Tiempo White Paper #1:
Introduction to Tiempo Technology

Version 1.1 - June 06, 2009

www.tiempo-ic.com

Introduction

This document provides a preliminary introduction to Tiempo unique asynchronous and delay-insensitive design technology, targeting any audience who wish to get a first glance at it. Section 2 reviews the main design principles of Tiempo asynchronous circuits with a couple of examples given in section 3. Section 4 provides a non-exhaustive list of the benefits of Tiempo asynchronous design technology for end products.

Outline

1 Overview 3
2 Design principles.....4
2.1 Delay insensitivity 4
2.2 Muller gate.....4
2.3 Communication protocol 5
2.4 Data encodings.....6
3 Circuit examples.....7
3.1 Four bit adder 7
3.2 FSM.....8
4 Tiempo asynchronous circuits properties 10
4.1 Low power 10
4.2 Smooth current consumption and low noise 10
4.3 Wide voltage range.....10
4.4 Low latency.....11
4.5 Fast wake-up 11

1 Overview

Tiempo offers a portfolio of powerful IP cores based on its innovative asynchronous delay-insensitive design and synthesis technology. As a result, circuits integrating these cores show outstanding physical properties such as ultra-low power, ultra-low noise, ability to work at ultra-low and variable voltage levels, reactivity (sleep mode by default, immediate wake-up), robustness against process-voltage-temperature (PVT) variations and resistance to hardware attacks (e.g., power analysis, fault injections).

The ground reason for such capabilities is that Tiempo asynchronous circuits are *not* controlled by an external clock and their behavior is *not* governed by the sampling of states using clocked flip-flops or latches (there is neither global clock nor local clock).

On the contrary, Tiempo asynchronous (or clock-less) circuits are self-controlled as their behavior is governed by signal transitions rendezvous and signal levels memorization. These principles, implemented with specific signal encoding and glitch-free logic, ensure functional correctness regardless of any actual delay through gates and wires.

This document provides an overview of the fundamental design techniques used in Tiempo asynchronous circuits.

2 Design principles

2.1 Delay insensitivity

Tiempo asynchronous circuits are **delay insensitive** which means that the functional correctness does not rely on the knowledge of the timing. In other words, Tiempo asynchronous circuits' behavior remains correct even if there are timing variations in the gate and wire delays. This fundamental property makes Tiempo circuits very robust against any perturbation impacting the timing, like the voltage, the temperature or the fabrication process (known as PVT variations).

Building such circuits using Tiempo design technology and methodology is made possible using a particular type of logic circuit which is processing signal transitions, always making sure that the generated transitions produce the expected effects (the acknowledgement principle).

2.2 Muller gate

Designs built with this asynchronous logic use a key gate called the C-Element or Muller gate. The C-Element implements two major functionalities: rendezvous and memorization.

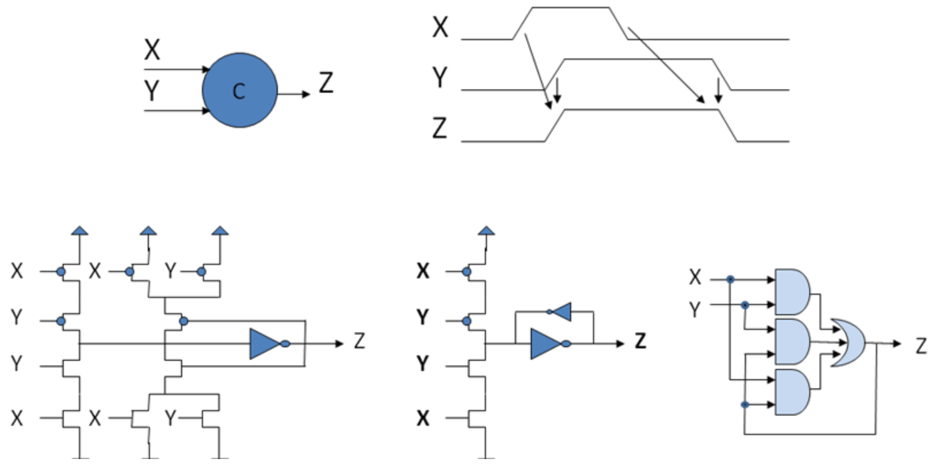

Figure 1: C-element or Muller gate, symbol, chronograms and three possible implementations.

