A Low-Power VLIW processor for 3GPP-LTE Complex Numbers Processing

Christian Bernard and Fabien Clermidy
CEA-LETI / MINATEC campus
Grenoble, FRANCE
{first_name}.name@cea.fr

***Abstract*—new generation of telecommunication applications requires highly efficient processing units to tackle with the increasing signal processing algorithmic complexity. They also need to be flexible for handling a large range of radio access technology with specifications moving very fast. As devices including telecommunication features are, per nature, mobile, the high level of flexibility must be achieved while preserving very low power consumption. In this paper, a high performance low-power application-specific processor is proposed for complex signal processing. Thanks to dedicated control architecture, this processor exhibits an average 81% utilization rate of its principal operator, a complex MAC for a 3GPP-LTE application. The main innovations are the use of a reconfigurable profile and instruction cache strategy to reduce power consumption. This leads to a 10x reduction of the control power consumption. As a result, an average 50 mW power consumption is measured after implementation in a low-power 65 nm technology while delivering 3.2 GOPS. Finally, a comparison with state-of-the-art low-power DSP shows at least 24 % gain.**

Digital Baseband, Signal processor, VLIW, Low-Power, 3GPP-LTE

I. INTRODUCTION

Next generation of telecommunication applications is including high throughput requirements (more than 300 Mbits/s) associated with high flexibility due to new usages, such as cognitive radio for better spectrum efficiency. As a consequence, the digital workload of such future smartphones requires very high performance (up to 100GOPS) with limited power budget (less than 1 W) and high flexibility [1]. Particularly, only a few hundreds of mW can be devoted to the radio baseband which needs to support more than 10 protocols, including high-end 3GPP-LTE. The performance versus power consumption trade-off is impossible to achieve with classical Digital Signal Processors (DSP). On the contrary, purely dedicated hardware cannot tackle the flexibility requirement. A medium way is to develop Application Specific Instruction set Processor (ASIP) which are processors using dedicated operators and control for solving a dedicated class of algorithms.

In this paper, we propose an ASIP-based architecture called MEPHISTO designed for the processing of algorithms solving complex numbers equations. This unit is based on a Very Long Instruction Word (VLIW) architecture built around a complex number MAC operator. To reduce the power consumption in the control part, a combination of two main techniques is applied: a small instruction cache associated with a profile table. The rest of the paper is organized as follows. Section II gives the rational for the approach while section III gives an overview of the architecture as well as the description of the data-path. Section IV focuses on the control and its innovations while section V presents the results and comparison with state-of-the-art low-power DSPs.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. RATIONAL OF THE APPROACH

A. Application requirements

Typical functions of a 3GPP-LTE Multiple Inputs, Multiple Outputs (MIMO) scheme [13] are presented for a 4x2 MIMO scheme in Fig. 1. Each function consists of a set of small-sized algorithms working on the flow of data. The computing / communication ratio is typically very high (5 to 20) and hard real-time constraints must be taken into account (1 ms for decoding a frame in this example).

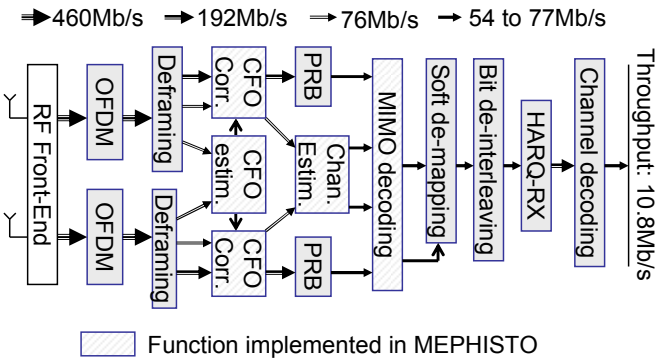


Figure 1. 3GPP-LTE 4x2 MIMO receiver chain

An example of algorithm for computing the MIMO decoding function is given in (1).

$$M = H^H * (H^H * H + \sigma^2 * I_1)^{-1}, \quad (1)$$

Where H is a 4x4 Matrix obtained with pilots information at two different sampling times t and t+1 from the 8 possible paths on the 4 emitting antennas to the 2 receiving antennas, σ is the estimated noise level of the transmission channel, I_1 is the identity matrix. All the values are complex numbers (I,Q)

with each scalar coded on 16 bits. As shown in this example, complex operations such as matrix multiplication, inversion and transposition are involved in such algorithm. Finally, considering the number of 10 functions, a power budget of 50 mW per function is a reasonable target for a 500mW total budget of the digital baseband [1].

