Low Power IEEE 802.15.4a UWB Digital Rx Baseband Architecture

Michael De Nil, Ben Busze, Alex Young, Dries Neirynck, Hans Pflug,
Kathleen Philips, Jos Huisken, Jan Stuyt and Harmke de Groot

Holst Centre / imec
High Tech Campus 31
Eindhoven, The Netherlands

Abstract - **This paper presents a low power ultra-wide band digital receiver baseband architecture for handling IEEE 802.15.4a packets in real-time. Real-time processing allows duty cycling of the analog frontend, which is key to achieve low power consumption. The architecture consists of a programmable application specific instruction set processor and a set of application specific integrated circuits. The design is split up into multiple clock domains to meet timing requirements of up to 499.2MHz. The architecture runs in real-time on FPGA, and is synthesizable for 90nm CMOS at 0.84V. This results in a low power flexible C programmable baseband architecture, which is ideal for algorithm development and tuning.**

I. INTRODUCTION

Ultra-wide band (UWB) has many attributes that make it an attractive communication method for battery operated and even batteryless devices [1]. In order to achieve low energy consumption, duty cycling of the analog frontend is key [2]. To turn on and off the analog frontend during packet reception, a closed loop system architecture is required, where a baseband architecture needs to process incoming data in real-time.

This paper describes a programmable digital baseband architecture for an IEEE 802.15.4a UWB receiver (as depicted in Fig. 1). In this receiver, the incoming pulses are down converted to baseband, sampled by two parallel 5-bit ADCs at 499.2 MS/s and processed digitally. It has been shown that digital UWB receivers perform significantly better than analog correlation receivers [2]. On the other hand, given the sample rate of 499.2 MHz, many digital architectures suffer from high power consumption. Low power implementations do exist [7], but are not programmable and thus have limited flexibility.

In the past few years imec designed two ASIP (Application Specific Instruction Set Processor) based solutions for processing UWB packets [3] [4]. To meet the timing requirements, these processors had multiple scalar and vector issue slots for parallel processing and were clocked at a frequency of up to 220 MHz. This resulted in a rather big and power consuming architecture, which could only support a subset of the modes defined in the IEEE 802.15.4a standard. The main timing and power bottleneck for both designs was signal detection and synchronization. This caused an average power consumption of 10-20mW for the processor and memories. Because of the high frequency used, these designs could not run in real time on an FPGA.

Due to these limitations a new architecture was designed, which combines an ASIP with several ASIC components. In order to lower power consumption, and to be able to map the design on FPGA, several clock domains were introduced, such that only a limited set of ASIC components needs to run at a high clock frequency.

The paper is structured as follows. In Section II we introduce the structure of a typical real time algorithm for IEEE 802.15.4a packet decoding. Section III describes the proposed baseband architecture. Sections IV and V describe how the architecture was mapped on a FPGA and synthesized for low voltage 90nm CMOS. In Section VI we list the conclusions.

II. ALGORITHM

An IEEE 802.15.4a UWB frame (as illustrated in Fig. 2) has 3 major components: the Synchronization Header (SHR), the PHY Header (PHR) and the PHY Service Data Unit (PSDU) [5]. The SHR consists of a given sequence of isolated pulses. During PHR and PSDU data is transmitted in bursts. A real-time algorithm for receiving IEEE 802.15.4a UWB frames will consist of the following main components [6]:

A. Signal detection and isolated pulse synchronization

Detect if there are any isolated pulses in the air. If so, synchronize (pulse level) with the transmitter and start capturing isolated pulses. If not, turn off the analog frontend and go to sleep for a duration shorter than the that of the SHR.

During this mode, the analog frontend is activated and all data is captured and processed at 499.2MHz. This mode will thus consume a lot of power, and should be turned off as quickly as possible.

