VLSI Implementation of DS-CDMA Receiver Using Asynchronous Design Techniques

Bengt Oelmann, Henk Martijn, and Hannu Tenhunen

Electronic System Design Laboratory, Royal Institute of Technology,
ESDLab/KTH-Electrum. Electrum 229, S-164 40 Kista, Sweden
Email: bengt@ele.kth.se

Abstract: This paper describes the implementation of an asynchronous self-timed direct sequence spread spectrum radio receiver. The designed receiver is planned to be used in MINT [1] (Mobile InterNet Router) radio interface for broadband wireless data-communication. The receiver has been implemented in a 0.8 μm CMOS technology using a standard cell library. The final design contains approximately 100 000 transistors.

1. Introduction

One of the most critical issues in designing electronics for portable equipment is keeping the power consumption low.

Systems that are doing any kind of real-time digital signal processing, especially those which are implemented as ASICs (Application Specific Integrated Circuits), often have hard requirements on high throughput rate. CMOS technology is normally considered to be a low-power technology, since it practically draws no current when the circuits are idle. At higher clock frequencies a fast clock signal must be distributed over the entire chip. Large clock buffers are needed to drive large capacitive loads. In high-speed synchronous designs the clock distribution only will consume as much as 40% of the total power [9,10].

Asynchronous circuits have the advantage of not being dependent on global signals for synchronization. Instead, it enables all synchronization and communication to be handled at a local level. The size of the system can be scaled without effecting the speed, and introducing extra overhead in power consumption.

Asynchronous circuits offer an event-driven approach to system design. Only circuits that are doing useful work will be activated. In CMOS technology this means that idle circuits automatically get in a power-down mode when idle [8].

The benefits of using asynchronous design can be expected on global chip level. On local level asynchronous circuits have a larger area and they will also be slower. Dependent on asynchronous design style the area overhead is typically more than 40%, and the increase in delay more than 30% compared to synchronous function blocks [6,7].

In spite of the disadvantages asynchronous circuits have on local level we find it worth while investigating how it can be used for full-scale VLSI-designs in the area of communication and digital signal processing.

This paper presents a practical approach to asynchronous system design. We have been using design tools and technology that are today industry standard. We will point out in which respects these are insufficient for asynchronous design. Finally, we will discuss the performance of the asynchronous design compared to a synchronous implementation of a DS-CDMA digital receiver.

2. Digital Radio Receiver Architecture

The asynchronous implementation presented in this paper is based on a DS-CDMA receiver architecture presented in [1]. The main objectives were to develop a digital radio interface suitable for high degree of integration and with low power consumption.

One aspect of the problem is to investigate and evaluate the feasibility of using asynchronous design techniques for the digital radio receiver.

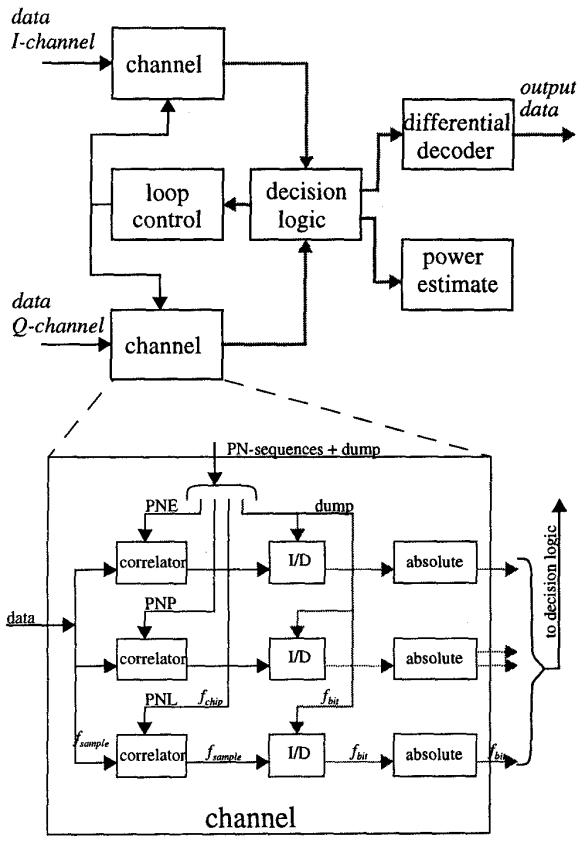
In this section we will give an overview description of the receiver and explain the working principles and functions.