Figure 1 details two different transistor implementations and a standard cell based implementation of this gate. Its behavior is the following:

- When both input signals X and Y are equal, the output Z takes the inputs value, i.e. the input is propagated.
- When input signals X and Y are not equal, the output Z is not changed, i.e. the output is memorized.

A nice way to express the Muller gate behavior is to say that it is performing a logical AND between events. A 0-to-1 event on X and a 0-to-1 event on Y produce a 0-to-1 event at the output Z, and the same behavior applies for the 1-downto-0 events.

Note that this gate propagates a transition if and only if there were transitions on both inputs. Therefore, a transition at the output guarantees that transitions occurred at the inputs whatever the relative timing of the inputs. It therefore performs a AND between asynchronous signal transitions, i.e. a **rendezvous**.

2.3 Communication protocol

In order to build complex asynchronous circuits, a specific protocol has been defined for the different components to communicate between each other in a delay insensitive manner. Figure 2 describes the basic principle of this so-called **four-phase communication protocol**. There exist a lot of variants of such protocols (for example two-phase protocols) that are globally referred-to as **handshaking protocols**, but for the sake of conciseness, the description here is limited to this four-phase protocol.

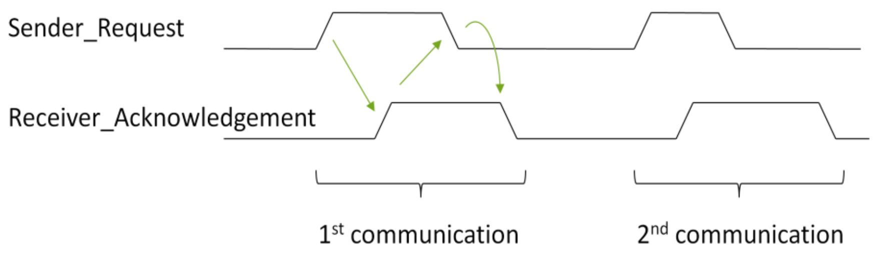

Figure 2: four-phase communication protocol.

In Figure 2, the sender and the receiver are communicating through **a single rail channel**. This channel is composed of a request signal, generated by the sender, and an acknowledgement signal, generated by the receiver. The sender initiates the communication with a rising edge of the request signal. The receiver answers with a rising edge of the acknowledgement signal. Then starts a return-to-zero phase, in which the sender generates a falling transition of the request which is followed by a falling transition of the acknowledgement.

Note that this protocol is only governed by the propagation of transitions, and that sender and receiver actions are interleaved. Therefore, they are actually acknowledging each other, i.e. every time a transition is issued by the sender, the receiver responds to it using a transition and vice versa. In conclusion, as long as the transitions occur, this protocol is robust to any delay in the transitions propagation. It is qualified as delay insensitive.

2.4 Data encodings

In order to exchange data between processing units in an asynchronous circuit, data are encoded so that they are only transported using transitions and without altering the delay insensitive property of neither the logic nor the protocol.

In order to do so, delay insensitive codes are used. One of the most famous delay insensitive codes is the so-called one-hot encoding, or **1-out-of-n code**. Figure 3 describes a 1-out-of-2 or **dual rail channel** along with the corresponding data encoding.

Data	Invalid	0	1	Forbidden
D0	0	1	0	1
D1	0	0	1	1

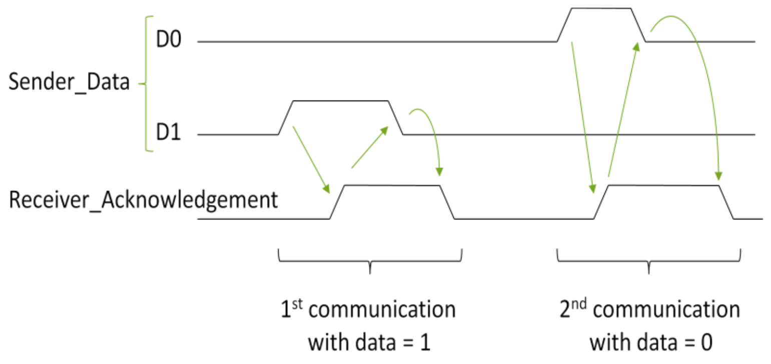
D0
Sender_Data
D1
Receiver_Acknowledgement
1st communication with data = 1
2nd communication with data = 0

Figure 3: typical data encoding and communication protocol.