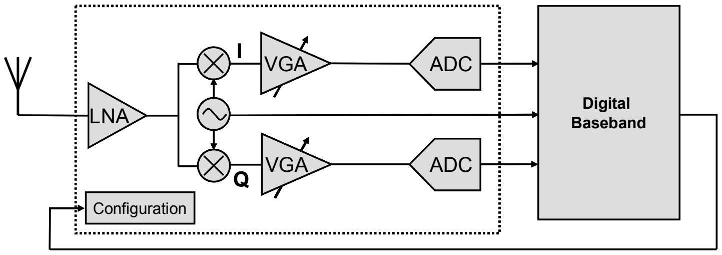

Figure 1. UWB receiver block diagram.

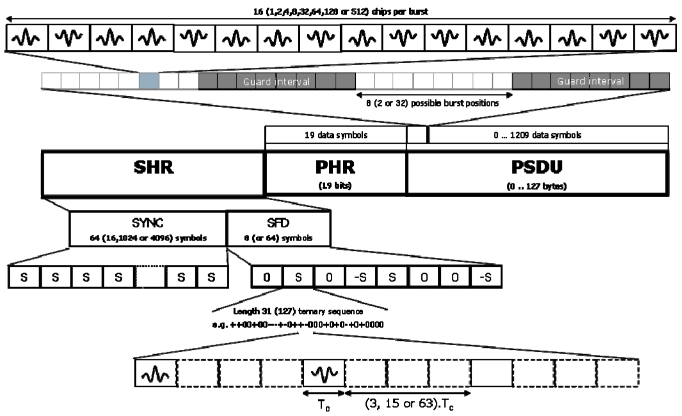

Figure 2. IEEE 802.15.4a UWB packet format.

B. *Isolated pulse storage, SHR symbol synchronization and SFD detection*

Store isolated pulses and detect the start of an SHR symbol, which has a length of 31 or 127 isolated pulses. Synchronize (SHR symbol level) with the transmitter, and start SFD detection. Once the SFD is found, activate burst reception.

Depending on the data rate, isolated pulses arrive at 124.8MHz, 31.2MHz or 7.8MHz. The current analog frontend supports duty cycling for the lower repetition frequencies.

C. *Burst reception*

Per PHR/PSDU symbol, compare the incoming data at the 2 possible burst positions and extract the resulting bit. The PHR specifies the mode and the length of the PSDU. The PSDU contains the actual data.

Depending on the data rate, symbols will arrive at a rate of 3.9MHz, 15.6MHz or 62.4MHz. For every symbol, 2 bursts of length 1, 2, 4, 8, 16, 32, 64, 128 or 512 pulses need to be captured and compared.

III. Digital Architecture

A. *Overview*

Fig. 3 shows an overview of the baseband architecture. The baseband architecture has a control interface to the analog frontend and is connected to two 5 bit ADCs clocked at 499.2MHz. Inside the baseband architecture, the ADC data is parallelized and processed by a set of heavily pipelined ASIC blocks at half of the ADC clock frequency. The system also contains a Target based vector ASIP, a 16 bit program memory, a 16 bit data memory, a 160 bit (32 x 5 bit) vector memory, a JTAG debug interface, an SPI slave port, a vector FIFO block and a central control block.

B. *Clock domains*

In order to meet timing requirements, reduce power consumption and support external interfaces, the design was split up into 6 clock domains. For every clock domain crossing, synchronization flip flops or asynchronous FIFOs are used.

1. Master clock (124.8MHz): this clock is always on during reception, and will be used to turn on and off other clock domains. Internally only the control block and vector FIFO interface are clocked with the master clock.

2. ADC clock (499.2MHz): this clock originates in the analog frontend where it is used as the sample clock for the ADCs, and is generated from the RF carrier frequency. It is unrelated to the 124.8MHz master clock, and can be turned on and off by the control block. In the digital domain, this clock is only used by the serial-to-parallel conversion block.

3. ADC clock/2 (249.6MHz): this clock is derived from the 499.2MHz ADC clock. Most ASIC components use this clock and will thus only be active when the ADC clock is active.

4. ASIP clock (~50MHz): this clock drives the ASIP core and its synchronous memories. This clock frequency can either come from an external source, can be generated on chip such that the frequency is scalable, or can be divided from the master clock.

5. JTAG clock (~20MHz): this is the external clock of the JTAG debugger, which is connected to the JTAG tap interface. This clock will only be used while debugging.

