Fully flexible baseband DSP processors for future SDR/JTRS

Dake Liu, Eric Tell, Anders Nilsson (J)[1] and Ingemar Söderquist[1, 2]
[1]*Dept of Electrical Engineering, Linköping University, Linköping, Sweden*
Email: {dake, erite, andni, ingemars}@isy.liu.se
[2]*Saab AB, Linköping, Sweden*
Email: ingemar.soderquist@saabtech.se

Abstract. A real SDR/JTRS (Software Defined Radio / Joint Tactical Radio System) transceiver can be linked to any public or private radio system. The transceiver baseband processor must adapt to the actual radio channel coding and shall manage ISI, mobility, synchronization, and different interferers. Current baseband solutions are unable to fulfil these requirements because of limited flexibility.

A fully programmable baseband DSP processor (BBP) was designed by a research team in Linköping University, Sweden and fabricated using 0.18 µm digital CMOS technology. The BBP can handle 802.11a, b, g, and GSM/GPRS by running different software on the same processor. The power consumption is less than 100mW running 802.11a peak reception mode at frequency 160MHz. The die size is only 2.9mm^2, about half compared to non-programmable solutions thanks to the high hardware utilization. The new BBP facilitates a fully programmable baseband processor for any current available radio link standard for SDR and JTRS.

1. Introduction

Three kinds of processors can be found in a communication terminal, DSP baseband processors (BBP), DSP application processors (APP), and micro controllers (MCU). BBP covers all operations between ADC/DAC and the MAC (Medium Access Control) layer. The function coverage of a baseband processor is given in Figure 1.

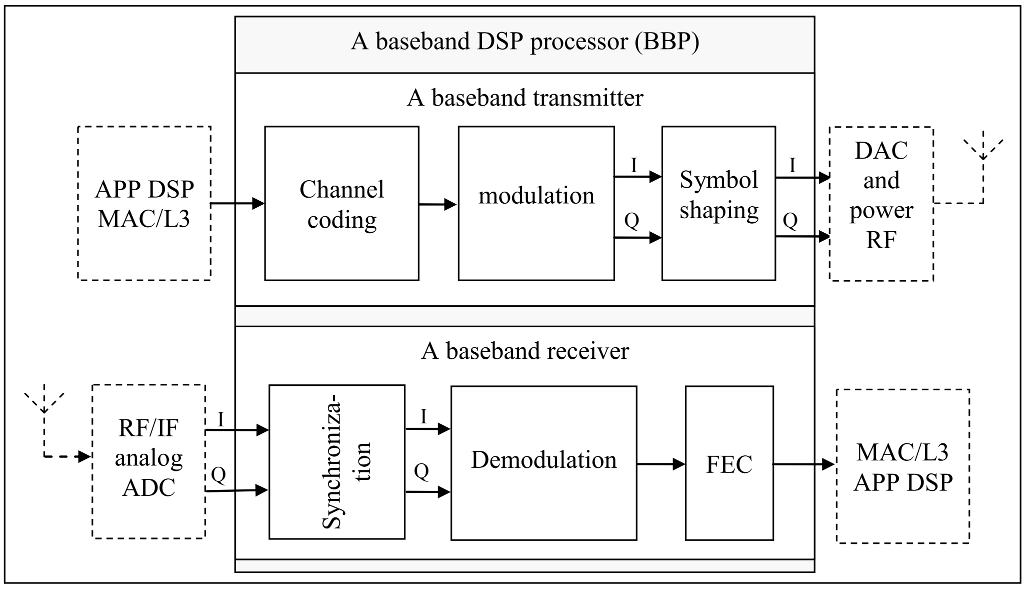

Figure 1 A BBP in a radio communication transceiver

From different way of modulation and demodulation, a classical BBP should be designed either for high speed simple symbol radio system (e.g. CDMA) or for low speed symbol carrying many bites (e.g. OFDM). A BBP receiver can also be implemented based on synchronous reception or an asynchronous reception. However, for SDR, all different classical architectures must be converged into one processor. Furthermore, a SDR BBP must take heavy tasks in a communication system and requires high performance because:

1. BBP must manage heavy computing for radio channel estimation and adaptation in order to manage the ISI (Inter Symbol Interference) problem.
2. The heavy task of channel estimation and adaptation must be repeated frequently because the estimated radio channel changes fast under high mobility conditions.
3. For synchronous receiving, the BBP must lock the de-mapping frequency and phase to the frequency and phase of the DA converter of the transmitter.
4. BBP must manage the baseband DSP processing on both word level and bit level with low computing latency to eliminate burst and other random interferences.
5. Heavy computing of channel decoding algorithms must be supported by BBP to recover data from noisy channels, for example Viterbi or Turbo decoders.
6. To design a practical baseband processor, we must use limited processing precision and adapt ultra high dynamic range. Complicated and fast gain control is required.
7. We must schedule executions of BBP tasks in a short time interval because the computing latency is limited by the MAC/L3 requirements.
8. For software defined radio, the BBP must be able to manage both synchronous reception and asynchronous reception.

It is difficult to design a high quality Baseband IC to fulfil requirements listed above, especially with high performance and low power consumption. Classical Baseband IC was not programmable. By mapping tasks in every step in the receiver and the transmitter, baseband DSP use to be implemented as an ASIC with many modules performing computing algorithms and these modules are controlled by a micro controller (for example ARM7) or a programmable DSP processors (for example TI C55).

A non-programmable digital baseband ASIC consists of many complex multiplier-accumulators (CMAC), many look up tables (LUT), many small memories, and many FSMs (Finite State Machines). A direct ASIC solution is implemented by mapping algorithm dataflow into circuits. Hardware reuse and multiplexing is almost impossible because the complexity of an ASIC must be limited and verification time of an ASIC must be acceptable. An example of implementing IEEE802.11a/b into an ASIC requires up to 8 real-valued MACs, 17 CMACs, 15 LUTs, and many small memory blocks. Including control and memory buffers, the equivalent gate count of this solution could be up to 500k gates, yet it is neither flexible nor scalable.

The assembly instruction set will be investigated in chapter 2. The processor core architecture and hardware accelerators will also be discussed and investigated. Furthermore task and loop level scheduling including hardware acceleration will be discussed. Finally, conclusions are drawn based upon silicon test chip.

2. Instruction set architecture of our programmable BBP

We investigated instruction set architecture of programmable BBP following eight listed requirements in the previous chapter. Two datapaths are necessary in a BBP. The complex datapath is designed to accept complex data format and to execute complex convolution and complex FFT translations. The real data path is designed to manage miscellaneous functions including the BBP super-mode and SW FSM, the mapping / de-mapping, AGC and AFC, channel decoding, and data quality control.

IEEE802.11a b and g standards were first used as requirements to form our BBP instruction set architecture. At least 4000 MIPS were expected for channel estimation or payload reception. 100% programmability is not feasible. Fortunately, a BBP does not need 100% programmability. The same as designing other ASIP (Application Specific Instruction set Processor) the essential issue in design of programmable BBP is to find the right partition of programmability and configurability. Taking IEEE802.11a/b as examples, we list tasks in following table 1. In this table, the MIPS cost is defined as the number of instructions required to process a certain task divided by scheduled time for that task. The MIPS costs in table below are based on a complex datapath being able to run one complex arithmetic operation (e.g. Complex MAC) per cycle. The cost does not include hidden MIPS costs of memory access for operands and results.

