# Very Wide Register: An Asymmetric Register File Organization for Low Power Embedded Processors

Praveen Raghavan *†, Andy Lambrechts *†, Murali Jayapala *,
Francky Catthoor *†, Diederik Verkest *†‡, Henk Corporaal *∪
* IMEC vzw, Kapeldreef 75, 3001 Leuven, Belgium
† KULeuven, Belgium; ‡ VUB, Belgium; ∪ TU Eindhoven, Netherlands
{ragha, lambreca, jayapala}@imec.be

## Abstract

*In current embedded systems processors, multi-ported register files are one of the most power hungry parts of the processor, even when they are clustered. This paper presents a novel register file architecture, which has single ported cells and asymmetric interfaces to the memory and to the datapath. Several realistic kernels from the TI DSP benchmark and from Software Defined Radio (SDR) are mapped on the architecture. A complete physical design of the architecture is done in TSMC 90nm technology. The novel architecture presented is shown to obtain energy gains of upto 10X with respect to conventional multi-ported register file over the different benchmarks.*

## 1 Introduction

Future mobile terminals need to support new multimedia and wireless communication standards with a high computational complexity and have an extreme energy efficiency to provide a long battery life. Current state of the art processors are facing several bottlenecks that prevent this required combination of performance and energy efficiency. Multi-ported data Register Files (RF) are one of the most power hungry parts of any processor, especially VLIWs [13]. On average every operation requires three accesses (two reads and one write) to the RF, which make them a very active part of the processor. Current architectures try to achieve a high performance by exploiting parallelism, and therefore perform multiple operations per cycle (eg. Instruction Level Parallelism or ILP, as used in VLIW processors). This quickly results in a large port requirement for the register file organization (also interchangeably refered to as foreground memory organization), that is mostly implemented as a (set of) large multi-ported register files. A high number of ports has a negative impact on the energy efficiency of register files. Traditionally, this problem is addressed through various clustering techniques [16] that partition (or bank) the RF. Data can then only be passed from one partition to another through inter-cluster communication [17, 14]. However, as partitions get smaller the cost of inter-cluster copies quickly grows and the resulting register files are still multi-ported. For high energy efficiency, it is preferable that the registers be single ported [9].

Another important energy efficiency bottleneck is formed by the Level-1 data memories (scratch pads or caches) [13]. Reducing this bottleneck can be achieved by improving one or more of three aspects: the memory design (circuit level), the mapping of data onto the memory, the memory organization (interface). In this paper we will use a standard state-of-the-art memory, with a slightly modified organization compatible with most memory generators: a wide memory organization, where at the interface between the memory and the foreground memory, the complete width of the memory will be read out (complete line)[1]. By making wide memories, related blocks of data can be loaded in parallel, thereby reducing the decoder overhead. This requires the bus between the memories and the register file to be wide as well.

In this paper, we present a novel asymmetric register file organization, together with its interface to the wide memory, that achieves a significantly higher energy efficiency than conventional organizations. The proposed register file or foreground memory organization is shown in Figure 1. Three aspects are important in the proposed organization: the interface to the memory, single ported cells and the interface to the datapath. The interface of this foreground memory organization is asymmetric: *wide* towards the memory and *narrower* towards the datapath. This foreground memory is similar to a register file which is incorporated in the datapath pipeline cycles. The wide interface enables to exploit the locality of access of applications through wide loads from the memory to the foreground memories (registers). At the same time the datapath is able to access words of a smaller width for the actual computations (further details in Section 3).

[1]The details of the memory design and mapping are beyond the scope of this paper. However, we qualitatively motivate in Section 3 that for energy efficiency wider memories are suitable.

978-3-9810801-2-4/DATE07 © 2007 EDAA

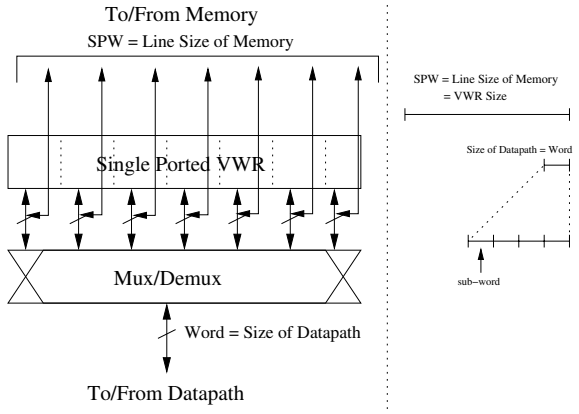


**Figure 1. Very Wide Register Organization**

A set of Very Wide Registers (VWR), with a single port each is used to replace a traditional register file. Micro-architecturally, every single VWR is made of single ported cells and it has no pre-decode circuit. A post-decode circuit consisting of a multiplexer (MUX) is provided to select the appropriate word(s). For evaluating the gains achieved by the proposed organization, we present a detailed and optimized physical design of: register files, interfaces to the memory and the datapath.

