Low Power GALS Interface Implementation with Stretchable Clocking Scheme

Anju C and Kirti S Pande

Department of ECE, Amrita Vishwa Vidyapeetham, Amrita School of Engineering
Bangalore, Karnataka, India

Abstract

Complex SoC imply the seamless integration of numerous IPs performing different functions and operating at different clock frequencies. The integration of several heterogeneous components into a single system gives rise to new challenges. Major issue includes controlling the clock frequencies of the different modules. As chips become faster and larger, designers face significant challenges including global clock distribution and power dissipation.

In-order to achieve global synchronization with high performance and low power conception globally asynchronous locally synchronous (GALS) method is used. In GALS, local modules can operate with their own clock and the entire module is communicating asynchronously. In this paper we implemented a low power GALS interface with stretchable clocking scheme in verilog HDL and compare the dynamic power of the interface with and without stretchable clocking with Synopsys Design Compiler.

Keywords: *Globally asynchronous locally synchronous, System on Chip, Stretchable clocking scheme, Asynchronous wrapper*

1. Introduction

Synchronous circuits use a central clock distribution for its operation. As speed increased, distributing the timing signals has become more and more difficult. These problems may be worse on large soc design. System complexity and demand for higher processing frequencies revealed several inherent problems with synchronous design styles. In synchronous circuit, the clock's rhythm must be slow enough to accommodate the slowest action in the chip's circuit. If it takes a billionth of a second for one circuit to complete its operation, the chip cannot run faster than one gigahertz. Even though many other circuits on that chip may be able to complete their operations in less time, these circuits must wait until the clock ticks again before proceeding to the next logical step. Whereas in asynchronous system, each part can operate with its own speed independent of the rhythm of the central clock. Asynchronous circuits communicate via handshakes. The handshake consists of a series of signal events send back and forth between the communicating elements. However because of high implementation costs and difficulties asynchronous circuit is seldom used on digital system design.

To integrate both the advantages of synchronous and asynchronous circuits, the globally asynchronous locally synchronous (GALS) methodology was proposed. The GALS provides a reliable communication between different modules in a SoC. GALS provide global synchronization with high performance and low power conception. In GALS locally synchronous modules are wrapped around an asynchronous wrapper which provides asynchronous communication between the modules. Stretchable clocking scheme stretches the clock whenever required; otherwise it will release the clock. This method helps to reduce the dynamic power of the entire module.

The outline of the paper is as follows: related works is reviewed in section 2. Design approach and the main architecture are described in section 3. Implementation details are presented in section 4. Section 5 discusses the simulation results obtained for the different modules of the system. Conclusions are drawn in the section 6.

2. Related Works

A.Device and S.M.Nowick provide an introduction to asynchronous digital circuit design [1]. Where they compare synchronous and asynchronous design methodologies and provide an idea about different classes of asynchronous circuits. Synchronous circuits are based on global clock distribution, this clock must be carefully controlled since it must be globally distributed, this often

provides to be difficult. Different control signals are used for the proper operation of the asynchronous circuits.

They also discuss about different classes of asynchronous circuits. A delay-insensitive (DI) circuit is one which is designed to operate correctly regardless of the delays on its gates and wires. That is an unbounded gate and wire delay is assumed. A quasi delay-insensitive (QDI) circuit is delay-insensitive except that isochronic forks are required. An isochronic fork is a froked wire where all branches have exactly the same delay. A self timed circuit contains a group of self timed elements. Each element contained in an equi-potential region, where wires have negligible or well bounded delay.

Yun and Donohue describe a novel communication scheme that is pausible clocking which can be used in heterogeneous systems [2]. In this scheme, communication between every pair of modules is done through an asynchronous FIFO channel; communication between a module and the FIFO is done using a request/acknowledge handshaking. The local clock built out of a ring oscillator is paused or stretched, if necessary, to ensure that the handshaking signal satisfies setup and hold time constraints with respect to the local clock.

The upper bound on the local clock rate is due to the ring oscillator, not the pausible clocking control. However tuning the ring oscillator frequency would require more control pins and hence are more expensive. Furthermore, anything in the clock path designed to generate multiple frequencies and/or to minimize jitter creates a problem for pausible clocking. Its main drawback is that a meta-stability state could occur when the request and the rising edges of the clock arrive simultaneously.