Table 1 Partition of main transceiver algorithms and map them to processor or accelerators

Tasks	Function	Algorithms	MIPS a/b	HW usage
Receive Filter	Anti alias & BPF	Complex FIR convolution	1200/440	accelerator
Packet detection	Start synchronization	Auto correlation $\Sigma\, x_i{*}x_{i-\tau}$	80/44	processor
Energy detection	Antenna diversity	Auto Correlation	80/44	processor
Freq. offset est.	Rotor coefficient	FFT / phase decision	100/44	processor
Ch-estimation	channel modelling	FFT, FSM	160/80	processor
Payload Rotor	Rotor	Vector product	40/22	processor
IFFT/FFT	IFFT/FFT	64 points FFT and IFFT	60/-	processor
Normalization	Normalization	1/x and vector product	40/-	processor
Ch-compensation	Sub carrier compensate	LMS adaptive algorithms	80/20	processor
De-mapping	symbols to bits	Decision FSM	40/22	accelerator
De-interleaving	Permutation	Permutation algorithm	260/-	accelerator
Descrambling	Bit permutation	$Y = x^7+x^4+1$, algorithm	80/16	accelerator
Viterbi decoding	Viterbi decoding	ACS and acceleration	1700/-	accelerator
Channel coding	Convolution coding	Convolutional coding	160/-	accelerator
Mapping	Modulation	Look up table	80/11	accelerator
Symbol shaping	Low pass filter	Two Real FIR convolution	1200/220	accelerator
RAKE receiver	4 fingers receiver	FIFO buffer / sum up taps	-/242	accelerator
De-spread	Despread	11 taps convolution	-/88	processor
CCK decoding	CCK decoding	DWT decoder	-/280	accelerator
CCK coding	Modulation	CCK FSM	-/70	processor
Spreading	Spread	Vector multiplication	11	processor

Notice that we actually do not need to execute all tasks in the table above at the same time. Algorithms are actually distributed into eight different operating modes in Figure 2. An example of a timing critical path is the processing of the long preamble in IEEE802.11a. This requires at least 20k clock cycles to be performed within 8 microseconds. This equals at least 2500 MIPS (not including FEC/Viterbi). Running a processor at the frequency much more than 2500 MHz is obviously impossible and we must either lower the MIPS cost or must perform several operations in parallel. There are three ways to lower the MIPS requirements: Reducing or hidden the memory access cost; instruction level acceleration; and moving tasks out of the processor core by hardware acceleration.

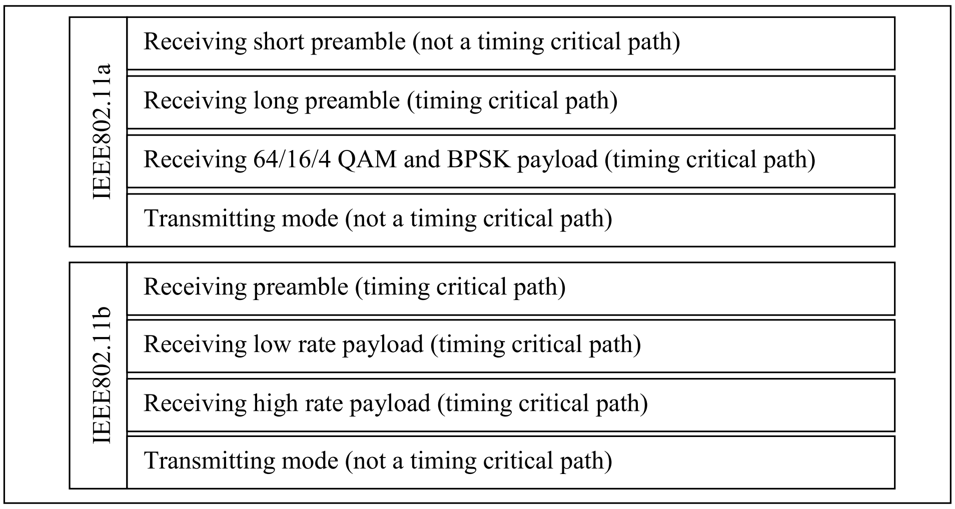
IEEE802.11a
Receiving short preamble (not a timing critical path)
Receiving long preamble (timing critical path)
Receiving 64/16/4 QAM and BPSK payload (timing critical path)
Transmitting mode (not a timing critical path)
IEEE802.11b
Receiving preamble (timing critical path)
Receiving low rate payload (timing critical path)
Receiving high rate payload (timing critical path)
Transmitting mode (not a timing critical path)