From the analog front-end the receiver chip get the baseband signal that is not synchronized with the receiver. The receiver is doing synchronization and de-modulation of the signal.

Differential BPSK (binary phase shift-keyed) modulation scheme is used. The binary information that are transmitted are first differential encoded and after that multiplied with a 13 bit long pseudonoise (PN) sequence, and finally modulated using BPSK.

After down-mixing in a quadrature mixer and A/D-conversion the baseband signal is fed to the digital receiver, shown in figure 1. In the block *channel* the received data is first correlated with three shifted versions of the PN-sequence. The PN-sequence is identical to the one used when the data were transmitted.

0-7803-3002-1/95 $4.00 ©1995 IEEE



f_{sample}	sample rate	PNE = PN-sequence Early
$f_{bit}=f_{chip}/13$	bit rate	PNP = PN-sequence Punctual
$f_{chip}=f_{sample}/N$	chip rate	PNL = PN-sequence Late
N = 3.5	oversampling, times per chip	

Figure 1. Block diagram over the digital part of the receiver with the block *channel* expanded. Sample rate signals are drawn with black lines and bit rate signals are drawn with gray lines.

The correlated data are integrated over one bit period in the integrate and dump unit (I/D). After integrate and dump is done the data rate is going from sample rate down to bit rate. The channel provides signals to the decision logic. From these signals are 1) an synchronization error signal, 2) an amplitude estimate of the channel, and 3) a bit stream derived.

The synchronization is done by a tracking loop. In the block *decision logic* the synchronization error signal from the *channel* with highest amplitude is selected to be steered to the *loop control*. In the block *loop control* the PN-code is adjusted, based on the error signal, to obtain synchronization [2]. In the block *power estimate* an average value of the received power over 64 bits is calculated.

3. Asynchronous Self-Timed Circuit Design

There are basically two ways of doing asynchronous self-timed circuits. With dual-rail data encoding and completion detection it is possible to detect when the combinatorical logic has finished its computation [12]. This can be an advantage when computation time is very data-dependent and average computation time can be exploited. The extra circuits for completion detection will in many cases slow down the computation stage, especially for wide bit-parallel data where all bits must be checked. The other way is to use the bundled data approach. Here a delay element is placed in parallel with the combinatorical logic. This matched delay must be larger than the worst-case computation time for the logic. The computation in the self-timed element using bundled data convention is shown in figure 2. The computation is initiated with control signal *I* (initiate). The signal *DV* (Data Valid) is indicating that the computation has finished, and data are valid on the outputs.

In this work we have used a design methodology called micropipelines [4], and is based on data bundle signalling. We see two reasons for selecting micropipelines. Firstly, the receiver chip is working in an environment where the data arrive in fixed time steps controlled by the sample clock. The chip must be designed in a way so that it can process each of every sample, irrespective of its value, within the time of one sample clock period. Therefore it must be designed for worst-case operation. Secondly, we expect the use of micropipelines to result in more efficient circuits than any other design method [5].

A. Basic Operation

In the pipeline stage shown i figure 2. the control is very simple. The data transfer from one register to another is done in the following way. A transition (positive and negative has the same meaning) on the request wire indicates to the receiving register that new data are available on its inputs. When the data have been captured the register submits a transition on the acknowledge wire. The source register may now start the next cycle by submitting a new request.