Through “Practical design of globally asynchronous locally synchronous systems” J.Muttersbach, T.Villiger and W.Fichtner [3] describe a complete design methodology for a globally asynchronous on-chip communication network connecting both locally-synchronous and asynchronous modules. Synchronous modules are equipped with asynchronous wrappers which adapt their interfaces to the self-timed environment and prevent meta-stability.

Equipping LS modules with an asynchronous interface adds a high degree of modularity. Surrounding a synchronous module with an asynchronous wrapper aims at making its external interface completely asynchronous. Each data vector entering or leaving the module is accompanied by a request-acknowledge pair of handshake signals (bundled data). To signal validity of data they use a four phase protocol.

S.Zhuang, W.Li, J.Carlsson, K.Palmkvist, L.Wanhammar propose an asynchronous wrapper with novel handshake circuits for data communication to be used in GALS systems [4]. The handshake circuits include two communication ports and a local clock controller. The paper presents two approaches for the implementation of communication ports; one with pure standard cells and the other with Muller C elements. The interface circuits with Muller C elements can operate at higher frequency than those with standard cells.

Y.T.Chang, W.C.Chen, H.Y.Tsai, W. M. Cheng, C. J. Chen, F. C. Cheng introduce a GALS interface scheme using stretchable clocking scheme [5]. The design operates correctly when the sender and the receiver work in different clock rates. While this implementation reduces latency, it has complex design. However they did not use any FIFO to store the data so there is a chance for data loss if the receiver clock rate is less than that of the sender.

In this paper we proposed a new GALS interface using the stretchable clocking scheme. This design consists of an asynchronous wrapper module which wraps the locally synchronous module. Operation of the locally synchronous module is based on its own clock frequency whereas the asynchronous wrappers communicate each other based on the handshaking protocols.

3. Design Approach

3.1 GALS Interface

In a large SoC, each components or IPs may be designed by different teams or even different companies. Integrating them on a single die may be a very difficult job. The most important reason is that these different designs have to operate correctly in different clock frequencies.

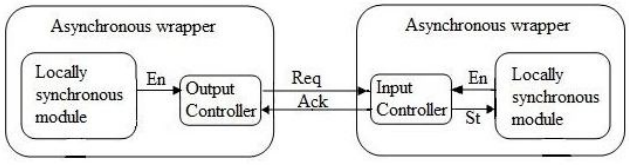


Fig. 1 GALS interface

The GALS methodology tries to wrap the synchronous circuit with asynchronous wrapper which is shown in

Fig.1. Thus, the whole system can communicate through asynchronous channels, while each local circuit can operate in their own local clock

3.2 STG of the proposed scheme

STG of the GALS system is shown in Fig. 2. It is based on a stretchable clocking method. Here *st* signal is used to stretch the clock during data transfer. *wr*+ means that the LS module is ready to accept data and will set *st* signal high (*st*+). Whenever *st*+ occurs (*st* signal become logic 1) stretchable clock generator stretches the clock. After data transmission *st* signal will set to *st*- (*st* signal become logic 0).

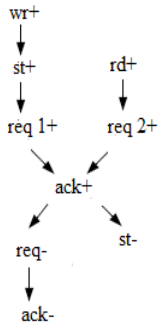


Fig. 2 STG of the GALS interface

3.3 Stretchable clocking scheme

In plausible clocking method there is a chance for meta-stability when request and rising edge of the clock arrives simultaneously. We can eliminate that problem by using stretchable clocking scheme. With stretching the clock we can provide the locally synchronous module a highly flexible clock to avoid synchronization failure.

In this paper we used a clock gating circuit to reduce the power consumption. As shown in Fig. 3 the *lclk* signal depends upon the state of the input signal *st*, whenever *st* signal goes high it will stretch the *lclk* signal.

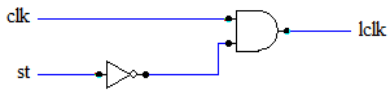


Fig. 3 Stretchable clock generator

3.4 Asynchronous wrapper

A digital system may be integrated with many different modules. By using the GALS methodology, all these modules are designed with traditional synchronous way and encapsulated within an asynchronous wrapper as shown in Fig. 4.

The basic behavior of asynchronous wrapper is described as follows:

- When the locally synchronous module needs to receive data from its predecessor, it will enable the input controller by setting *wr* signal high which intern will set *st* signal high and the clock of the respective receiver will be stopped subsequently.
- When the locally synchronous module needs to send data, it will enable the output controller by setting *rd* signal high, which sends a request to input controller of the receiver, and the input controller acknowledge it only if the receiver needs data.
- After the handshake completes, the input controller will notify the clock generator. Afterward the two synchronous modules continue to operate.