Figure 2 Scheduling plan for IEEE802.11a/b baseband DSP

To accelerate tasks on instruction level, we need to conduct benchmarking carefully. The basic rule for instruction optimization is to find 10%-90% locality, i.e. to find 10% of instructions running during 90% of the time. The result is given in the following table:

Table 2: seven most used instructions for a baseband DSP processor

Instructions	Functional specification
Complex FFT or IFFT	A butterfly in one clock cycle + bit reversal addressing
Complex convolution with implied modulo addressing	For I = 1 to N do {Complex REG <= Complex REG + V1[i] * (Conjugate of) V1(or 2)[i]}
Complex vector product	For I = 1 to N do {V3 [i] <= V1[i] * (Conjugate of)V2[i]}
Sum energy of a vector	For I = 1 to N do {accumulator <= Re $(V[i]^2)$+Im $(V[i]^2)$}
ABS approximation of complex data	REG <= Approximation of a complex $(\text{Real}^2 + \text{Imaginery}^2)^{1/2}$
Fast modulo FIFO memory access	Memory access with implied modulo addressing
Look up table (single step / vector mode)	REG2 <= Memory [Segment + REG1]

By using a CMAC (Complex multiplication and accumulation unit) in the datapath and smart address generators, the MIPS requirement can be lowered by about 6 times. This decrease the MIPS cost from ~2500 MIPS to ~420 (FEC arithmetic not included). Memory accesses still consume about 30% to 50% of the total MIPS. By using memory sharing and DMA techniques we further reduce the MIPS cost by 30%. By these measures we have reached a final MIPS costs to ~300 MIPS. All opportunities of hardware acceleration were carefully investigated under the flexibility requirement. All recurring fixed functions were identified and hardware accelerators were designed with enough configurability. After hardware acceleration, the MIPS cost of preamble processing was only about 150 MIPS including MIPS consumed by the super mode program.

Table 3 Accelerators make the programmability feasible

Accelerators	Functions	Usage
FIR filter	Configurable complex / dual integer data types	Anti aliasing and symbol shaping
1/x circuit	Accelerates 1/x and gives a result per cycle	Normalization
De-mapper	Accelerates it and gives one result per cycle	BPSK/QPSK/16QAM/64QAM
Inerleaver	Accelerate permutation and its addressing	Block interleaving de-interleaving
CRC/Scrambler	A configurable CRC/SCR polynomial circuit	CRC, scrambling, descrambling
Conv encoder	Circuit to accept configurable conv polynomial	Encoding convolution codes
Viterbi decoder	Trellis decoding, ACS (Add compare select)	Decoding convolution codes
DMA manager	Manage DMA and bus connector of memories	SOC platform

3. Architecture and implementation of our programmable BBP

A programmable baseband DSP processor and its CMAC are given in Figure 3 and figure 4. The Silicon implementation is shown at the left part of figure 5 and the system level firmware scheduling is shown at the right part of figure 5. The scheduling shown in figure 5 is for reception of 64QAM modulation and it consumes the most MIPS. The chip was fabricated using 0.18um digital CMOS silicon with 6 metal layers. To be able to share the digital system clock with the AD/DA converters, the processor runs on 154MHz for 11b and 160MHz for 11a. The die size of the implemented BBP is 2.8 mm^2, and the chip size of the BBP is 4.9 mm^2 including 120 pins. This silicon cost does not include a Viterbi decoder. With Viterbi decoder, the total die area is estimated to 4 mm^2 including all memories.