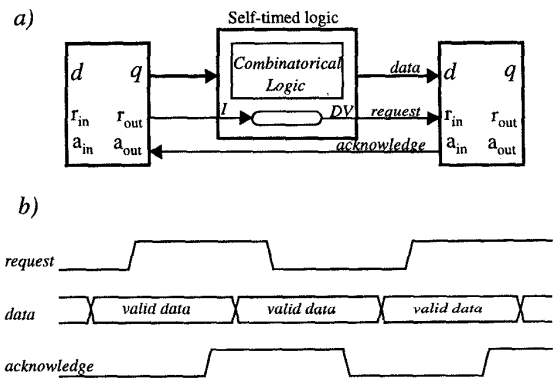


Figure 2. Bundled data convention

C. Control Modules

Control modules are used for controlling the data-flow in the system. The control paths are built from a set of primitives [4]. A few of the primitive modules we use are shown in figure 3. The delay element is inserted in the control path to meet the bundling constraints (fig. 3a). The C-element (fig 3b) is an AND-function for transitions. A transition on the output will only occur after there have been transitions on all of the inputs. The merge module (fig. 3c) merges two control paths. It acts like an OR-function for transitions. The selector module (fig. 3d) steers the input transition (e) to one of two outputs (t,f) based on the boolean value (s).

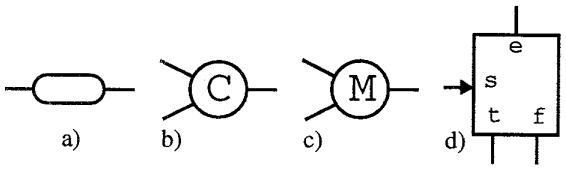


Figure 3. Control modules

E. Macro Cell Design

The receiver has been built using the basic components described above. We want to exploit the possibilities an asynchronous technique offers for this type of applications. In the block *channel* (see. fig. 1) the data-rate is reduced from sample rate down to symbol rate. It is taking place in the integrate-and-dump unit. A typical synchronous solution uses, beside the system clock, a slower clock signal for the dump-rate. In the asynchronous solution (see fig. 4) the output data rate is determined by both the input data-rate and a control bit, which is tagged to the data. The input data (`data`) are accumulated as long as their associated control bits (`dump`) are '1'.

When the `dump`-signal gets '0' the accumulated value is dumped. This means that the transition is directed to `ro_dump`. The following stage can now capture this value and acknowledge it by a transition on `ao_dump`.

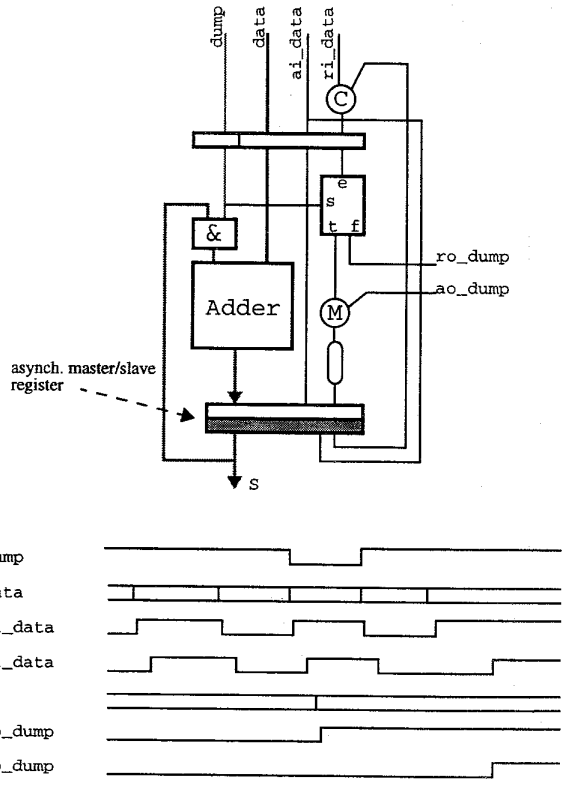


Figure 4. Integrate and dump (I/D)