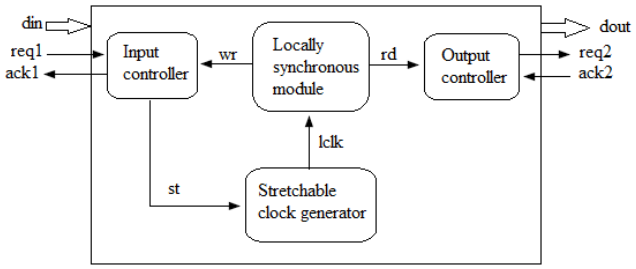


Fig. 4 Asynchronous wrapper

To ensure operation correctness, the clock of synchronous modules must be stopped when the data transfer occurs.

The asynchronous wrapper shown in Fig. 4 has two major parts: input controller and output controller.

For the input controller in Fig. 5:

- Initially *req1*, *wr*, *ack1* and *lst* are all logic 0.
- Signal *wr* is asserted if the receiver needs to accept data from the sender.

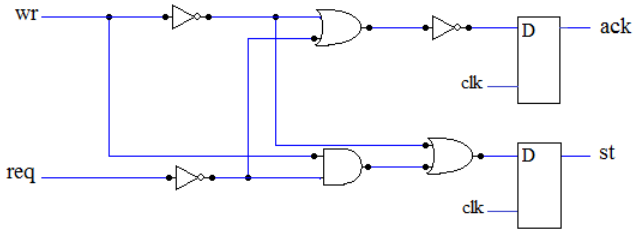


Fig. 5 Input controller

- If there is no request (*req1*) from the sender (sender is not ready to send the data) then stretch (*st1*) becomes logic1 and *ack1*signal will be in logic 0.
- After *req1* signal is received, the input controller resets *st* signal and pulls up *ack1* signal.
- After *st* signal become logic 0, clock of the receiver is restored.

For the output controller in Fig. 6:

- Initially *rd*, *ack2* and *req2* are all logic 0.
- Signal *rd* is asserted if the sender needs to transfer data to the receiver.
- Then request (*req2*) is generated and send to the input port controller of the receiver.
- If the receiver is ready to accept the data it will set *ack2* signal high.
- After *ack2* signal is received, the output controller resets *req2* signal.

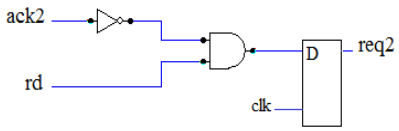


Fig. 6 Output controller

4. Implementation

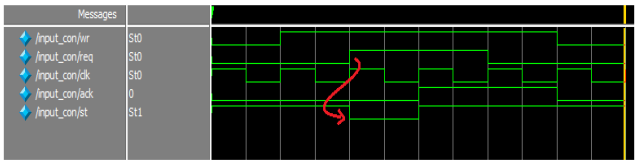
The proposed design is implemented using verilog HDL using ModelSim SE6.5. Synopsys Design Compiler is used to synthesis the design. Here we compare the power analysis of the wrapper with and without stretchable clocking scheme.

5. Result

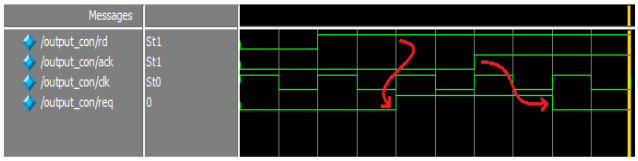
Input controller controls the data entering into the synchronous module. Inputs to the input controller are *wr* and *req* signals. Outputs are *st* and *ack* signals. As shown in Fig. 7(a) if the synchronous module is in write mode and if there is no request from other modules then it will stretch clock by setting *st* signal high. When the request comes the clock will retain its value by resetting the stretch signal and it will set *ack* signal high.

Output controller controls the data leaving from the synchronous module. Inputs to the output controller are *rd* and *ack* and outputs is *req*. As shown in Fig. 7(b) when the system is in read mode, then it will send a request to the receiver. If the receiver is ready to accept the data then it will send an acknowledgement.

