A remains low and in result *gate-signal* will remain low. When the *eclk-signal* becomes low, *A-signal* goes high and clock will be released. During this state transistor T4 will keep the A's value.
- *eclk-signal* is low: In this situation *A-signal* goes high, in result *gate-signal* goes high and clock will be released. While clock is in released mode, transistor T4 will charge A's value.
- *eclk-signal* is in transition mode: In this situation transistor T4 will prevent node A from getting charged to an intermediate value because regarding to inverter N1, *B-signal* could have one of the following values:
  - The first inverter gate (T5 and T6) assumes *A-signal* is low. So *B-signal* is high, afterward *gate-signal* goes low and clock remains gated.
  - The first inverter assumes *A-signal* is high. So *B-signal* goes low and *gate-signal* becomes high meanwhile T4 charges *A-signal* to high.
  - Depend on A's value, both T5 and T6 may be ON. Thus *B-signal* would be between low and high. This state is unstable because transistor T4

will charge *A-signal*. So clock will be released eventually. In this situation, the releasing path of *eclk-signal* is T2—T5—T4—T6—T7.

This analysis shows that the clock will be gated and released safely in all situations, and we do not face the metastablity problem. This is mainly correct because the local clock will be always in its low phase during asynchronous data communication.

3. GALS Implementation of Viterbi Circuit

As of benchmarking, Viterbi error detection and correction circuit is used. In Viterbi (2,1,6) circuit for each cycle, a 2-bit input stream is entered and 1-bit output is generated. For each input stream, the cost of each branch metric (BM) cost is calculated by branch metric unit (BMU). After calculating BM, the total path metric (PM) cost is calculated by path metric unit (PMU). We keep the PM costs in Survivor Path Storage. The Trace Back Unit reconstructs the true path from the data in Trace Back Unit and sends the result to shift register. Shift Register gets the 32-bit input vector and generates 32 single bit output signals. The details of the implementation of the Viterbi could be found in [11]. The Structure of Viterbi GALS Implementation is shown in figure 11.

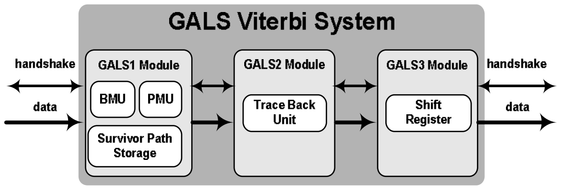


Figure 11- Viterbi GALS block diagram

The bottleneck of Viterbi implementation is the Trace Back Unit. Fortunately Trace Back Unit is not a busy module and is off in most of times. If we encapsulate BMU, PMU and Survivor Path Storage by a single GALS wrapper, Trace Back Unit in a separate GALS wrapper and shift register in another wrapper, we could expect power saving and performance improvement in contrast to totally synchronous implementation.

4. Results

For synthesizing port controllers in both pausible and gated clock based GALS schemes, we must prevent synthesis tool from optimizing port controllers, because Petrify resulted circuits must be instantiated exactly in wrapper circuit to guarantee speed independent nature of generated circuits. After acquiring timing reports of synthesis tool, we simulated synthesized port controllers. The delays of asynchronous port controllers for both pausible clock based wrapper circuit and gated clock based wrapper circuit, synthesized in 0.18µ technology are shown in Table 1. To estimate the delay of gate-signal synchronizer circuit, we simulated the gate synchronizer circuit in 0.18µ technology. GM1, GM2 and GM3 modules in Viterbi GALS system (figure 11) each need one input port controller and one output port controller for asynchronous data communication. By the way, the AND gate in figure 5 has 3 inputs.

Table 1- pausible clock based wrapper port delays

Wrapper component		Delay
Pausible clock based wrapper	*Input port latency* (T_{ppc}+ T_{or})	210
	Output port latency(T_{ppc}+T_{or})	410
Gated clock based wrapper	*Input port latency (Tgpc)*	112
	Output port latency (Tgpc)	200
	gate-signal synchronizer clock gating (Tgs)	90 ps
	AND gate delay (Tand)	90 ps

In pausible clock based GALS wrapper, the time it takes to pause the clock is 0.41 ns. In gated clock GALS wrapper, the time it takes to gate the clock is 0.2 ns. By using of gated clock based GALS wrapper, the minimum low phase duration of clock-signal is t_{and}+t_{gpc}+t_{gs}=0.38 ns. So in gated clock GALS Systems, LS modules can operate in higher frequencies.

After synthesizing all Viterbi modules in 0.18µ technology for the totally synchronous circuit the maximum clock frequency was reported as 166 MHz. The Viterbi GALS implementation results for locally synchronous modules mentioned as follow:

Table 2- GALS Viterbi Implementation Results

Module	clock Frequency	Activation
GM1 LS module	256 MHz	32
GM2 LS module	153 MHz	1
GM3 LS module	555 MHz	32

The LS modules can operate up to the frequencies shown in Table1. Note that these results are valid both for clock pausing and clock gating schemes, so the analysis of both systems can be done in the same way; If we ignore the environment delays due to the special structure of the circuit, pausing or gating of the clock-signal and also releasing it occurs before the rising edge of clock pulse of Viterbi GALS modules. In other words no clock pausing or gating will occur in case of the dominance of GM1 module in Viterbi GALS system because of its operational duration. Thus the frequency of the implemented Viterbi GALS system is 256 MHz. So the maximum frequency of GM1 module is 54% higher than totally synchronous Viterbi circuit frequency. By the way, implementing GALS methodology has performance yield of 0.54 in Viterbi error detection and correction system.