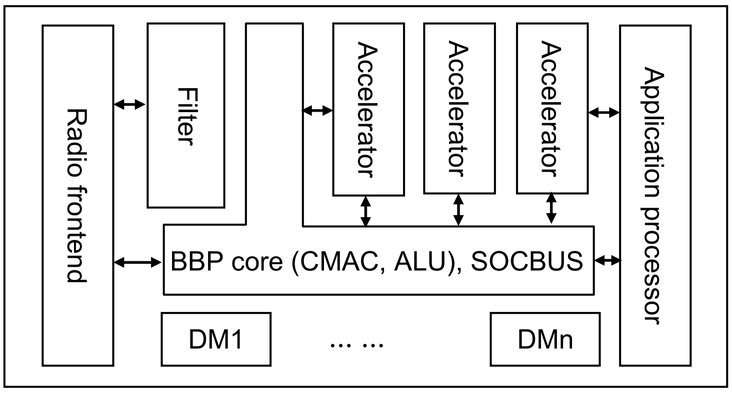

Figure 3 Architecture of a programmable BBP and its CMAC

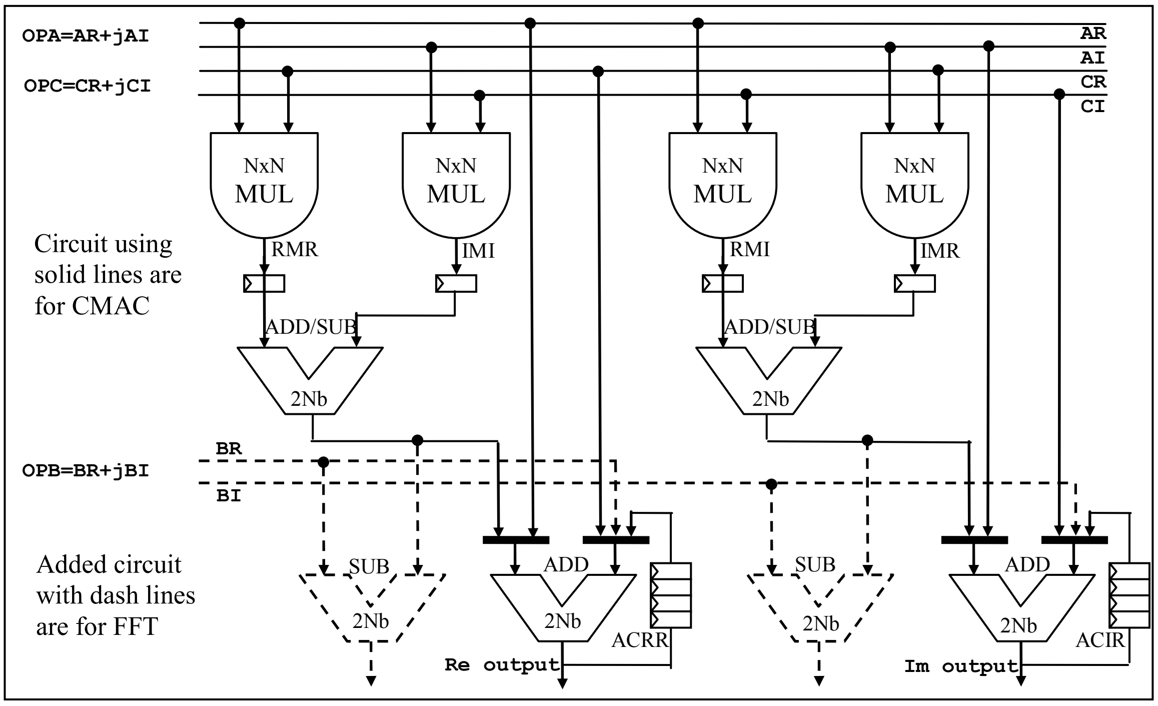

Figure 4 Simplified CMAC datapath

.