The rest of the paper is organized as follows: Section 2 gives an overview of related work in the area of register file architectures. Section 3 gives a detailed description of the proposed very wide register architecture and its connectivity of the VWR to the memory and the datapath. An example mapping of data to the VWR is illustrated in Section 4. The experimental setup and the results are presented in Section 5. Finally, Section 6 concludes this paper.

## 2 Related Work

A large amount of research exists on improving the performance and energy efficiency of register file organizations. Many techniques have been proposed at various levels of abstraction (namely circuit, architecture, compiler and system level). In this section we present an overview of the state-of-the-art architectural are presented and emphasize the differences with the proposed organization.

Clustering register files is a generic architectural technique that reduces energy consumption by splitting register files in smaller parts [17, 14]. Since the energy/power consumption increases super-linearly with number of ports, clustering techniques reduce the number ports per cluster and therefore improve energy efficiency. This comes at the extra cost of inter-cluster communication [16, 14]. Fully distributed register file organizations are an extreme form of clustering, where single ported registers are used at the outputs (or inputs) of the functional units [8, 5, 9]. A bypass network interfaces the functional units in the datapath and the memory. In our approach, multiple VWRs can be viewed as multiple clusters, with three VWRs per cluster. Each VWR has only a single port.

In state-of-the-art clustered organizations all the ports are symmetric i.e., the interfaces to datapath and to the memory are of equal width and often shared. In our approach the ports are asymmetric; there is a wide port to the memory and a narrow port towards the datapath. This allows the datapath to access the VWR through a cheaper, often used interface, while accesses from the memory go through a more expensive but less used interface. Data transfers between the memory and the VWR copy a complete line of the memory to the VWR, or a complete VWR to the memory (contiguous part of line possible for more flexibility). A read/write between the VWR and the datapath operates on a single word in the VWR (small part of a line). Hence this separate and asymmetric interface makes a more optimized usage of the energy and bandwidth possible.

The concept of wide registers is commonly used in vector registers for data-parallel architectures [11, 4, 12]. Multiple data elements are stored into the vector registers and all the data are read out to the datapath. This set of data is refered to as a word and each data element inside this word is refered to as sub-word. However, in that case the width of the register file and the datapath are still equal. The main motivation for wide registers in these architectures is to support data-parallelism. The same operation is performed on multiple data (Single Instruction Multiple Data, SIMD) that are stored in the vector registers. Our approach is complementary to the vector register approach. Firstly, our primary target is energy efficiency. Secondly, our approach can be used in both data-parallel and non-data-parallel contexts. In a data-parallel context, the widths of data read to the (SIMD) datapath from the VWR are the same as the vector data size. However, a single VWR is much wider than a vector register and hold multiple vector data. Figure 1 also illustrates the relative sizes of lines, words and subwords. SIMD datapaths can be used complementary to the proposed VWR based architecture. In such a case, each line contains a set of vector words and each vector word contains the different subwords. Usually a large amount of SIMD parallelism is not available in applications due to dependencies and therefore wider datapaths cannot be used. In case of such dependencies, the VWR can still be used, where consecutive words are dependent on each other.

## 3 Architecture Description

As described in Section 1, the motivation for the proposed architecture is derived from various parts of the processor. Section 3.1 gives the architectural innovations in the data memory hierarchy. Section 3.2 presents the Very Wide Register and its microarchitecture. Section 3.2 shows the interconnection between the scratchpad memory and the VWR and the interface between the VWR and the datapath

is described in Section 3.3.