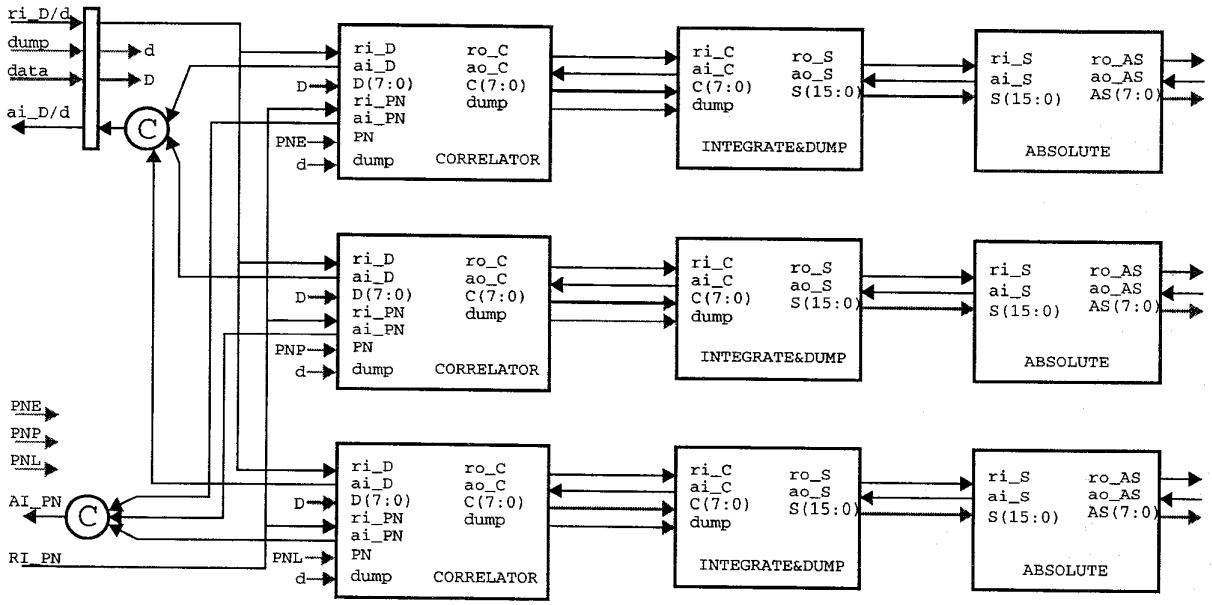


Figure 5. Asynchronous implementation of the block *channel*

4. VLSI-Implementation

Most tools and equipment available for IC-design are built for synchronous systems. For example RTL-synthesis from VHDL, ATPG (Automatic Test Pattern Generation), and ASIC-testers are only available for synchronous circuits. This becomes a problem when we use standard tools for asynchronous design. Some of the tasks that are normally done automatically must be done manually. We used a commercial standard cell library in 0.8 µm CMOS. This does not support latches and asynchronous modules. Those basic cells, specific for micropipelines, were also built from standard cells.

A. Design flow and Tools

Our design flow starts with a Matlab description. Relevant test data are created with this behavioural description. Test data are stored on files in a format that it can be used in the following stages in the design flow. The design is first partitioned into the major function units. These units are described in behavioural VHDL and interconnected in a structural fashion. At this stage only a very rough estimation of timing is done. VHDL is used for simulation and partly for design entry. The design is hierarchically refined to a point where macro cells can be identified. The macrocells have at this stage their behaviour, interfaces, and timing requirements defined in VHDL. Due to lack of synthesis tools we do the macro cells at gate-level with schematic entry. Gate-level simulations provide more detailed timing information back to the VHDL description. In many cases the behavioural description of the macro cell must be further refined to give an accurate timing model. When the design is described as an interconnection of macro cells and those are implemented at gate-level the following stages are more or less carried out automatically by different tools.

