TIR,
Design and Testing of a Simple GALS Circuit

Bart Blaauwendraad
LiTH-ISY-EX-3314-2002
2002-06-07
TIR,
Design and Testing of a Simple GALS Circuit

Examensarbete utfört vid elektroniksystem
vid Linköpings Tekniska Högskola
av Bart Blaauwendraad
LiTH-ISY-EX-3314-2002

Examinator: Kent Palmkvist
Linköping 14 juni 2002
### Title
TIR, design and testing of a Simple GALS

### Author
Bart Blaauwendraad

### Abstract
Globally-asynchronous locally-synchronous (GALS) systems may become a solution for nowadays challenges in the field of VLSI design. Fully synchronous chips are becoming not feasible anymore due to clock distribution and power consumption problems. The value of GALS lies in combination of well known synchronous design methods and relative simple asynchronous communication channels.

The key components are the communication control ports around the synchronous modules and the stretchable clock also called a wrapper. This clock has a unbound delay and is controlled by events the asynchronous channel.

A simple GALS system consisting of a 4-bit transmitter, integrator and receiver has been designed and layouted for a 0,35 micron CMOS process. A 4-phase bundled protocol is used with GasP FIFOs. Novel circuits has been designed to switch from the one wire asynchronous communication of the FIFO to the 4-phase of the wrapper.

The report also discusses the challenges for manufacture test on asynchronous designs. A test strategy for GALS systems is been developed.
Copyrights

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© bart blaauwendraad
# Table of contents

1 **Introduction**

2 **Asynchronous VLSI design**
   2.1 Asynchronous versus synchronous
   2.2 Asynchronous design
   2.3 Classification of asynchronous circuits
      2.3.1 Data representation
      2.3.2 Data processing
   2.4 Fundamental mode and input-output mode
      2.4.1 Fundamental mode:
      2.4.2 Input-output mode:
   2.5 Muller C-element
   2.6 Design concepts

3 **Test methods of asynchronous circuits**
   3.1 Design for testability
   3.2 Fault models and theory
      3.2.1 Stuck-at
      3.2.2 Stuck-open
      3.2.3 D.U.D.E.S.
      3.2.4 Fault equivalence
      3.2.5 Fault dominance
   3.3 Test methods
      3.3.1 Iddq
      3.3.2 Scan-path testing
      3.3.3 Built-in self-test (BIST)
   3.4 Test strategy for asynchronous designs

4 **A simple asynchronous TIR circuit**
   4.1 Asynchronous FIFO
   4.2 Wrapper
      4.2.1 W-port
      4.2.2 R-port
      4.2.3 Stretchable clock
   4.3 Synchronous integrator
5 Results

5.1 Planning and tools 19
5.2 Design the TIR 19
5.2.1 Module between FIFO and R-port and W-port 19
5.3 Performance and operation of the TIR 21
5.4 Recommended test strategy for GALS 23

6 Conclusion and recommendation

6.1 Evaluating the TIR design 25
6.2 Future work 25
6.3 The future of GALS 26

Bibliography 27

Appendix A 28

Appendix B 31
1 Introduction

The last decade an enormous increase of the number of transistor in VLSI designs has been reported. The developments will go further in the future with the downscaling of the technology. Fully synchronous design will not be feasible any more. There are several challenges for fully synchronous designs, which are becoming a increasing burden.

Power consumption is one of them. With the increased density of transistors, the power consumption increase rapidly. Besides voltage scaling other techniques are needed in order to limit the power requirements.

Clock distribution is another big challenge. The distribution time window reduces inverse proportional with the clock frequency and therefore has more and more effort to be put minimizing the clock skew. This is needed in order to ensure correct operation.

A special group of asynchronous designs may become a solution for the above described challenges. Splitting up the whole chip in different modules, which communicate asynchronously between each other, can ease the design requirements. This report discusses the design and layouting of such a globally-asynchronous locally-synchronous (GALS) circuit.

In the first chapter more details on asynchronous design are been discussed. Chapter three presents common methods of manufacture testing and report some results of asynchronous designs testing from the literature. The features of the developed GALS system, a transmitter integrator receiver (TIR), are described in chapter four. The next chapter discusses the results. Conclusion and recommendations can be found in chapter six.
2 Asynchronous VLSI design

This chapter gives an overview on asynchronous VLSI design. At first a comparison is being made between asynchronous and synchronous designs. Section 2.2 describes the general idea on asynchronous design. The next section defines the classification of asynchronous circuits. In section 2.4 the theory of fundamental mode and input-output mode are being discussed. Micropipelines and GALS design concepts for asynchronous circuits are described in section 2.5. The last section of this chapter discusses the Muller C-element.

2.1 Asynchronous versus synchronous

Although some years ago a big rise of asynchronous designs had been predicted in the literature, currently most VLSI designs are still synchronous. Despite the fact that Moore's Law is still valid these predictions are not odd. Looking at the advantage and shortcomings of asynchronous designs, there are some general benefits of using asynchronous designs compared to synchronous designs:

- The performance of a synchronous VLSI system is being limited by the worst-case latency. The clock frequency has to meet to worst global timing condition. In general this is rare but must be taken in account to avoid the metastability problem and incorrect results.