In practice, Tiempo asynchronous circuits are built using any of the 1-out-of-n codes, the choice between them depending on the context in which it is used. Note that the 1-out-of-1 channel is actually the single rail channel mentioned above, and that this channel does not carry out any data value. It is used for synchronization purposes only (rendezvous, triggering, etc… see FSM example in section 3.2).

3 Circuit examples

The basic principles exposed above are the basics of Tiempo asynchronous design technology that enable to design asynchronous circuits of any complexity. The next sections give the design examples of a simple four bit adder and a simple finite state machine.

3.1 Four bit adder

Figure 4 describes a four-bit adder built out of full-adder (FA) and half-adder (HA) cells. The data flow is here controlled using the data encoding and the acknowledgement signals.

It must be noted that in an asynchronous circuit, the different bits of a data path do not need to be synchronized in time as it is the case in a synchronous circuit. This is particularly beneficial in terms of speed - since data bits are produced as soon as they are ready - and of noise - since the dynamic power dissipation is spread over time (see section 4). Finally, one can verify that the computation is delay insensitive, since a transition is produced at the output if and only if relevant transitions occurred at the inputs.

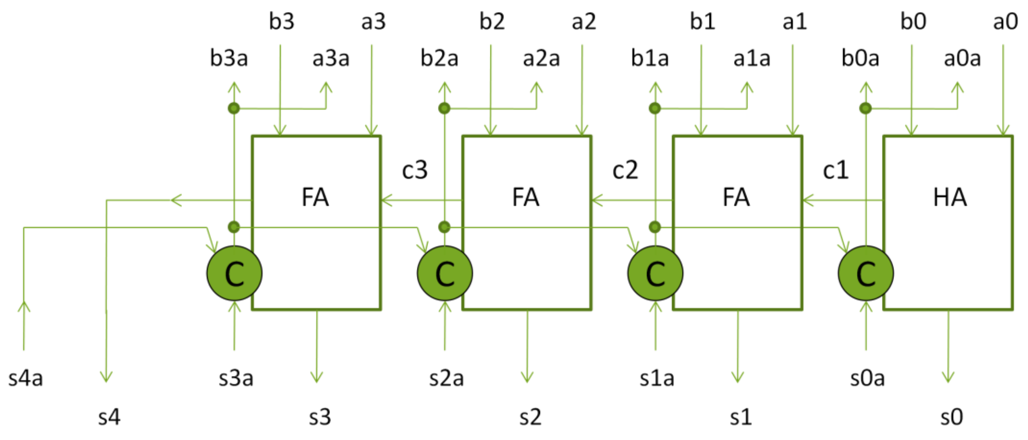

Figure 4: simple four bit asynchronous adder.

As an example to illustrate how the speed of an asynchronous circuit can be computed and improved, the performance of the four bit adder of Figure 4 can be analyzed as follows. As soon as the input bits arrive, the computation starts and the output is produced after a latency called the **forward latency**. Note that this latency has to be computed according to the data since it is dependent of the data values at the input of the adder (length of the carry propagation chain). Then, after the receiver processes the adder output, the acknowledgement signals come and the acknowledgements of the inputs are computed which takes some time, called the reverse latency. The return-to-zero phase then occurs taking another amount of time.

The circuit performance proposed in Figure 4 can be improved by applying **pipelining**, done by adding a self-controlled register at the output of the four bit adder cell and/or cutting the carry propagation chain. Figure 5 shows the new circuit which is faster because the registers are able to generate acknowledgement signals without waiting for the receiver acknowledgement signals. Of course, the data flow is controlled in a way that prevents any data loss.