B. Simulation

A flexible simulation environment is needed in order to try out many different solutions without too much effort. We found behavioural VHDL suitable for describing the asynchronous function units. However, the structural description of the entire design becomes very long in text-format.

It is important to have accurate delay information early in the design process. With both local synchronization and communication the performance for an asynchronous system is easier to predict than for system with global structures for control and communication. Accurate timing information is only available after the actual layout has been done. Our experiences shows that delays obtained from simulation of a single macro cell are close to those obtained from simulations of the entire system.

All the delay information is located in a single VHDL record variable and is passed down in the hierarchy. This gives a great deal of flexibility when optimizing the system for speed. If one for example wants to see what impact an improvement of the 16-bit adder will have on the system performance the delay is changed for adder. The same technique is used for delay matching. The critical path is identified and close matching is only done here. At all other places we set as large margins possible without slowing down the system.

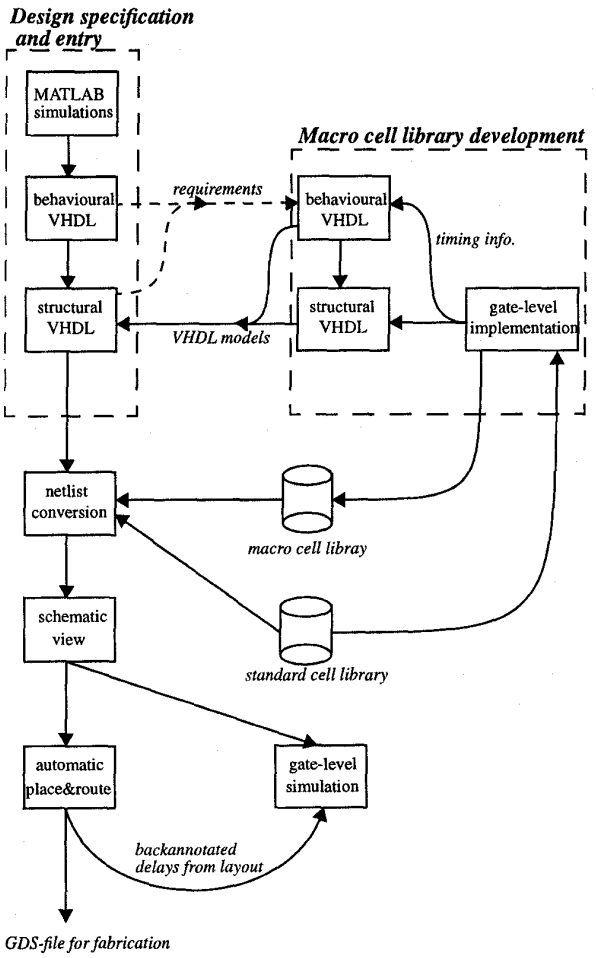


Figure 6. Design flow

C. Testing

Testing asynchronous circuits in an ASIC-tester is not a straight-forward task since they are designed for testing synchronous circuits. We added a configurable interface on-chip. In normal operation mode it has the asynchronous handshake protocol. In test mode it externally acts like a synchronous circuit. The ASIC-tester can now be used for testing the entire circuit except for the asynchronous interface that has been re-configured. We use a scan-element, that beside the function described above, also makes it possible to observe some internal control lines. We have only inserted this element at a few places since it is too complex to be inserted for every control signal.

5. Results and Conclusions

A fully asynchronous DS-CDMA radio receiver has been implemented by using the methods described in this paper. The performance and characteristics of the receiver chip are summarized in Table 1.