There are some performance variations between pausible and gated clock method regarding to the difference in their clock generator behavior.

In pausible clock based GALS scheme we generate the clock signal from chain of inverters and the delay of the chain specifies the generated clock frequency (figure 3). To calculate the number of inverters that can guarantee the critical path timing requirements of the circuit for pausible clock scheme, we have to use fast model for chained inverters, while using typical model for the circuits themselves. The clock gated system can take advantage of eliminating these margins and can result in notable performance improvement. The fallowing typical analysis shows the order of the possible performance improvement in 0.18µ technology. The chain of inverters post synthesis results can be seen in table 3 for both fast and typical delay models.

Table 3- Modules frequencies in process modes

Circuit		GM1	GM2	GM3
Pausible clock based Wrapper	*Fast mode frequency*	254 MHz	150 MHz	554 MHz
	Typical mode frequency	177 MHz	103 MHz	385 MHz
Gated clock based wrapper		256 MHz	153 MHz	555 MHz

By using gated clock scheme, an external clock can be used with least amount of frequency variation. The LS modules in gated clock based GALS scheme have a performance improvement of 44%~48% in contrast to LS modules in pausible clock based GALS scheme as can be computed from table 3. This performance improvement as stated before is due to the elimination of the extra clock margins required for pausible clocking. So in gated clock based GALS scheme 44%~48% performance can be achieved, moving from fast to typical operation models in 0.18µ technology.

5. Conclusion and Future Work

GALS methodology smoothes clock skew problems and utilizes the benefits of both synchronous and asynchronous design. While using on chip pausible clock generators in GALS wrapper circuits, leads to a tradeoff between robustness and performance, we have shown that both performance and robustness can be improved using the gated clock scheme.

Gated clock GALS systems combines the benefits of GALS design and clock gating idea. The new GALS wrapper circuit has an external clock input that is gated whenever asynchronous communication is needed. Better stability and robustness in our clock generator circuit is the direct consequence of using an external more robust, stable and reliable clock generator. By using more stable clock signal in gated clock scheme, we would have in turn improvements in performance. Also converting the pausible clock GALS implementations to gated clock GALS implementation, is as simple as replacing port controllers and clock generator with proposed circuits and the LS modules need no change.

Meanwhile gated clock GALS systems if can be implemented as Multi FPGA can present more benefits in prototyping and less design time, it requires the asynchronous port controllers to be mapped into FPGA LUT's which is our future target.

6. References

[1] A. Hemani, T. Meincke, et al. Lowering power consumption in clock by using globally asynchronous, locally synchronous design style. *In Proc. ACM/IEEE Design Automation Conference*, pp. 873-878, June 1999.

[2] D.M. Chapiro. Globally-Asynchronous Locally-Synchronous Systems. *PhD thesis*, Stanford University, 1984.

[3] David S. Bormann and Peter Y.K. Cheung. Asynchronous wrapper for heterogeneous systems. *In Proc.International Conf. Computer Design (ICCD)*, October 1997.

[4] J. Muttersbach, T. Villiger, and W. Fichtner. Practical Design of Globally-Asynchronous Locally-Synchronous Systems. *In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 2000.

[5] K.Y. Yun and A. E. Dooply, Pausible clocking based heterogeneous systems. *IEEE Transactions on VLSI Systems*, December 1999.

[6] S. Moore, G. Taylor, R. Mullins and P. Robinson. Point to point GALS interconnect. *In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 69-75, April 2002.

[7] Charles L. Seitz. System timing. In Carver A. Mead and Lynn A. Conway, editors, Introduction to VLSI Systems, chapter 7. Addison-Wesley, 1980.

[8] K. Saleh, M. Najibi, M. Naderi, H. Pedram, M. Sedighi, A Novel Clock Generation Scheme for Globally Asynchronous Locally Synchronous Systems: An FPGA-Validated Approach, *Proceedings of the 15th GLSVLSI ,Chicago*, April 2005.

[9] M. Singh, M. Theobald. Generalized Latency-Insensitive Systems for GALS Architectures. Proc. of the Workshop on Formal Methods for Globally Asynchronous Locally Synchronous (GALS) Architecture (FMGALS-03), held in conjunction with the 12th International Formal Methods. Europe Symposium, Pisa, Italy, September 2003.

[10] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. In *IEICE Trans. on Information and Systems,* pp. 315-325. March 1997.

[11] K.Saleh. Viterbi Hardware Implementation.Amirkabir University of Technology. Internal Report. June 2004.