B. State-of-the-art

Single Instruction Multiple Data (SIMD) or Single Instruction Multiple Threads (SIMT) architectures have been proposed for baseband processing [2-6]. For these proposals, the objective is to reduce the number of instructions per operation, limiting the control overhead in terms of area and power. The SODA architecture [3] is a pure SIMD with a phase of data alignment for feeding the vector computing unit. The limitation of such architecture is the important time required for doing load/store operations only. SODA II [4] is an evolution intended to hide the load/store operations thanks to efficient pipelining. It results in 120 mW power consumption for low data-rate single antenna WCDMA (2 Mbps). This power consumption is small, but the performance required for WCDMA is less than 10% of the one needed for 4*2 MIMO 3GPP-LTE delivering 10Mbps. The SIMT architecture [2] is an evolution of the SIMD concept with a multi-threaded control for different vector/non vector processing unit. It makes the scheme more flexible for scalar operations, but still lay on the fact that most of the operations are vector-based for good usage efficiency. The DVB application is demonstrated, which does not include any MIMO scheme, but large FFT of 8K.

As a conclusion, applications like WCDMA, HSDPA or DVB T/H present a high-level of vector operations [4]. The limitation of such architectures is irregular algorithms, where the data cannot be processed in parallel with the same instruction. This is typically the case with the MIMO decoding algorithm performing a complex matrix inversion. When mapped on SIMT or SIMD architecture, these algorithms are not using efficiently the operators or need a large memory for performing parallelism at a coarser grain.

VLIW architectures have been proved to be more flexible [7], and can achieve high performance on irregular algorithms. However, as pointed out in [2], their drawback resides in the large instruction to be fetched each cycle, leading to high power consumption.

C. MEPHISTO approach

In this paper, we propose to use a VLIW approach which is the only one giving the right level of performance for the envisioned algorithms. The objective is to have a compact unit able to tackle with a full computation-intensive algorithm. Then, this unit will be duplicated for the different algorithms required. In such a way, the scheme is modular and scalable, and can be adapted to different application requirements. Moreover, we can dynamically adapt the number of processing elements running depending on the level of performance of the application, and thus adapt the power consumption more easily than in SIMD or SIMT approaches. Multiple instances of this compact VLIW processor are then connected by a Network-on-Chip to provide the required performance, like in [10].

As the objective is a compact power-efficient ASIP, the firmware RAM and data RAM are limited to 8Kbytes each, and external RAM is used to store the flow of data [11]. To overcome the problem pointed out of large consumption in the control path, a reconfigurable profile and cache strategy is used to i) avoid frequent access to the FW memory ii) extend the 64 bits-large FirmWare Ram (FW) to the 270 bit-width instructions required to feed all the operators. In addition, an efficient implementation of a MAC operator working on complex numbers is performed for achieving high performance in 65 nm technology.

III. ARCHITECTURE DESCRIPTION

A. Overview

MEPHISTO is a 32-bit data path VLIW structure organized around a MAC dedicated to complex arithmetic but able to perform scalar operations as well, and two dedicated operators, a cordic/divider block and a compare/select one (Fig. 2). Data can be fed directly from two RAM banks and a register array. The control part extracts from the 1Kx64bits instruction memory (FW RAM) the compacted instructions needed to feed the data path and the five address generators.

The whole architecture presents a deep pipeline of 10 stages in order to achieve 400 MHz worst-case frequency in 65nm. Deep pipeline is not a drawback for telecom algorithms which require a minimal number of branch instructions.