Table 1. Summary of receiver characteristics

Max. Sample rate @ 5V	48 MSample/s
Max. Data rate	1 Mbit/s
Chip area core / total	21 / 30 mm^2
Power consumption (IO-circuits excluded)	500 mW
Number of transistors	97 645
Number of standard cells	18 700
IC Technology	0.8 µm CMOS
Package	40 pin DIL

It is hard to compare the performance of a reasonable complex asynchronous system to its synchronous counterpart due to the fact that a large scale design is rarely done as one synchronous and one asynchronous version. A fair comparison can only been done if the functions of both systems are the same and the same technology is used. We compare our design with a strict synchronous receiver with the same functions but implemented in FPGA:s [3]. In order to make a comparison based on the same technology we convert the synchronous design to the same technology we use for the asynchronous one. We base our results in Table 2. on SPICE simulations of different function blocks and a model for the clock distribution net [11]. The estimated improvement of using special cells for latches and asynchronous control logic can be seen in the middle column of Table 2.

Table 2. Comparison of different implementation techniques

	Asynch. standard cell only	**Asynch.** special standard cells	**Synchronous**
Speed [Msample/s]	48	48	70
Area (core) [mm^2]	21	16	13
Power @ 48Ms/s [mW]	500	400	350

It is obvious that the synchronous version would be superior either we consider speed, area, or power. The removal of the global clock tree and keeping the different parts running at lowest possible rate did not compensate for the increase in circuit overhead for handling local synchronization.

It is harder to do asynchronous design than clocked synchronous circuit design. Well-established design methods and mature design tools for design automization is shortening the design time for synchronous design. But asynchronous design techniques have many nice properties. During our work we appreciated the modularity. It is easy to specify and understand the interface of each module.

6. References

1. D. Kerek, H. Olson, H. Tenhunen, G. Maguire, F. Reichert, "Direct Sequene CDMA Technology and its Applications to Future Portable Multimedia Communication Systems", IEEE ISSSTA '94, Oulu, Finland. pp. 445-449.
2. H. Olsson, D. Kerek, H. Tenhunen, "Direct Sequence Spectrum Digital Radio Performance Analysis with Simulation", SIMS'94. pp. 462-466.
3. T. Saluvere, D. Kerek, H. Tenhunen, "Direct Sequence Spread Spectrum Digital Radio DSP prototyping using Xilinx FPGAs". 4th Int. Workshop for Field Programmable Logic and Applications (FPL-94), Praha, 1994.
4. I.E. Sutherland, "Micropipelines", Communications of the ACM, June 1989, vol. 32, pp.720-738.
5. J. Sparso, C. Nielsen, L. Nielsen, and J. Staunstrup, "Design of Self-Timed Multipliers: A Comparison", Proc. of IFIP TC10/WG10.5 Working Conference on Asynchronous Design Methodologies, Manchester, England, 31 March - 2 April 1993, IFIP Transactions, vol. A-28, pp. 165-180.
6. R. Auletta, B. Reese, and C. Traver, "A Comparison of synchronous and Asynchronous FSMD Designs", ICCD'93, pp. 178-182.
7. B. Oelmann, and H. Tenhunen, "Micropipelined Multiplier Design Analysis", Proc. of Norchip Conference 1994, p.187.
8. K. Berkel et al., "Asynchronous Circuits for Low Power: A DCC Error Corrector", IEEE Design&Test of Computers, summer 1994.
9. D. Liu, and C. Svensson, "Power Estimation in CMOS VLSI Chips", IEEE Journal of Solid-State Circuits, vol. 29, no. 6, June 1994.
10. D.W Dobberpuhl et al., "A 200MHz 64-b Dual-Issue CMOS Microprocessor", IEEE Journal of Solid-State Circuits, vol. 27, no. 11, June 1992.
11. H.B Bakoglu, "Circuits, Interconnections, and Packaging for VLSI", chapter 9, Addison-Wesley, 1990.
12. C.L Seitz "System Timing", in "Introduction to VLSI Systems", C.A Mead and L-A Conway, Eds., Addison-Wesley, 1980.