6. SPI clock (~10MHz): this is the external clock of an external SPI controller. This clock will only be used when an external controller is accessing the ASIP memories or the control interface.

C. *Energy function*

This block contains a power efficient implementation of an energy extract (vector magnitude) function. The ideal vector magnitude calculation would be the square root of $i^2 + q^2$.

By analyzing the tradeoff between complexity and performance, we concluded that the following blocks could be used, exploiting the narrow bit-widths in the system: a 4-bit x^2 function (one each for I and Q inputs), a 6-bit adder, a half-adder and a reduced complexity 9-bit square root function.

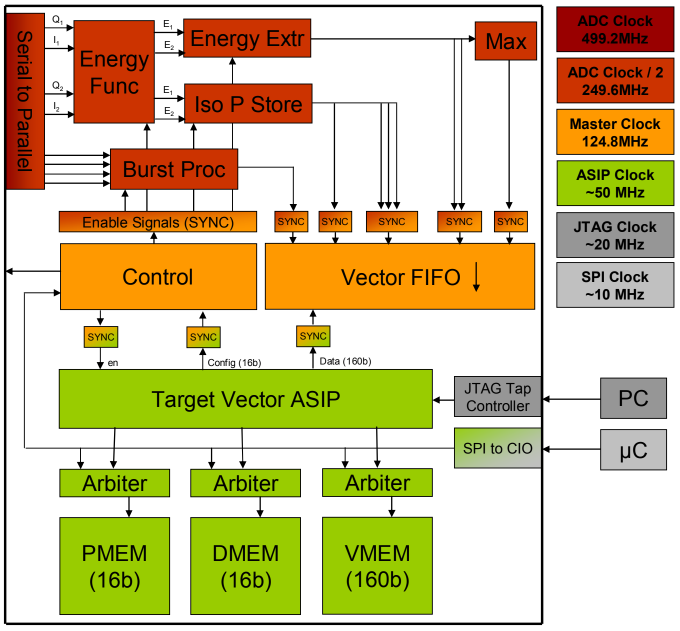

Figure 3. Baseband architecture overview.

D. *Energy extract block*

The energy extract block is activated during signal detection and isolated pulse synchronization. The behavior of this block is illustrated in Table I. The block creates vectors of 64 elements out of the results of the energy function. It then accumulates the energy over a number of vectors. This number of accumulations is programmable and can be changed at run-time by the algorithm. A small number of accumulations will result in lower power consumption and a faster synchronization. A greater number of accumulations might be required for noisy channels.

The result of the accumulations is a vector of 64 elements of each 12 bit. This result will then be shifted a programmable number of times to the right, after which the 5 least significant bits of all 64 elements are returned. The data is communicated to the processor via two 32x5b vector FIFO slots, and to the Max component. Depending on the mode, the result should contain 1, 4 or 16 peak values.

E. *Max component*

The Max component receives data from the energy extract block and looks for energy peaks. The behavior of this component is illustrated in Fig 4. For every four chips, it extracts the maximum value, and its location (0, 1, 2 or 3). It will then format this data in a single vector word, which is communicated to the processor via a 32x5b vector FIFO slot. Depending on the data rate, the results should contain 1 peak value with a certain offset, 4 peak values with the same offset, or 16 peak values with the same offset.

F. *Isolated pulse storage block*

Once a signal is detected and the receiver has locked on the isolated pulse, the isolated pulse storage block can be activated. This block will store the energy of selected isolated pulses during synchronization. Isolated pulses can be selected by setting a 64 bit pulse mask, such that 1 out of every 4, 16 or 64 incoming pulses is stored. A programmable number of pulses will be captured before they are sent to the processor via one or more vector FIFO's. Typically either 31 or 127 pulses will be captured, as this is the amount of pulses in 1 synchronization symbol.

These symbols are then processed on the ASIP, which will synchronize on symbol level, and will start looking for the SFD.