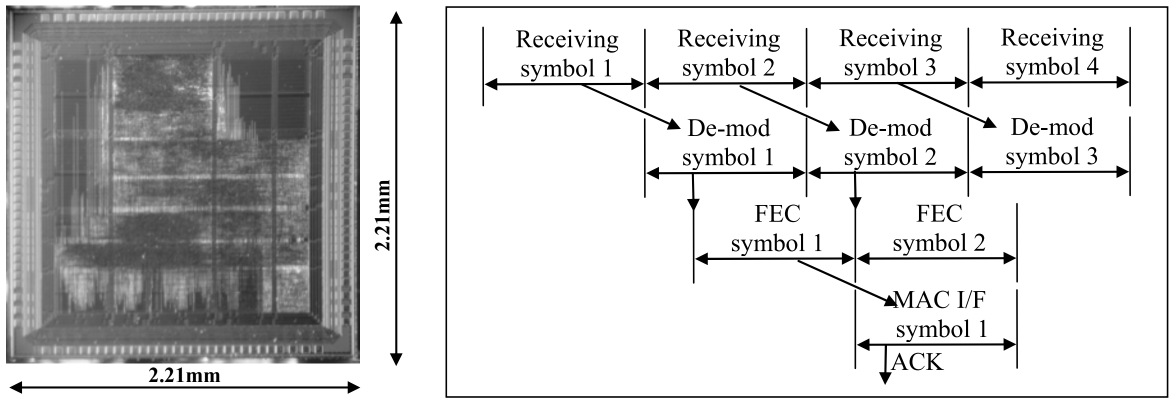

Figure 5 silicon implementation and task level (inter symbol) scheduling

4. Conclusions

Programmable baseband processors are required in SDR systems. By using advanced hardware parallelization techniques, hardware acceleration techniques, memory sharing technique, and custom static scheduling in super mode SW, we have demonstrated a programmable baseband DSP processor for IEEE802.11a/b/g as well GSM/GPRS. The silicon cost is considerably less comparing to custom ASIC solutions. The power consumption is not higher than the power consumption of an ASIC when an operand masking technique is applied in the datapath.

5. Acknowledgements and Future work

The research [1-4] has been partly financed by SSF via Stringent centre of Linköping University and partly financed by Socware (ISA) Sweden.

The flexibility of the demonstrated BBP is not enough to support even more demanding wireless standards (TD-SCDMA or WCDMA for example). We are currently working on a second demonstrator based on a VLIW/SIMD mixed architecture.

We are currently working with Saab to demonstrate a configurable dynamic link based on programmable BBP, RF front-end and processing experience from electronic warfare systems [7]. Our goal is to establish a programmable BBP platform for different levels of applications including low power JTRS terminals and broadband link for flight vehicles.

References

[1]. Dake Liu and Eric Tell, Low-Power Baseband processors for Communications, chapter 23 in Low power Electronics Design, edited by C. Piguet, CRC 2004, ISBN 0849319412.

[2]. Anders Nilsson, Eric Tell, and Dake Liu, An accelerator architecture for programmable multi-standard baseband processors, in *Proc. of WNET2004, July, 2004, Canada.*

[3]. Eric Tell and Dake Liu, A Hardware Architecture for a Multi-Standard Block Interleaver, *Moscow, Russia July, 2004*

[4]. Eric Tell and Dake Liu, A Converged Hardware Solution for FFT, DCT and Walsh Transform, in *Proc. of ISSPA2003, July, 2003, Paris France*

[5]. Heiskala, H. and Terry, J. T., *OFDM Wireless LANs: A Theoretical and practical guide*, Sams Publishing, 2002, ISBN 0672321572.

[6]. IEEE Std. 802.11a-1999 and IEEE Std. 802.11b-1999

[7]. Ingemar Söderquist, *CMOS Circuits for Digital RF Systems,* Linköping studies in science and technology. Dissertation, No 775, 2002. ISBN 91-7373-427-2, October 2002