- Also timing problems related to the clock are been avoided. Problems with metastability can occur when data is invalid at clock transitions. Also clock distribution problems like jitter and skew do not appear in asynchronous design. Therefore asynchronous designs have a higher re-usability.

- The power consumption of asynchronous designs can be lower in comparisons to the synchronous counterpart. The lower power consumption is a result of no unnecessarily energy-consuming transitions happening. In an ideal asynchronous design is each transition useful. In synchronous designs parts of the circuit that are not involved in a computation process still consumes power at clock transitions.

- Asynchronous designs are more robust to variation of temperature and voltage. Correct operation over a large range of supply voltages and temperatures are being reported [Mol99].

- The electromagnetic radiation can be reduced by the absence of a global clock and no clock harmonics in the emission spectra can be found. This results in a lower electrical interference with its environment.

However there are also some disadvantages. Most of today's CAD tools are not really suitable for asynchronous design. Especially tools for testing and test vector generation [Spa01] Also the designer must pay greater attention to the dynamic properties of asynchronous circuit. More effort must also be put in the design proces, to avoid hazards and deadlocks in the circuit.

2.2 Asynchronous design

In an asynchronous design, some form of handshaking between neighboring circuits has replaced the clock signal. There are different types of handshaking protocols. Figure 2.1 represents a simple asynchronous circuit. A general solution is based on using two wires. One wire is for signaling a request and the other for signaling an acknowledgement to that request. Assume the request and acknowledge wire of figure 2.1 to be low in their initial state. A 2-phase handshake contains the sequence “request-up followed by acknowledge-up”. The next 2-phase handshake is then “request-down; acknowledge-down”. See figure 2.2.
The main disadvantage of the 2-phase handshake protocol is that the state differs from the state before that handshake. A 4-phase consists of two transitions more to return in the initial state. This can be seen in Figure 2.3. In comparison with the 2-phase handshake protocol, a 4-phase is slower and consumes more power. In general, the circuits are simpler and less expensive. Other types of handshaking are also possible. In [Ber96] circuits are proposed, using only one wire with active circuits at both ends to pull the wire up or down.

The introduced protocols above all assume that the sender is the active party that initiates the data transfer over the channel. This is known as a ‘push channel’. The opposite, the receiver asking for new data is also possible and is called a ‘pull channel’. The direction of the request and acknowledge signals are then being reversed. The validity of the data is indicated in the acknowledge signal from the sender to the receiver.
2.3 Classification of asynchronous circuits

In general synchronous designs can be seen as a particular case of representing data processing designs in the multi-dimensional asynchronous world. There are many different approaches to designing asynchronous VLSI circuits. Nevertheless, the most popular design approaches currently in use can be categorized by the way data is represented and processed.

2.3.1 Data representation

Data in asynchronous designs can be represented either by using a dual rail encoding technique or a data bundling approach. In the dual rail encoded data representation, each Boolean variable is represented by two wires. Here each wire carries the data and timing information. The data itself is represented by logic levels (e.g. a one is represented by a high voltage and a logic zero by a low voltage) or by transition encoding where a change of a signal level conveys information. The bundled data approach uses one wire for each data bit and a separate control wire containing the timing information. In combination with the handshaking protocols several different data representations can be plot. The 4-phase bundled-data, 2-phase bundled-data and 4-phase dual-rail are used the most frequently.

2.3.2 Data processing

At gate level there are three basic models for data processing: speed-independent, delay-insensitive or self-timed.

![Figure 2.4: A delay tree of logic gates](image)

Speed-independent (SI) circuits assume that the logic elements of the VLSI design may have an arbitrary propagation delay but transmission along wires is instantaneously. Referring to figure 2.4 this means a positive bounded but unknown delay for $d_a$, $d_b$, and $d_c$. The delay of the wires is assumed to be zero, $d_1=d_2=d_3=0$. For today’s semiconductor processes this assumption is not very realistic.

A circuit that operates correctly with positive, bounded but unknown delays in the wires as well as in gates is been called delay-insensitive (DI). Looking at figure 2.4 this means an arbitrary delay for $d_a$, $d_b$, $d_c$, $d_1$, $d_2$ and $d_3$. These circuits are obviously extremely robust. Unfortunately these circuits can only be made out of Muller C-elements and inverters. By carefully design at gate level circuits can be made quasi-delay-insensitive. This requires the delays $d_2$ and $d_3$ in the wire fork in figure 2.4 to be equal. When signal transitions occur at the same time in all end-points of a wire, such wire is called isochronic.