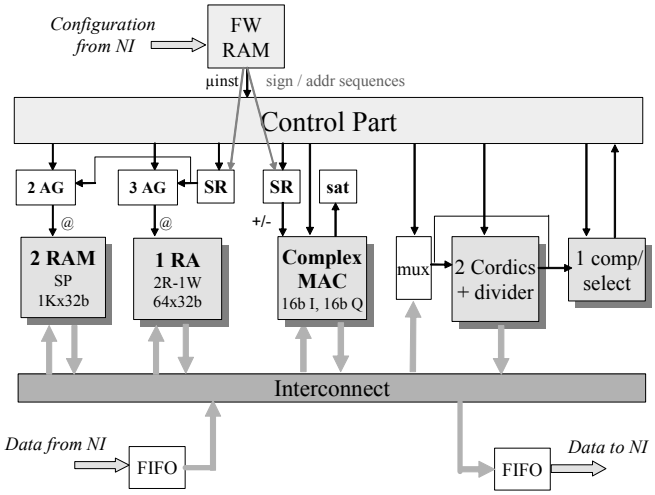


Figure 2. MEPHISTO architecture

B. Data Path

The MAC can perform complex and scalar operations with 2 independent sub-parts (16b I, 16b Q). It is a 4-stage pipeline composed of i) four 16bx16b multipliers (32b result) using carry save notation, ii) 2 partial adders and 2 pairs of 40b accumulators, iii) a conversion stage in 2-complement notation and iv) finally a programmable shift and saturation. For the whole operation, the MAC offers 3,2 scalar GOPS.

To efficiently feed this MAC, 5 programmable data transfers and sequencing (including hardware loop) can occur within a cycle. Two 1K 32-bits single port RAM and a 64-32 bits words Register Array (RA) with 2 read and 1 write ports

are accessible in parallel. Each RAM and RA port has its own Address Generator (AG) to schedule the operands. Each AG is controlled independently and hosts an address pointer array: a pointer or a constant from a current µinstruction can be added to another pointer with modulo features.

Sign and address sequences may be used for irregular operations such as determinant calculation for matrix inversion or pilot correlation. There are 2 shift registers (SR), one for bit sequences and another for address sequences. A SR is initialized by a µinstruction with a sequence address and word count. The SR reads the sequence from Firmware (FW) RAM through an intermediate 64-bit buffer to load a shift register that is shifted according to µinstruction requirement. Up to 3 addresses may be issued at a time to AGs.

Finally, Compare/Select, 12b cordics and 32b inverter (1/x) operators give flexibility for specialized operations. The MEPHISTO inputs/outputs consist in FIFOs to receive/send data from/to the Network-on-Chip (NoC) through a Network Interface (NI).

IV. VLIW CONTROL

To feed all the operators in parallel, a 270bit large instruction must be loaded at each cycle. A straightforward implementation of a VLIW control part leads to 70 mW consumption (see section VI-B). MEPHISTO reduces this consumption by a factor of 10 using registers arrays and adding a reconfigurable profile and cache instruction strategy (Fig. 3). This organization is based on the observation that targeted algorithms commonly use a restricted set of operations: the cache contains a subset of instructions while the profiles embed the most commonly used instructions for a set of operators.

When studying the various algorithms of the 3GPP-LTE application [13], it appears that the kernel of the algorithm requires only a small number of instructions with nested loops. Thus, for area reduction reasons the register cache is limited to 4 pages of 8 µinstructions. A 4-entry directory translates the RAM page address into cache page index. Due to its reduced size, the directory consumption cost is acceptable and highly compensated by avoiding power-hungry RAM read. A µinstruction is 64-bits and is composed of one 32-bit sequencing field and 8 times 4-bits index fields. Each index points to a 16-words profile table. Each command profile (from 17 to 60 bits-large) directly controls a set of operators.

The profile tables are reconfigured from the 1Kx64bit FW RAM before any µprogram execution, as well as the register cache and datapath RAMs and registers. In some way, the instruction set can be considered as reconfigurable. The use of profile tables greatly reduces the amount of bits to read in the FW RAM when loading a new page in the cache during the µprogram execution, leading to an efficient consumption reduction. For pipelined operators, the whole command profile is delivered when starting an operation, and the command bits are then shifted along with the operands. The main sequencing µfunction is loop start. Up to 4 loop levels can be nested. Only one loop start can be programmed in a cycle but the 4 levels may be unstacked simultaneously.