Stretchable clocking is used to reduce the power by stretching the clock signal. As shown in Fig. 7(c) stretchable clock generator uses a stretch signal to control the clock. If the stretch signal is high then it will stretch the clock. Otherwise it will release the clock.

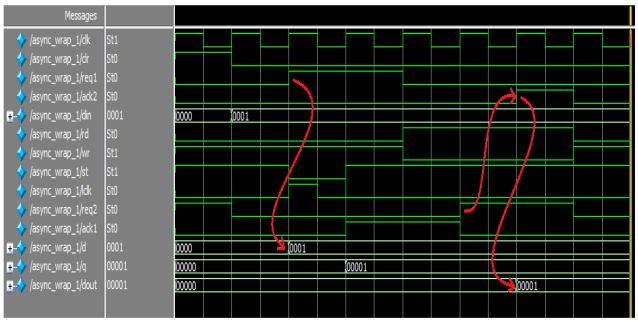


(a) Input controller



(b) Output controller

(c) Stretchable clock generator



(d) Asynchronous wrapper

Fig. 7 Simulation results

Asynchronous wrapper helps the communication between synchronous and asynchronous module. Inputs to the wrapper are *din*, *req1*, *ack2*, *clk*, and *clr*. Outputs are *dout*, *ack1* and *req2*. As shown in Fig. 7(d) when *wr* signal is in logic 1 and if a request (*req1*) comes then it will send back an acknowledgement (*ack1*) informing the sender that it is ready to accept the data and will release the clock.

Here we designed a 4 bit accumulator as the locally synchronous module. When operation takes place inside the accumulator, it will be in busy mode and when the output data is ready, it will set *rd* signal high.

In the read mode if the module wants to send a data, it will send a request (*req2*) to the receiver. If the receiver is

ready to accept the data then it will send an acknowledgement (*ack2*) back to the sender.

6. Conclusion

Globally asynchronous locally synchronous (GALS) method is used for communication between modules in system on chip (SoC). In order to provide reliable communication between modules with different clock frequencies, we propose an asynchronous wrapper circuit based on stretchable clocking scheme.

Table 1 show that the wrapper with stretchable clocking has less dynamic power than that without stretchable clocking. From the table it is clear that with stretchable clocking scheme we can achieve around 20% reduction in dynamic power.

Table 1: Dynamic power of GALS interface

	With stretchable clock	Without stretchable clock
Dynamic power	173.3934 nw	216.1948nw

To reduce the latency during the communication, we can introduce a FIFO in between the asynchronous wrapper modules.

References

[1] A. Davis and S.M. Nowick, "An Introduction to Asynchronous Circuit Design", Technical Report, UUCS- 97-013, Computer Science Department, University of Utah, Sep. 1997, pp.1-54

[2] K. Y. Yun and R. P. Donohue, "Pausible clocking: A first step toward heterogeneous systems", International Conf. Computer Design (ICCD), Oct. 1996, pp.1-6.

[3] J.Muttersbach, T.Villiger, and W.Fichtner. "Practical Design of Globally-Asynchronous Locally-Synchronous Systems", International Symposium on Advanced Research in Asynchronous Circuits and Systems, April 2000, pp.1-8.

[4] S. Zhuang; Weidong Li, J. Carlsson, K. Palmkvist, L. Wanhammar, "An asynchronous wrapper with novel handshake circuits for GALS systems," International Conference on Communications, Circuits and Systems and West Sino Expositions, 2002, Vol. 2, pp. 1521 - 1525.

[5] Yuan-Teng Chang, Wei-Che Chen, Hung-Yue Tsai, Wei-Min Cheng, Chang-Jiu Chen, Fu-Chiung Cheng "A low latency GALS interface implementation, " IEEE Asia Pacific Conference on Circuits and systems (APCCAS)2010,pp.1-4

Anju C received her B.Tech degree in Electronics and Communication from Cochin University of Science and Technology, Kerala, India in 2009 and currently pursuing M.Tech in VLSI Design from Amrita Vishwa Vidyapeetham. Her research interests include VLSI design and system on chip.

Kirti S Pande She graduated from Nagpur University in 2001. She completed post graduation from Amrita Vishwa Vidyapeetham in 2010. She worked as a lecturer at various universities like Nanded University, Visvesvaraya Technological University. Presently she is working as an Asst.Professor at Amrita Vishwa Vidyapeetham and has been engaged with research in VLSI design, processor based design and FSM based digital design.