Circuits whose operation relies on more elaborate and/or engineering timings assumptions are called self-timed. They are without well-defined properties under the unbound gate and wire delay model. They run at the faith of the designer.
2.4 Fundamental mode and input-output mode

In addition to the delays in the gates and wires, it is also necessary to formalize the interaction between the circuit being designed and its environment. Again, strong assumptions may simplify the design of the circuit. Design methods that have been proposed over time all have their roots in one of the following assumptions [Spa01].

2.4.1 Fundamental mode:

The circuit is assumed to be in a state where all input, internal and output signals are stable. In such state the environment is allowed to change one input signal. After that the environment is not allowed to change the input signals again until the entire circuit has stabilized. The internal signals such as state variables are unknown to the environment. This implies that the longest delay in the circuit must be calculated and the environment is required to keep the input signals stable for at least this amount of time. Therefore the delay in gates and wires has to be bounded from above. This limitation on the environment is formulated as an absolute time requirement. The fundamental mode design approach for asynchronous sequential circuits is based on the work of Huffman in the 1950s.
2.4.2 Input-output mode:
Again the circuit is assumed to be in a stable state. Here the environment is allowed to change the inputs. When the circuit has produced the corresponding output, the environment is allowed to change the inputs again. There are no assumptions about internal signals and it is therefore possible that the next input change occurs before the circuit has stabilized in response to the previous input signal change. These circuits are speed-independent.

The restrictions on the environment are formulated as causal relations between input signal transitions and output signal transitions. For this reason the circuits are often specified using trace based methods where the designer specifies all possible sequences of input and output signal transitions that can be observed on the interface of the circuit. David Muller pioneered the input-output mode of operation in the 1950s.

2.5 Muller C-element
The Muller C-element is an important state holding component of asynchronous circuits. In comparison to an AND or OR gates conclusions can be made on the inputs when the output change from 0 to 1 or vice versa. When both inputs are 0 the output of the Muller C-element is set to 0 and when they are 1 the output changes to 1. Because handshaking involves cyclic transitions between 0 and 1, it is clear that the Muller C-element is a fundamental component and is the AND function for two events.

![Figure 2.7: A Muller C-element implementation](image)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 2.8: Muller C-element truth table*
2.6 Design concepts

An important concept in asynchronous design is called 'micropipelines' [Sut89]. By using a pipeline the number of elements doing computations at a given time increases. A micropipeline is processing the data asynchronously. Figure 2.5 presents a general structure of a 2-phase event driven micropipeline. For this implementation a special latch is needed for altering between catch and capture states.

![Figure 2.5: A Muller C-element micropipeline](image)

A major advantage of the micropipeline structure is the possibility of filtering out all the hazards in the logic blocks. Another feature is that an asynchronous micropipeline is automatically elastic. Data can be sent to and received from a micropipeline at arbitrary times. Also 4-phase bundled-data pipelines can be build. In [Ber96] a single-track signaling for micropipelines is discussed. They operate in a similar manner. However all these micropipeline do not harmonize easily with synchronous modules. GALS are a solution to combine the advantage of asynchronous and synchronous circuits design.

Globally-asynchronous locally-synchronous (GALS) design is a paradigm to replace fully synchronous designs. Synchronous design methodology has several proven qualities. However, as the semiconductor technology scales down and the complexity increases, fully synchronous designs will eventually not be feasible on the chip level. The idea of GALS is to use self-timed locally synchronous modules with stretchable clocks, which communicate asynchronously with other modules. Request and acknowledge signals are being used for handshaking to transfer data between the synchronous modules. This idea dates back to the 1960s and was the topic of the PhD thesis of Capiro in 1984 [Cap84]. A basic structure of a GALS is shown in figure 2.6.

![Figure 2.6: Basic structure of a GALS](image)

The main challenges in the design of GALS circuits is to design a unified and robust asynchronous wrapper. This wrapper isolates the locally synchronous modules and has to glue with various synchronous paradigms and can generate a reliable stretchable clock frequency. It has to be able to handle fast data communications in order to become realistic alternative to fully synchronous designs.
3 Test methods of asynchronous circuits

When a chip has been designed, tests must be developed to separate faulty chips from good ones. A fault is a manifestation of a manufacturing defect. They may be caused by mechanisms ranging from crystalline dislocations to lithography errors and bad etching of vias.

The aim of testing is to predict to a very high level of certainty if a manufactured chip correctly works. It is impossible to predict totally fault-free chips. By applying one or several special designed tests, fault coverage rate between 80 and 99.9% can be achieved. However, these manufacture tests take time and therefore cost money. Thus these tests must be optimized for time and accuracy. Manufacturing tests also helps to catch low reliability chips.

Asynchronous circuits, loosely speaking, are arbitrary interconnections of logic gates with the restriction that no gate outputs can be tied together. Synchronous circuits satisfy the additional restriction that all cycles in it must be broken by clocked memory elements. This restriction makes the analysis, synthesis and testing of synchronous circuits generally easier than their asynchronous counterparts.

At first are design methods for testability presented in section 3.1. The next section gives an overview on fault models and theory. Section 3.3 describes some widespread test methods. The last section describes some results on asynchronous design testing methods from the literature.