The MEPHISTO RAMs and tables are configured from the network by configuration packets before any program execution, or by the µloader when starting a program. The µloader then executes a configuration sequence of load instructions read from FW RAM. The Data to load are also in the FW RAM. A load instruction contains a source address, a target address (cache, profile tables, register and pointer arrays, data RAMs), a word count and a bit indicating the last instruction of the sequence. The µloader is also in charge of the FW RAM access arbitration.

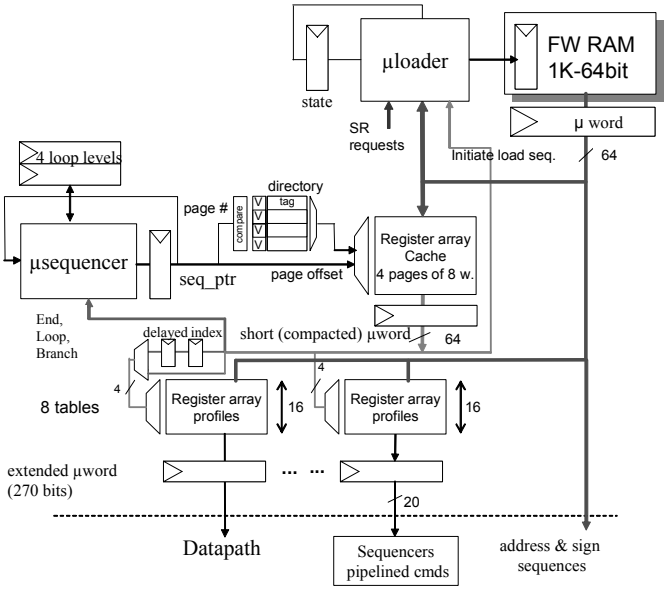


Figure 3. MEPHISTO control

To avoid cache miss latency, a µfunction is used to pre-fetch a page into the cache. The directory entry corresponding to the replaced page is then invalidated and a page load request is sent to the µloader. When the page is filled into the cache, the directory entry is validated again. However the HW is able to fetch itself the required page if the directory misses. A µprogram may need more than 16 command profiles per operator. In such a case, the profile table can be dynamically reconfigured depending on the page to be loaded in the cache: the pre-fetch µfunction sends to the µloader the address of a command sequence that loads the profiles before loading the page itself. Therefore the profiles being replaced must no more be used by the µprogram before entering in the new page. As there is no associative access to the profiles, the compiler must also manage the translation from virtual profile index to real index at each step of µprogram.

Some further improvement consists in delaying the profile indexes thanks to specific pipeline mode µinstructions inducing loop compaction. For example, when executing matrix operations, it is convenient to describe in one µinstruction, operand reading, MAC operation and result writing. Therefore the index of MAC command table may be delayed from 2 cycles and the write port from 7 cycles, according to pipeline mode registers. The consumption cost of

delaying small indexes is reduced compared to delay the whole profiles themselves.

In order to sequentially run multiple algorithms, up to 8 µprograms may co-exist in the FW RAM thanks to a table of 8 configuration descriptors. A descriptor contains a FW base, a load sequence base and starting pointers of µprogram (A) and initial configuration (B), as shown in Fig. 4. Any pointer used by MEPHISTO control is relative to the bases, so the code is easily translatable in the FW RAM. Even more, a same code may be used by several configurations with different program start addresses (A) and initial load sequence addresses (B).