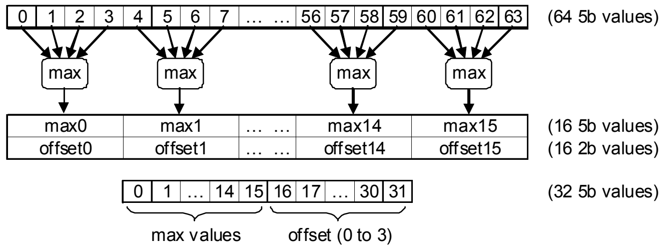

Figure 4 Max component.

TABLE I
ENERGY EXTRACT BLOCK

	mag_0	mag_1	…	mag_{62}	mag_{63}	64x5b vector
						n max 127
	mag_{64*n+0}	mag_{64*n+1}	…	$mag_{64*n+62}$	$mag_{64*n+63}$	64x5b vector
+	sum_0	sum_1	…	sum_{62}	sum_{63}	64x12b vector
>>	$result_0$	$result_1$	…	$result_{62}$	$result_{63}$	64x5b values

G. *Burst process block*

The burst process block is activated when the SFD is detected. For every PHR and PSDU symbol a programmable number of chips are captured at two possible burst positions.. After correlation with the expected burst sequence, these are then compared with each other to determine the resulting output bit. After processing a programmable number of symbols, the data is transferred to the ASIP via a vector FIFO.

H. *Control block*

The control block contains a number of 16 bit configuration registers which can be read and written via the processor configuration memory interface, or externally via the SPI interface. Via these configuration registers all ASIC blocks can be controlled, i.e. configured or turned on/off. The control block also contains a linear feedback shift register (LFSR), which will determine the position of the next burst, and the expected burst sequence.

I. *Vector FIFO interface*

The vector FIFO interface has a number of parallel handshake based write ports connected to the ASIC components, and a single addressable read interface connected to the processor vector FIFO interface. This allows the ASIP to be duty cycled while waiting for data.

J. *Vector ASIP*

The system contains a single issue slot application specific instruction set vector processor. The processor is designed using the Target IP Designer, and is fully C programmable. The processor also has a JTAG debug interface which allows program loading, setting of breakpoints and single stepping through the code via the ChessDE GUI.

The basic scalar instructions on the processor are taken from the Target Base processor. Application specific vector instructions were added to reduce the cycle count. The processor has 5 memory interfaces listed in Table II.

TABLE II
ASIP MEMORY INTERFACES

	Type	Width	Addresses	R/W	SPI	JTAG
PMEM	Sync	16 bit	4x2048	RW	x	x
DMEM	Sync	16 bit	4x2048	RW	x	x
CMEM	Async	16 bit	32	RW	x	x
VMEM	Sync	32x5 bit	512	RW	x	
VFIFO	Async	32x5 bit	16	R		

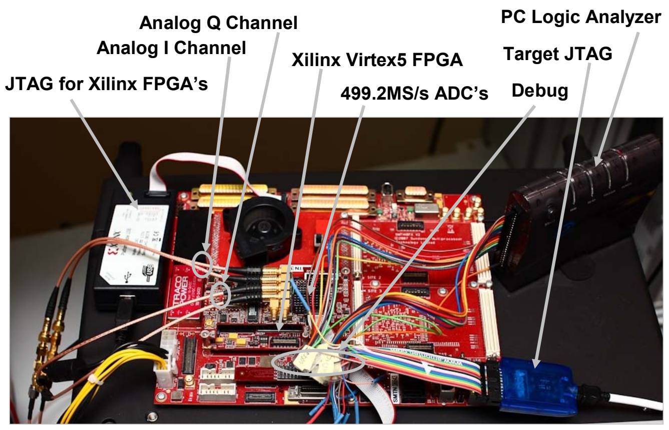

Figure 5. FPGA lab setup.