3.1 Design for testability

Controllability and observability are two key concepts in design for testability (DFT). Controllability refers to ease of producing test patterns to the inputs of the sub-circuits via the primary inputs. Observability refers to the ease of determining the response of the sub-circuit at the primary outputs of the circuit. Using additional logic elements and control terminals can increase the degree of controllability and observability of a circuit.

To improve the testability three groups of DFT techniques are distinguishable: ad hoc strategy, structured approaches and build-in self-test techniques. When choosing the most suitable method some criteria must be taken in account:

- Impact on the original VLSI design
  - Increase in chip area
  - Effects on performance
  - Testability of the extra logic
- The ease of implementation of the chosen method
  - The effects on test pattern generation
  - Reduction in computational time
  - Improving fault coverage
- Reduction in engineering effort
- Additional requirements for automatic test generation

The ad-hoc strategy is based on recommendations for improving the testability of VLSI circuits. These recommendations make test pattern generation easier; simplify test application and fault isolation.

For instance multiplexers and demultiplexers can be used to improve the controllability and observability characteristics of a VLSI circuit. They allow the test engineer to change the direction of data streams inside the circuit. The major penalties of this approach are hardware redundancy and additional propagation delays in the VLSI circuit.

Structural test approches and build-in self-test are discussed in more detail in section 3.3
3.2 Fault models and theory

There are several fault models like stuck-at, stuck-open, which in different ways describes the fault. With these models fault test strategies can be made.

3.2.1 Stuck-at

Stuck-at fault model is one of the widely used. It assumes that manufacture faults will result in wires at the gate logic being permanently logic zero or one (stuck-at-0 either stuck-at-1). Many circuit faults can be modeled by the stuck-at fault model at the logic level. Theoretically, for any circuit with multiple stuck-at faults the total number of faulty circuits grows exponential with the number of gate. Therefore, in practice, only single stuck-at faults are considered in order to eliminate an incredible large number of faulty VLSI circuits.

3.2.2 Stuck-open

Another fault model is the stuck-open model, which models faults in individual transistors instead of entire logic gates. A stuck-open fault can lead to a gate behaving as if it was a memory. For example, if input b of the nmos transistor of the NAND gate is stuck open it is only visible after applying the inputs 0-1-0 and 11. The output than remains 1.

3.2.3 D.U.D.E.S.

A relative new fault model is been called DUDES and has been developed to address the problem of fault collapsing in asynchronous circuits. DUDES is developed [Shi00] to analyze single stuck-at faults by mapping these faults into a set of high-level faults that appear on the input and output leads of the basic block

A set of faults is introduced, namely down-enabled, up-enabled, down-disabled and up-disabled faults together with the stuck-at fault model. The stuck-at model is already described in section 3.2.1. Up/down-enabled are faults at a gate output, which may be able to make a transition only in one direction. Up/down-disabled considers inputs that fire premature. The circuit then generates an output without waiting for the correct input. The DUDES fault model consider-
ers pattern-sequence-dependent faults. The corresponding abstraction can be used in test pattern generation.

![Figure 3.2: A example combinational logic circuit](image)

3.2.4 Fault equivalence
A fault is said to be functionally equivalent to another if and only if the output function realized by the circuit with the first fault present is equal to the function realized when only the second fault is present. For example, in the network of figure 3.2, in the presence of the $c$ stuck-at 0 the function implemented is $\text{out} = ab$. Hence the fault $c$ stuck-at 0 and $y$ stuck-at 0 are functionally the same. So if a pattern is generated that detects $c$ stuck-at 0 the pattern will also detect $y$ stuck-at 0 and vice versa. Therefore, only one of them needs to be considered.

3.2.5 Fault dominance
Consider the fault $y$ stuck-at 1. The set of test patterns that detect this fault is $A = \{abc = 000, 001, 010, 011, 100\}$. Similarly for the fault $c$ stuck-at 1, the set of patterns that detect this fault is $B = \{abc = 100\}$. So, $B$ is a subset of $A$. Hence, if we generate a pattern that test for $c$ stuck-at 1, $y$ stuck-at 1 will also be detected. Therefore $y$ stuck-at 1 can be removed from the list of faults to consider. In this case the fault $y$ stuck-at 1 dominates the fault $c$ stuck-at 1.

3.3 Test methods
The different fault models generate at starting point for test strategies. Based on these models a test can be generated. In general there are two different types of test: current tests and voltage tests. In practice both are being used beside each other to achieve the highest fault coverage.

3.3.1 Iddq
Iddq testing is an important current test. In steady state, when all switching transients have settled-down, a CMOS circuit dissipates almost zero static current. In a defect-free CMOS circuit the leakage current is negligible, in the order of a few nano-amperes. In case of a defect such as gate-oxide short or a short between two metal lines, a conduction path can be formed from the power-supply (Vdd) to the ground (Gnd) and subsequently significantly high current are being dissipated. This faulty current is a few orders of magnitude higher than the fault free leakage current. Thus, by monitoring the power-supply current, one may distinguish between faulty and fault-free circuits.