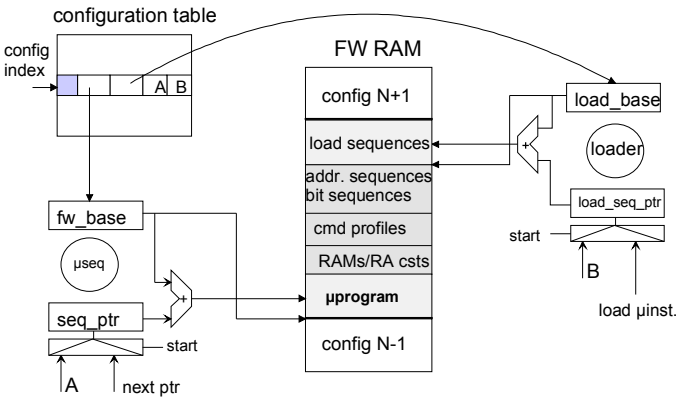


Figure 4. MEPHISTO configuration

V. PROGRAMMING

A programming environment is necessary for porting new algorithms on the processor. MEPHISTO is not a general-purpose processor, and algorithms have to be optimized in terms of performance and power consumption. Thus, the code must be pushed near to the best performance of the architecture and having a full compiler is not relevant. Instead, it is useful to have a programming environment for testing different optimizations. This is the purpose of the MEPHISTO-C code presented in Fig. 5. Based on classical C code, it offers the possibility of trying some optimization by using the register arrays and processing data with the available dedicated operators. As MEPHISTO has a classical processor structure, basic C-code can be used and compiled, and then refined by introducing i) the register array; ii) the dedicated operators; iii) the profiles. For each refinement, the code is compiled using a classical GCC compiler, and can be compared to reference results from a pure fixed-point C-code. The compilation of profiles and build of the instruction words taking into account profiles and caches size is then automated in an assembly phase to obtain binary code.

```
// ------- Computing P = H * N-1
// --------------------
ptr(ptrp0,RL1,0); ptrp0=0; // -------- pointer to registers mapping ; pointer initialization
ptr(ptrp1,RL2,0); ptrp1=0;
ptr(ptwp,WR,0); ptwp=0;

for(i=0;i<4;i++) {
  for(j=0;j<4;j++) {
    Acc0.I=0; Acc0.Q=0; // Accumulator initialization
    for(k=0;k<4;k++) {
      Acc0.I+=N1[ptrp1].I*RA_rd(RA_buf,ptrp0).I+N1[ptrp1].Q*RA_rd(RA_buf,ptrp0).Q;
      Acc0.Q+=N1[ptrp1].Q*RA_rd(RA_buf,ptrp0).I-N1[ptrp1].I*RA_rd(RA_buf,ptrp0).Q;
      ptrp0+=1; // ---------- H matrix line number
      ptrp1+=1; // ---------- N-1 matrix column number
    }  // end for k
    P[ptwp]=dec(Acc0,10);
    ptrp1-=4;
    ptwp+=1; // ---------- Next P column index
  } // end for j
  ptrp1+=4;
  ptrp0=0;
} end for I
dump(P,"P",16); // dump results
```

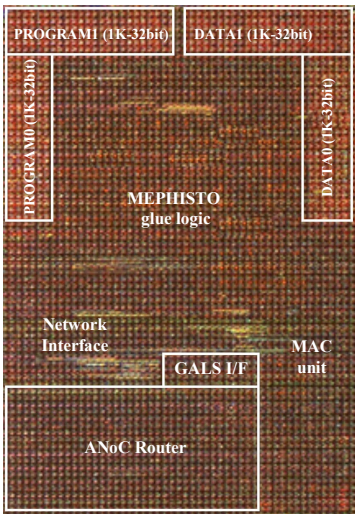
Figure 5. An example of MEPHISTO C code for describing application

VI. IMPLEMENTATION AND RESULTS

A. *MEPHISTO layout*

The MEPHISTO processor is implemented in a Low-Power 65 nm technology using three transistors V_T for better performance / energy trade-off. Five instances of this block have been put into a complete NoC-based SoC [10]. This chip includes dedicated IPs for FFT/IFFT, bit-level processing units including channel decoding, distributed memory management cores and an ARM11 processor for MAC control. In such a scheme, MEPHISTO is used to perform signal processing, except the FFT/IFFT.

The MEPHISTO area (Network Interface included) is 0,631 mm² (Fig. 6) for a 400 Mhz worst-case frequency (110 °C, 1,1V, process worst). The leakage is minimized to 320 µW thanks to a large usage of Low-Power transistors (high and standard threshold voltage transistors). Typical clock gating is used to further optimize the dynamic power consumption. Around 97 % of the registers are clock-gated.