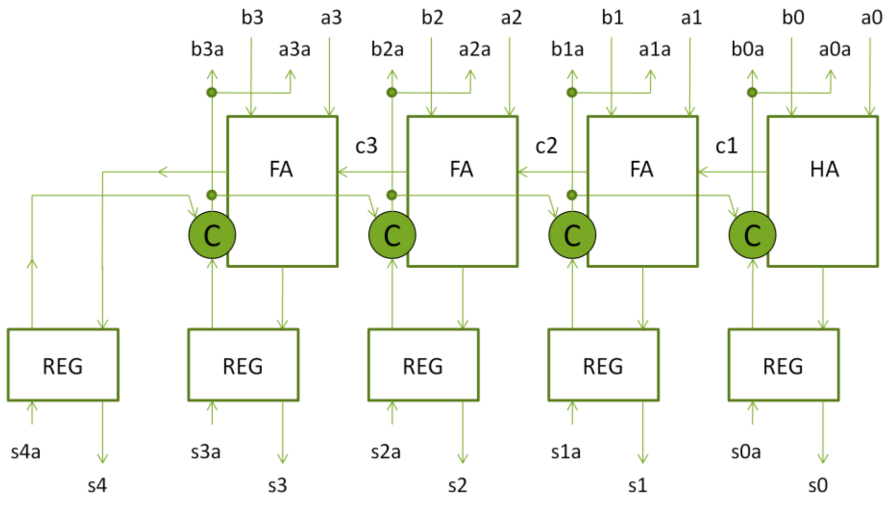

Figure 5: pipelined four bit asynchronous adder.

3.2 FSM

In order to illustrate how control machines are designed with Tiempo asynchronous technology, a very simple finite state machine is considered here with the specification given in Figure 6. The system considered here waits for an event on the GO channel which triggers a sequence composed of two successive actions, named Action1 and Action2. The actions consist in performing a full communication through channels A1 and then A2. After Action2 is completed, the system resumes the communication on the GO channel, and then waits for the next event.

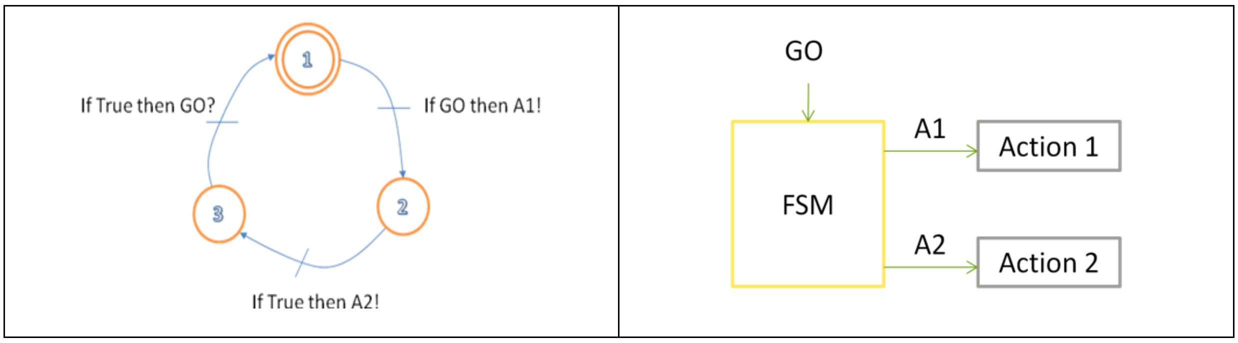

Figure 6: finite state machine specification and the system structure.

The implementation of such finite state machines uses a basic building block called a **sequencer**. The sequencer specification and implementation are described in Figure

7. Note that this sequencer is a very simple cell, including only three gates, but implementing a rather complex behavior.

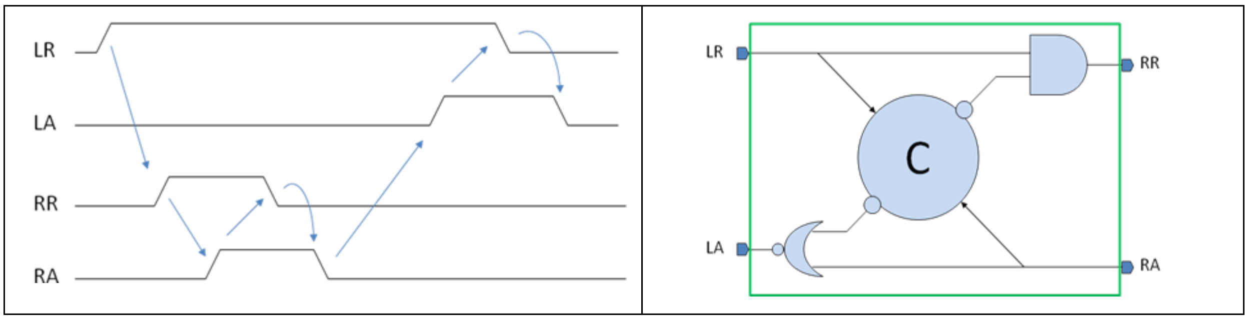

Figure 7: the sequencer specification and a possible gate implementation.