Because there is no clock to create a discrete time, fewer quiet states can be found in asynchronous designs. Therefore additional circuits are required to create more stable states. In [Ron96] a HOLD element is introduced to control the handshaking during tests. To limit the amount of design for test area the HOLD circuits are placed in the endpoints of the control logic at those places where interaction with the datapaths occurs.

3.3.2 Scan-path testing
Test vectors are being used to perform a Boolean/voltage level test by applying the logic levels to the inputs of a circuit and check for correct logic level on the outputs. This approach assumes that during the test all the memory elements of the sequential circuit are configured into a long shift register called the scan-path. All the memory elements of the circuit can be
controlled and observed by means of shifting in and shifting out data along the scan-path.

This technique can be used to partition a VLSI structure to a number of less complex subcircuits by organizing the scan-path to pass through a number of combinational networks. The sequential depth of such a circuit is much less than the depth of the original one, which alleviates the test problem considerably. To test the scan-path itself, flush and shift test are applied. The flush test consists of all zeros and all ones. The shift test exercises the memory elements of the scan-path through all their possible combinations of initial and next states.

If the scan-path operates correctly test vector are applied to test the combinational logic circuits. Automated test pattern generation (ATGP) is used to create these sequences of test vectors. The essence of test generation is identifying cases for which the good and faulty circuits give different results [Wol98]. The theory fault equivalence and dominance is used to minimize the length of the test pattern. Therefore these tests does not form a substitute for a functional verification of the circuit.

### 3.3.3 Built-in self-test (BIST)

Built-in self-test methods form an alternative for applying a test vector from the outside of a chip. An important advantage is the test speed especially for large chip. Testing can be done at the internal speed of the chip. This is faster than using an external test setting. To prevent the use of a big and costly chip area, (pseudo)random sequences generated are used instead of one created by an ATGP program. A microprocessor could also be used to generate functional patterns for the test. A linear feedback shift register can be used to generate pseudo-random sequences.

### 3.4 Test strategy for asynchronous designs

In [Pet94] a method has been developed for testing micropipelines. The test procedure is split up in testing both the control part and the datapaths. Single stuck-at faults in the control circuit are tested during normal operation whereas two test patterns are required to detect those faults in the datapaths of the micropipeline.

In [Ron00] a Cellular Automata (CA) has been used as a finite state machine for the generation and evaluation of test patterns. This method is based on an abstract hardware description model of the instruction length decoder. This is independent of the implementation details and hence also independent of the asynchronous circuit style. This CA-BIST solution presents a fault coverage of 94% which is similar to the clocked circuit.

The HOLD element of [Ron96] has proven to be successful in order to achieve 100% stuck-at fault coverage with a combination of scanpath, deadlock and Iddq testing. This DFT method is implemented in the Philips high-level test compiler for asynchronous circuits.
4 A simple asynchronous TIR circuit

The assignment of this project was to develop and layout a simple GALS. Therefore a small communication network with transmitter, integrator, and receiver (TIR) has been chosen. It performs basic operations and is therefore suitable for this project. An overview of the TIR is given in figure 4.1. This chapter discusses different parts of the TIR. In section 4.1 a special asynchronous FIFO is been discussed. The next section the implementation of the wrapper. Function of the W-port, R-port and stretchable clock generator are explained. The synchronous integrator is the topic of section 4.3.

Figure 4.1 Overview of the TIR circuit

4.1 Asynchronous FIFO

To increase more flexibility a FIFO is placed in the datapaths around the wrapped integrator. See figure 4.1. To prevent the FIFO becoming a bottleneck in the system a GasP control FIFO has been chosen at the begin of the project. GasP stands for globally asynchronous pipeline. This FIFO control is self-timed asynchronous circuit, which behaves only in large asynchronously [Sut01a].

Figure 4.2: a GasP FIFO

The FIFO consists of two parts PLACE and PATH. The PLACE keeps the state of the FIFO by using a data latch with three inverters for the data path and a control wire between two PATHs (e.g. S1, Status OUT). The voltage level on node S1 figure 4.2 is setting the status of the data in the path. With the definition low means FULL and high means EMPTY data can be transferred. The PATH functions like a door between two PLACES. When it has fired data is being moved one place further.
In the case of a FULL on node status IN and EMPTY on S1 the circuit will be triggered to pass the data to the next data latch. The output of the Nand will go low. As a result the PATH will conduct. At the same time status IN will be set to high and thus becomes EMPTY. At the other side S1 will be set to FULL after own inverter delay by d.

In [Sut01a] throughput of 1.5 giga data items per second are been reported for a 0.35-micron chip. Each FIFO stage operates at the speed of a three-inverter ring oscillator. A simulation result in spectre of four GasP stages is shown in figure 4.3. It shows the elastic characteristics of the FIFO. ST_out remains low for a relative long period and after four data shifts ST_in is put low indicating that the FIFO is completely filled with data. The short pulse on ST_out generates an avalanche of shift operations and for a short time ST_in remains empty.