Process	65 nm
Area	0,631 mm²
Gate count	243 K
Program memory	64 Kbit
Data memory	64 Kbit
Freq (worst case)	400 MHz
Freq (typ. case)	710 MHz
Average power @ 400MHz	50 mW
Leakage power	320 µW

Figure 6. MEPHISTO layout

B. *Cache efficiency*

Based on the general presented architecture, we made a study on the benefit of the chosen control strategy. Table I summarizes the results. They clearly show the benefit of the 2 cache levels to reduce the consumption and avoid too large area overhead. By introducing the first level, we reduce the number of RAM accesses. Anyway, we don't reduce the cost for accessing a full RAM instruction word. Moreover, the memory area greatly increases due to the added caches.

When introducing the profiles, we reduce both the cost of reading a µinstruction in the RAM and the size of the RAM itself. Another consequence is a reduction of the cache width, as instructions are smaller. Consequently, the area is reduced by 25% and power consumption by 80% compared to the cache solution alone. The Control Part area is increased by 80% compared to the simple implementation, while the control power consumption is divided by 10. This result point out the necessary trade-off between power and area saving. It is worth noticing that this overhead is only on the control part, and reduced for the whole design.

TABLE I. POWER EFFICIENCY OF DIFFERENT CONTROL STRATEGIES

	without Cache	with Cache	with Cache + profiles
Cache miss rate	100%	16%	16%
Surface (RAM 1K µinst)	0,10 mm2	0,24 mm2	0,18 mm2
number of cache bit registers	0	8692	5522
Power comsumption	**72 mW**	**29 mW**	**6 mW**

C. *Algorithms Mapping Efficiency*

In order to prove its efficiency, four different functions have been programmed on MEPHISTO. These algorithms have been extracted from a typical 3GPP-LTE application shown in Fig. 1. The real-time constraint on this chain is 1 ms, and this constraint is used to decide the functions distribution on the different MEPHISTO cores.

Table II shows the execution times for the different algorithms. At 400 MHz, 115% of a single MEPHISTO peak performance is required to perform all the operations, regardless of the configuration and NoC transfer times. In this example, 4 MEPHISTO cores are used. The MAC occupation ratio is kept high, between 70% for irregular algorithms (e.g. MIMO decoding) until 96 % for regular algorithms (e.g. channel estimation), leaving some margin for more complex algorithms. For Carrier Frequency Offset (CFO) tracking, MAC receives its 2 operands from the input FIFO and thus cannot be used at more than 50%. For channel estimation, large coefficient tables are memorized into the RAM, with poor locality leading to a high RAM access rate (87%). Channel estimation has a high cache hit ratio and loosely uses FW RAM (1.5%). MIMO decoding makes frequent accesses to the FW RAM due to reading of address sequences for determinant calculation. Interpolation has a high cache miss ratio due to µprogram structure: main loop is bigger than cache and rapidly executed, then cache pages are continuously swapped.

TABLE II. POWER CONSUMPTION DISTRIBUTION FOR TYPICAL ALGORITHMS

operation	MIMO decoding	Channel estimation	Interpol	CFO tracking	Average
FW RAM accesses	44,5%	1,5%	52,7%	1,4%	27,6%
write FIFO out	2,0%	10,5%	11,4%	0,8%	6,1%
read FIFO in	5,0%	1,5%	28,6%	95,5%	6,1%
RAM0 accesses	6,3%	87,0%	0,00%	0,0%	38,0%
RAM1 accesses	18,1%	9,3%	0,00%	0,0%	13,1%
MAC	71,7%	96,4%	68,6%	48,8%	81,2%
read RA0	65,4%	94,6%	68,6%	49,4%	77,2%
read RA1	48,3%	0,0%	11,4%	48,5%	26,3%
write RA	2,9%	1,7%	40,0%	96,3%	5,9%
µinstruction count	87	63	55	41	
Duration@400MHz	596,9µs	457,40µs	84µs	8,4µs	
Power consumption	52 mW	48 mW	52 mW	42 mW	50 mW