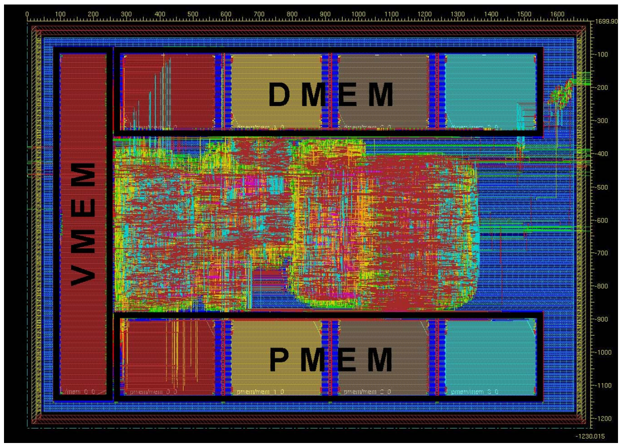

Figure 6 Preliminary IC layout.

IV. FPGA Implementation

The complete architecture has been demonstrated on a Sundance FPGA platform, as shown on Fig. 5. The platform consists of an SMT148-FX carrier board, an SMT351T-LX110-1 FPGA module, an SMT391 ADC module and an SMT599 GPIO module. The digital baseband is mapped on a Xilinx Virtex-5 FPGA and is connected to a dual channel 8 bit Atmel ADC clocked at 499.2MHz. The architecture is programmable through SPI with a microcontroller or through JTAG with a PC.

For synthesis Synplify Pro was used, place and route was performed with Xilinx ISE. To meet timing, constraints and clock relationships were specified in SDC format for synthesis and in UCF format for place and route.

V. From FPGA to Silicon

The system is designed to work at 0.84V in a TSMC 90nm LP process. The RTL code, which was mapped onto standard cells, is identical to the RTL that was mapped onto the FPGA, expect for the ADC interface.

To verify that the design meets the timing requirements it was synthesized, placed, and routed using TSMC standard cells characterized at 0.84V and Virage memories for VMEM, PMEM and DMEM characterized at 1.2V. A preliminary layout is shown in Fig. 6. The design is structured to allow power gating of both memory banks as well as logical blocks. This will reduce static power consumption.

VI. Conclusion

In this paper a programmable architecture for real time IEEE 802.15.4a UWB Rx baseband processing was presented. Real time processing is key for duty cycling the analog frontend, and thus for low power consumption. Both real-time behavior on FPGA and 90nm CMOS, as well as low power consumption on 90nm CMOS was achieved by splitting the design into multiple clock domains.

The design is controlled by an ASIP, and is thus fully programmable in C. Since mapping a new application on the system only takes seconds, it is ideal for algorithm development and tuning. Since the design runs in real time on FPGA, the ASIC and even the ASIP can be modified, mapped and tested on FPGA within hours.

References

[1] Chandrakasan et al., "Low-Power Impulse UWB Architectures and Circuits," *Proceedings of the IEEE, Vol. 97, No. 2, February 2009*.

[2] J. Ryckaert et al., "A CMOS Ultra-Wideband Receiver for Low Data-Rate Communication, *J. Solid-State Circuits, vol.42, no. 11, Nov 2007*.

[3] Jochem Govers, Jos Huisken, Mladen Berekovic, Olivier Rousseaux, Frank Bouwens, Michael De Nil and Jef van Meerbergen, "Implementation of an UWB Impulse-Radio Acquisition and Despreading Algorithm on a Low Power ASIP," *HiPEAC 2008*.

[4] Christian Bachmann, Andreas Genser, Jos Hulzink, Mladen Berekovic and Christian Steger, "A Low-Power ASIP for IEEE 802.15.4a Ultra-Wideband Impulse Radio Baseband Processing," *DATE 2009*.

[5] IEEE Standard 802.15.4a-2007.

[6] C. Desset, M. Badaroglu, J. Ryckaert, and B. Van Poucke, "UWB search strategies for minimal-length preamble and a low-complexity analog receiver," *Proc. IEEE Workshop on Statistical Signal Process. Advances in Wireless Commun. (SPAWC), Cannes, France, July. 2006*.

[7] Marian Verhelst, Nick Van Helleputte, Georges Gielen and Wim Dehaene, "A Reconfigurable, 0.13µm CMOS 110pJ/pulse, Fully Integrated IR-UWB Receiver for Communication and Sub-cm Ranging," *ISSCC 2009*.