Figure 4.3: A GasP FIFO control simulation

4.2 Wrapper

Wrapper is been design to encapsulated a locally synchronous circuit into a GALS. Although not much research has been done in compare with asynchronous design, a few solutions are been presented in the literature [Nlo01], [Zhu02] and [Lil01]. For the TIR a basic point-to-point GALS system as shown in figure 4.4 is sufficiently. A wrapper consists basically of a clock generator, a port to receive data end to send data. In the next sections each part is discussed in more detail.

Figure 4.4: Detail schematic of a point-to-point
4.2.1 W-port
The W-port forms the active part of the wrapper [Zhu02]. For the correct operation of the system, the data communication must be highly reliable and robust. In figure 4.5 a Muller C-element implementation is given. The handshaking for the asynchronous side is based on the 4-phase bundled data protocol. On the synchronous side WR\(^+\) initiates a write operation and the STRETCH signal indicates the progress of the operation. When STRETCH is high its prevents the generation of a new clock signal and so an new WR\(^+\). The advantage in comparing to other purposed W-port implantations is that WR need not to be high during the complete cycle of transition as shown in figure 4.6. This may speed up the communication. The reset is also used for this.

![Figure 4.5: W-port using Muller C-elements](image)

![Figure 4.6: W-port signal transition graph](image)

4.2.2 R-port
The R-port forms the non-active side of the asynchronous channel. It has to react on a request under the condition of a read request from the synchronous part. A signal transition graph is shown in figure 4.8. For the implantation two Muller C-elements are being used as shown in figure 4.7. The set node puts the output of Muller C on the left to high. This prevents the transition of a new clock pulse and set the R-port in the position to receive data.

![Figure 4.7: R-port using Muller C-elements](image)
4.2.3 Stretchable clock

The main motivation behind machines with stretchable clocks has been to avoid the metastability problems. Stretchable clock can stretch a clock phase for an unbounded period of time. In the meanwhile inputs and outputs will become valid. Therefore they are suitable for interaction with the global asynchronous characteristics of a GALS.

The stretchable clock consists of a ring oscillator and a Muller C-element as shown in figure 4.9. For safety and reliability of the clock a Muller C-element is been used. If STRETCH is not asserted to low, the output and inputs of the C-element will follow the signal transitions of figure 4.10. If STRETCH is asserted to high, the input $X_a$ is set to low, the output of the C-element could be either low or high. The output will eventually be maintained at a low level. The next rising edge is postponed the $STRETCH^+$. An OR gate is used for multiple request for stretching the clock.
4.3 Synchronous integrator

The synchronous integrator consists basically of a adder and memory element. The input is being add up to the previous result. This configuration is shown in figure 4.11. When Cout is high the adder is at its maximum. By using this signal to reset the Flip-Flop a new integration can be made.

![Figure 4.11: Implementation of integrator](image)
5 Results

This chapter the result of this project is been presented. The first section describes the choose strategy and planning. The next section important discuses solutions for some major problems encountered during the design phase. Section 5.3 gives an overview of the performance of the system. The latest section presents some recommendation for testing GALS systems in relation to the test theory described in chapter 3.

5.1 Planning and tools

There no special CAD tools available to create GALS system with a high synthesis tools. Only for the synchronous part are reliable, highly productive and well-know CAD-tools available. At the start of the project CADENCE custom IC design designer 4.4.5 has been chosen. Together with the simulator Spectre it forms a suitable CAD tools environment for small projects.

The project was divided in two parts; first to design and layout a simple GALS circuit and secondly to discuses test strategies for asynchronous design. The TIR described in chapter 3 is chosen for realizing a simple GALS.

In the end of March a halfway report has been made to create an overview of the work that has been done so far and generate a more specific planning. This document can be found in Appendix B. Not all the goals have been achieved. For example, no other implantation for the wrapper has been design and layouted.

5.2 Design the TIR

During the design process some thing did not work out as at first expected. Getting the FIFO control working took far more time and a lot of different solutions from [Sut01a] has been simulated. Also the data transfer was corrupt therefore instate of using one Nmos a switch with a Nmos and Pmos has been used.

The reason for these troubles was lying in the feedback and the need for well-balanced transistor sizes. As described in section 4.1, the GasP FIFO control works with as a three-gate inverter delay oscillator. For a fast operation the input capacitance of a logic gate has to be almost the same as the capacitance that needs to be loaded or discharged at the output. This is based on the theorem of logical effort [Sut01b].

5.2.1 Module between FIFO and R-port and W-port

The implemented GALS use two different type of handshaking. The handshaking of the wrapper is based on the 4-phase bundled data and the FIFO uses one wire to indicate its status. Therefore a module had to be designed to solve this communication problem. The characteristics of the R and W port differ, so a general solution is not possible. There are several methods to synthesis an asynchronous circuit. In this case for both circuits a signal transition graph is been made. The circuit has to be reliable and free of hazards in order to prevent deadlock states in the TIR.