D. *Power Distribution*

The power consumption at full speed is between 42 mW for algorithms which don't use RAM (CFO tracking) and 52 mW for functions with more accesses to the RAM (MIMO decoding and interpolation). The total power for executing demodulation part of a TTI reception (1ms duration) is 58 mW and is in line with a 4G mobile power budget [1]. The power budget repartition for a typical algorithm is given Fig. 8 and can be compared to a typical processor power consumption distribution shown in Fig. 7. It shows the smaller part taken by the control part which is 50% for MEPHISTO instead of 66% in RISC processor. It leads to a better usage of the arithmetic part which represents 23% instead of only 6% for a general purpose processor [7]. These results prove the efficiency of the ASIP approach as well as the control strategy.

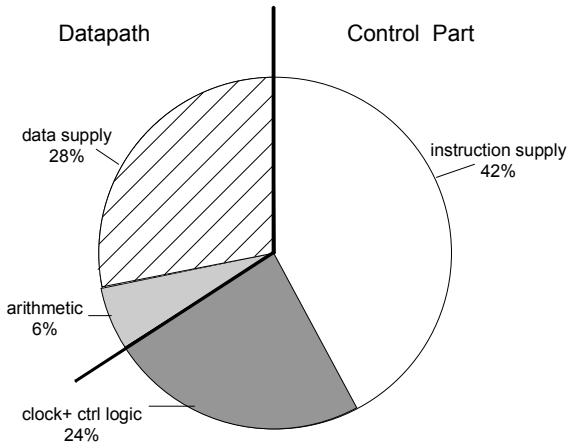


Figure 7. Risc processor power consumption typical distribution from [7]

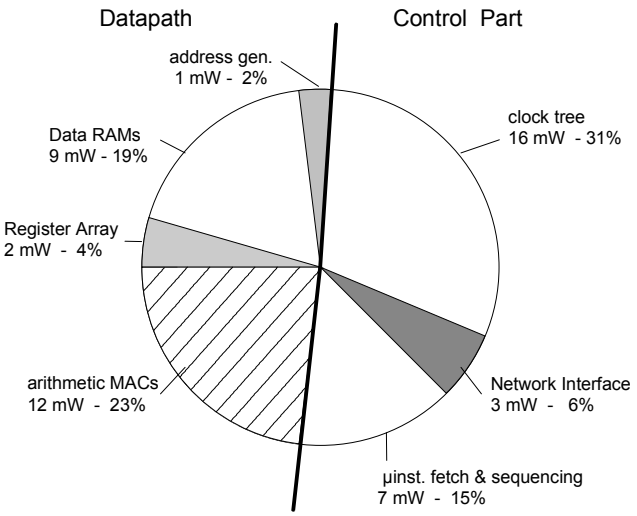


Figure 8. MEPHISTO power consumption distribution

For the complete application, 4 MEPHISTO are used, leading to 3.2 mm2 in 65nm. The architecture proposed in [2] has similar area (3.3mm2), reported to the same technology node. But computation complexity is very different: the 4*2 MIMO scheme is more than 2 times more complex than 2*2 MIMO developed in [2], and the computed bandwidth is 20 MHz for MEPHISTO, when only 10 MHz for [2]. As a result, the performance available in MEPHISTO is approximately 4 times the one in [2]. Finally, the power consumption reported to one frame is 58 mW for the 4 MEPHISTO (corresponding to the 115% MEPHISTO peak performance required in VI-C) instead of 70 mW in [2]. As the technology node used in [2] is 120nm, we assume the non-ideal scaling scheme of [12], and thus the power scale only with Vdd^2. As the voltage of [2] and this design are the same, we conclude that MEPHISTO is approximately 4 times more efficient than [2].

E. Comparison with some DSP

Regarding power efficiency, MEPHISTO is compared to the dedicated OFDM engine proposed in [10] and other specialized low-power DSP on a 256-point FFT [8][9] using normalized power at 1V (Table III). Even if MEPHISTO is not optimized for FFT (no radix operator), it is able to achieve similar performance than Macgic [8] with a better power budget (24% power reduction) and clearly exhibits better performances than Icyflex [8] and Coolflux [9].