The implementation of the finite state machine using this sequencer is shown in Figure 8. Note that this circuit is very compact (only four gates) and is fully reactive to its environment. Indeed, it first waits for the GO channel to start a transaction, then starts A1 and waits for Action1 to complete, then starts A2 and waits for Action 2 to resume, and finally completes the GO transaction and waits for the next GO transaction.

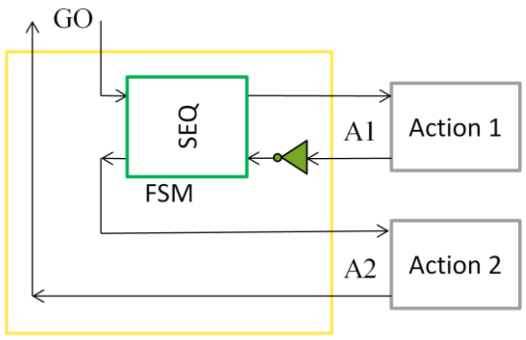

Figure 8: implementation of the finite state machine using sequencers.

It is also noticeable that this finite state machine implements all this behavior using only the rendezvous primitive to memorize the state and hence limits the number of transitions to the minimum. The dynamic power is therefore very low and does not depend on the timing of actions 1 and 2. Moreover, the response time of the finite state machine to its environment is extremely low, i.e. a few gate transitions.

Tiempo design methodology allows the design of FSMs of any complexity.

4 Tiempo asynchronous circuits properties

4.1 Low power

Tiempo asynchronous circuits' **dynamic power** is very low because the different parts of a given circuit are in sleep mode by default and therefore consume no dynamic power at all when not operated. It is moreover not needed to spend software design effort to manage sleep modes.

Regarding **static power**, Tiempo applies specific techniques to optimize it. These techniques take advantage of the asynchronous logic in two ways: it utilizes the knowledge of the state of the signals to optimize the static power of the standard cells, and it utilizes the handshaking signals to control power switches.

In the context of an FSM or a microcontroller, Tiempo circuits support ultra low power communication mechanisms with peripherals.

Finally, it is important to note also that many power reduction techniques used with traditional (synchronous) designs – such as sub-threshold design techniques – can also be combined with Tiempo asynchronous designs.

4.2 Smooth current consumption and low noise

Tiempo asynchronous circuits have **very low current peaks** as it is clear from the description of the circuits' structure and behavior and as it was illustrated with the four bit adder example. Indeed, the digits of a bus are not processed at the same time by the processing parts and hence the switching activity is spread over time.

Therefore, the **noise** generated inside the circuit, in the power rails and in the substrate, is very low as it is the case for the **electromagnetic emissions**. SoC and system designers can take advantage of this property as such asynchronous digital parts do not decrease the signal to noise ratio of adjacent analog parts in a SoC and do not pollute the environment in system having electromagnetic compatibility constraints (automotive, space, avionic…).

4.3 Wide voltage range

Tiempo circuits can operate at **very low voltage** (below transistors Vth) because of the delay insensitivity property. In fact, as long as the gates are able to propagate transitions, no matter their latency, Tiempo circuits remain operational.

This allows designing very low voltage systems, powered by a single element battery or by an energy harvesting source, or even remotely powered. In all cases, the **supply voltage can be of low quality**, i.e. it does not require high quality regulation which significantly relaxes the power supply circuitry design.

Moreover, in some applications, it allows implementing safety procedures even when the power supply voltage becomes very low.

4.4 Low latency

Asynchronous circuits are self timed and, because they do not require any external clock to operate, they are very reactive systems, with a **very low latency** (no need to wait for the next clock edge). Therefore, Tiempo asynchronous design technology allows designing low latency controllers without having the drawback of generating and distributing a clock.

Hence, in a reactive system or in a feedback loop system (as it is the case in power management applications), one can take advantage of this performance to decrease the response time.

4.5 Fast wake-up

Moreover, due to their ability to process signal transitions, Tiempo asynchronous circuits **wake up in a few nanoseconds** (no need to restart or synchronize a clock).

For example, Tiempo asynchronous microcontrollers go from sleep mode to full speed operation in a few nanoseconds in response to an external event (interrupt, timer event, analog to digital converter event…).

This is particularly beneficial in case of emergency procedures, low latency processing needs and low power applications (no changing modes overhead).

For further information

Tiempo
110 rue Blaise Pascal
Bâtiment Viseo – Inovallée
38330 Montbonnot Saint Martin
FRANCE
T : +33 4 76 61 10 00
F : +33 4 76 44 19 69

www.tiempo-ic.com