Figure 5.1 shows the module FIFO to R-port. This circuit design is based on a signal transition graph in figure 5.2.
Figure 5.1: FIFO to R-port handshake module

Figure 5.2: STG of a FIFO to R-port handshake module

Figure 5.3: W-port to FIFO handshake module

Figure 5.4: STG of a W-port to FIFO handshake module

Figure 5.5: A spectre simulation is shown with the correct functioning of the W-port to FIFO module. The transitions are following the signal transition graph of figure 5.4. Although WR is pulsing to often for example during period of Stretch high. The reason is the shortcoming of the simulation program on sources. It is impossible to let them behave event driven.

In figure 5.5 After the first WR pulse STRETCH goes at once to high and remains high during the up and down transitions of the request and acknowledge signals. Directly after the up tran-
sition of Ack, the FIFO stages is being filled with data. After 15ns FIFO data is being shift after a short high on the status out wire. The acknowledgement cycle is completed and a new write cycle starts.

![Figure 5.5: W-port to FIFO simulation](image)

### 5.3 Performance and operation of the TIR

The TIR has been design for the 0.35 CMOS process. The supply voltage is 3.3 Volt. The total chip area without external pads is unknown. Due to a lack of time a total layout has not been made. Although standard-layout cells for the R-port W-port and other modules are made. In the simulations of the schematics it can operate at least the speed of 200 Mhz. In Appendix A the testbench for this simulation can be found and layout figures from some basic modules.

The performance of the FIFO is extremely high. In less than 0.65 nano seconds a shift operation has been made. This means it can operate at the speed of 1.5 giga Hertz. See in figure 5.5 the graphs on top. So the FIFO is no bottleneck for the communication channel.

During the implantation a method had to be chosen to create the WR and RD signals in the wrapper. These signal are not specially generated by the synchronous part. Which could be put as an requirement in the synchronous design. So the high level of the clock signal is being used as RD and the inverted clock signal to create WR. Therefore RD and WR are never active on the same time. A drawback of this method is the reduction of the computation time. Instead of a full clock period only half of the time is available.

Figure 5.6 shows the some important control signal of a simulate TIR. The WR and RD signals are generated in the transmitter respectively the receiver. The have to be set before the simulation and make it difficult to find the maximum speed of the circuit. Because they can not be event driven during the simulation. This is one of reasons why Cadence design environment not really suitable is for asynchronous designing. Figure 5.7 shows a simulation result of the synchronous integrator.
Figure 5.6: Simulation of the control path of the wrapper

Figure 5.7: Simulation of the synchronous part only
5.4 Recommended test strategy for GALS

In chapter 3 is dedicated to the theory of testing with a focus on asynchronous circuits. This section focuses more specific on recommendations on test development for GALS.

Create one or more scan-path to reach all latches or flip-flops together of the synchronous parts. The scan-path may operate only when the circuit is in the test mode and can be generated by tools for synchronous designing. Automate test pattern generation tools can therefore be applied in order to minimize the test time and maximize the fault coverage. A external clock is needed in order the shift the states of the scan-path.

The control circuit of the asynchronous part can be tested in normal operation mode. If the handshaking between modules get stuck in compare with a golden chip, the chip is faulty and therefore useless. Therefore a test pattern has to be designed which has to activate all the asynchronous channels in the chip.

An Iddq test can be done by implementing a general Stretch to stop the all stretchable clock of the different wrapped modules in the circuits.

For large chips the method of using scan-paths may be use to much time. In those cases build-in self-test provide a suitable solution just as the do for synchronous designs. Each module can have it’s own BIST. They can even work concurrently.
6 Conclusion and recommendation

This chapter is dedicated to the final conclusions and recommendations on this project. In the first section the TIR design is evaluated. In section 6.2 recommendations for future work are done. In the last section discusses my opinion on the future of GALS.

6.1 Evaluating the TIR design

The results of this project is a designed and layouted small GALS. It consists of a transmitter, integrator and receiver. The system is capable to integrate 4-bits data stream. However this is not enough for real applications. Since the layout work has been done modular, it should be relatively easy to increase the datapath wide.

One of the results of this project are the standard cell layouts of the R-port W-port and stretchable clock. These standard layouts uses only poly and metal one wiring. With these standard cell larger GALS systems can be build in order to test and verify the behavior of multi-channel GALS.

The FIFO is designed on the self-timed principle. The delay of different data wires can vary. The time window for shifting data is really short 200 ps. Some early simulations resulted in unsuccessful operation of the datapath. In the worst case the data is been distorted. Therefore the FIFO is not a really robust part.

Two novel modules has been design to let the GasP FIFO work with the 4-phase handshaking protocol of the wrapper. These modules operate robust and follow the signal transition specification of figure 5.2 and 5.4.