TABLE III. COMPARISON ON A 256 POINTS FFT

	TRX_OFDM [10]	MEPHISTO (*this work*)	Macgic [8]	Icyflex [8]	Coolflux [9]
Supply voltage (V)	1,2V	1,2V	1,1V	1,1V	0,9V
Frequency (MHz)	400	400	150	150	15
256-bit FFT cycle count	592	4096	1410	2580	5500
abs. performance (µs)	1,48	10,24	9,40	17,20	379,31
Power per FFT (mW)	105,6	48,0	54,4	36,3	1,50
Power per FFT @ 1V (µW)	73,3	33,3	45,0	30,0	1,8
Normalized energy per FFT @ 1 V	0,32	1,00	1,24	1,51	2,06

VII. CONCLUSION

In this paper, we have presented a new VLIW ASIP optimized for complex matrix computing, largely used in MIMO chains and especially in 3GPP-LTE protocols. This processor shows a very good average utilisation ratio of 81 % for typical telecommunication algorithms. The innovations provided in the control scheme highlight a gain of 10 times for the control part compared to direct implementation of a VLIW processor. Finally, the performance versus power ratio improves state-of-the-art results by at least 24%, showing the efficiency of the new proposed control scheme.

REFERENCES

[1] C. H. van Berkel, "Multi-Core for Mobile Phones", Proc. of Design Automation and Test in Europe, DATE 2009, Nice, France, 2009.

[2] A. Nilsson et al., "An 11 mm2, 70 mW Fully Programmable Baseband Processor for Mobile WiMAX and DVB-T/H in 0,12µm CMOS", IEEE Journal of Solid-State Circuits, Vol. 44, No1, Jan. 2009, pp 90-97.

[3] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "SODA: A low-power architecture for software radio," in *Proc. Int. Symp. Comput. Architecture*, Jun. 2006, pp. 89–101.

[4] H. Lee, C. Chakrabarti and T. Mudge, "A Low-Power DSP for Wireless Communications", IEEE trans. On Very Large Scale Integration (VLSI) Systems, May 2009

[5] W. Raab, J. Berthold, U. Hachmann, D. Langen, M. Schreiner, H. Eisenreich, J. Schluessler, G. Ellguth, "Low Power Design of the X-GOLD® SDR 20 Baseband Processor", industrial presentation at Design, Automation and Test in Europe, DATE'10, March 2010

[6] K. van Berkel et al., "Vector processing as an Enabler for Software-Defined Radio in Handheld Devices," EURASIP Journal on Applied Signal Processing, vol. 16, pp 2613-2632, 2005.

[7] W.Dally et al., "Efficient Embedded Computing", IEEE Computer, July 2008, pp 27-32.

[8] C. Arm et al., "Low-power 32-bit dual-MAC 120µW/MHz 1.0V icyflex DSP/MCU core", ESSCIRC 2008, Edinburgh, UK, Sept. 2008, pp 190-193.

[9] H. Roeven et al., "CoolFlux DSP, the embedded ultra low power C-programmable DSP core", Intl. Signal Processing Conf. (GSPx), Santa Clara, USA, 2004

[10] F. Clermidy et al., "A 477mW NoC-Based Digital Baseband for MIMO 4G SDR", IEEE International Symposium on Solid-State Circuits, ISSCC'10, Feb. 2010.

[11] J. Martin et al., "A Microprogrammable Memory Controller for High-Performance Dataflow Applications", European Solid-State Circuits Conference (ESSCIRC), Athens, Greece, September 14-18 2009

[12] J. Rabaey, A. Chandrakasan, and B. Nikolic, "Digital Integrated Circuits: A Design Perspective", Prentice Hall, Upper Saddle River, New Jersey, 2003.

[13] D. T. Phan Huy, R. Legouable, D. Kténas, L. Brunel, M. Assaad, "Downlink B3G MIMO OFDMA Link and System Level Performance", IEEE VTC spring 2008, Singapore, May 2008