During this project a effort has been putting in managing the circuit to work. Therefore it works on faith of the designer. At this moment I would not advice to take this design in production due to the FIFO. The reliability has to be improved by using a elastic FIFO using a 4-phase handshaking protocol.

6.2 Future work

A really simple GALS has been developed. Advanced design are more interesting for comparing the GALS design method with their synchronous counterpart. High synthesis methods, using for example VHDL, are recommended to use for designing the synchronous parts of a GALS design instead of a analogue design environment. If provides a more productivity and more reliable circuits.

Also a more robust FIFO could be implemented. This FIFO needs to be based on the 4-phase bundled data approach. Also different type of communication ports for the wrapper can be implemented.

The DUDES fault model [Shi00] is useful for generating test patterns for asynchronous circuits without considering the internal implantations. Therefore this fault model can be very useful for generating automated test pattern generation.

The last recommendation for future work is on the test strategy described in section 5.4. The suggest strategy should be applied the on TIR design and a much larger design in order to prove it success.
6.3 The future of GALS

In my opinion GALS are capable to replace some type of fully synchronous circuits. Especially when power consumption is an important requirement. Another interesting point of GALS is its concurrent operation if designed in such way. GALS are therefore a good starting point for the design of systems on silicon.

The lack of design tools capable of the generation of the wrapper in a highlevel environment, however forms a major drawback. I guess this will be a temporarily problem. Asynchronous designing has attention of CAD tools developers and GALS systems are too interesting to forget.
Bibliography


Appendix A

Figure A.1: Testbench of the total TIR design with in the middle the wrapper

Figure A.2: A shematic from Cadence of the wrapper
Figure A.3: The layout of the R-port for a 0.35 micron process

Figure A.4: Testbench of the total TIR design with in the middle the wrapper
Appendix B

The design of a small global asynchronous local asynchronous system

Halfway report of the internship of Bart Blaauwendraad

Introduction

The area of asynchronous system design have gained more and more interest during the last years. It has some advantages compared to synchronous system design. The most important one is absent of clock skew problems. As part of my program of Master degree in electrical engineering an internship of four months has to be completed. This report describes briefly the project goals, the results of the last two months and the plans to the upcoming two months. This report ends with a small reflection on the project so far.

Project goal

The goal is to design a chip layout of a globally asynchronous and locally synchronous system (GALS). This system consists of a transmitter, a synchronous integrator and a receiver (TIR). A second goal is to develop a test strategy suitable for GALS systems.

Results

The first step is to design and simulate a schematic of the system in cadence. Because of the asynchronous part and the goal to make a layout. High-level syntheses tools cannot be used. Therefore the analogue design tools is being used. A drawback is the long simulating time in spectre. To cope with this, the design has been divided in parts. Each part has been simulated separately. A final simulation has been done for the wrapped integrator. The schematic and the plot of a transient simulation can be found in appendix 1 and 2 respectively.

A lot of time has been put in realizing a correct functioning FIFO control. Several different schematics has been tested from [Ber96] and [Sut01]. Finally four different schematics has been put to work. On two other there are as still failures.

The design approach to layout is bottom up by creating standard cells if they are not available already. In week 12 I started to layout the first standard cells. At the moment the following standard cells are ready but not tested; Muller-C element (basis, set and reset), MS flip-flop, data-path and W-port. A layout of a 16-bit Manchester chain adder is also been made. This design is imperfect due to large transistor sizes and a probably malfunctioning carry chain. All the sixteen propagate transistors have been stuck together. The drain-source voltage drop could be a major problem.
Testing
Although the process of making CMOS chips is controlled to a very high degree. There is still is a chance of small production errors. It's therefore necessary to conduct a test before the chips leaves the factory. It imported to have a small test-time and a high degree of accuracy. Testing asynchronous system leads too new challenges. In [Ron96] a method of using hold elements has been suggested. They operated only in the test mode of the chip and can be used to delay control signals. More research has to been done on this subject.

Plans
Week 14
- Layout the R-Port and W-port
- Discuss progress
- Run a complete schematic simulation of the TIR
Week 15
- Test the created standard-cell layouts
- Improve adder
- Create Layout of the complete TIR
- Write/read about test strategy
Week 16
- Simulate layout
- Compare with schematic
- Improve layout
- Discuss test strategy
Week 17
- Short vacation
Week 18
- Redesign of some part e.g. FIFO to a Squared FIFO, Different R- and W ports
- Start writing the final report
Week 19
- Writing final report
- Redesign of some part e.g. FIFO to a Squared FIFO, Different R- and W ports
Week 20,21
- Discussing concept report
- Last simulations on complete layout and schematics
Week 22,23
- Final report
- Presentation
- Document work

Reflection
I am not completely satisfied with my speed of progress in this project. Therefore I will make a more detailed planning in weeks as already been done in this document. The support I get is really good. I can easily go with my questions to Weidong, Shengxian, or Jonas.