### 3.1 Data Memory Organization and Interface

As mentioned in Section 1, energy in memories can be reduced by improving one or more of three aspects: the memory design (circuit level), the mapping of data onto the memory, the memory organization (interface). In this section, we discuss the memory organization. A detailed energy breakdown of an SRAM based scratchpad[2] shows that for a typical size for the level-1 data memory (eg. 64Kb) about half of the energy is spent in the decoder and the wordline activation [3, 6]. The other half is spent in the actual storage cells and in the sense-amplifiers. The decode cost is the price that is paid for being able to access words in any given order. The energy consumption in the memory organization can be optimized further by performing as few decodings as possible by reading out more data for every decode. In the embedded systems domain this can be achieved by exploiting the available spatial locality of data.

The row address (*Row Addr* in Figure 2) selects the desired row in the memory through the pre-decoder. The sense-amplifiers and precharge lines are only activated for the words that are needed and only these will consume energy and are read out. Figure 2 shows the proposed scratchpad organization and the address that has to be provided. To be able to handle partial rows (less optimal for energy, but more flexible), the full address contains two additional fields: *Position* decides at which word the read-out will start, while *No. Words* decides the number of words that has to be read out. Hence, at most a complete row and at least one word of the SRAM can be read out and will be transfered from the scratchpad to the VWR registers. The scratchpad can be internally partitioned or banked and the proposed technique can be applied on top of the banked structure.

This architecture is compatible with almost all existing SRAM generators (e.g. Artisan), but in actual instantiation such a wide interface is not yet used, so the energy and performance models may have to be extended for such a wide output. If the used design library does not contain such a wide memory, it can be composed from multiple narrower memories by connecting them in parallel, but the overhead due to extra decoding would limit the gains.

### 3.2 Foreground Memory Organization

We propose a single ported register cell as shown in Figure 1. This register organization is called *Very Wide Register* (VWR). The VWR has asymmetric interfaces: a wide interface towards the memory and a narrow interface to the datapath. Every VWR is as wide as the line size of the scratch pad or background memory, and *complete* or *partial lines*

[2]Scratchpad based memories are used instead of Cache based L1 memories as they have been shown to be energy efficient [10].

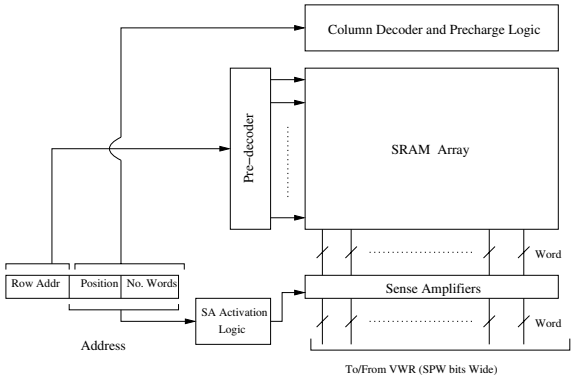


**Figure 2. VWR & Scratchpad Organization**

can be read from the scratchpad into these VWRs. The VWRs have only a post-decode circuit, which consists of a Multiplexer/De-Multiplexer (MUX/DEMUX). This circuit selects the words that will be read from or written to the VWR. Each VWR has its own MUX and DEMUX, as shown in Figure 1. The controls of the MUX and DEMUX on which register is to be accessed is derived from the instructions. Because of the single-ported cell design, the read and write access of the registers to the scratchpad memory and access to the datapath cannot happen in parallel. The VWR is a part of datapath pipeline with a single cycle access similar to register files.

Since the interface of the VWR to the memory is as wide as a complete VWR, which is of the same width as the memory and the bus, the load/store unit is also different. It is capable of loading or storing complete (or partial [3]) lines from the scratchpad to the VWRs. Section 4, shows an example on how the load/store operations are performed between the memory and the VWR.

During *placement and routing* (e.g. using Magma Fusion Blast place and route tool) the cells of the VWR are aligned with pitch of the sense amplifiers of the memory to reduce the amount of interconnect and the related energy. This enables clear direct routing between the wide memory and the VWR without much interconnect overhead. The same optimization cannot be done in case of a traditional register file, because of the fact that the memory and datapath interfaces of the register file are shared and due to the multi-ported nature of these register files.

Since the VWR is single ported, it is important that data that needed the same cycle/operation are placed in different VWRs. Arrays are mapped on the VWR during a separate mapping process (explained further in Section 4). Scalar data like scalar constants, iterators and addresses etc. can be mapped to a separate Scalar Register File (SRF) in order not to pollute the data in the VWR with intermediate results[4].

[3]multiple contiguous words in the same row of the SRAM

[4]Since we target the data-arrays and large data-structures and due to space constraints, the details of the SRF are not discussed further in this

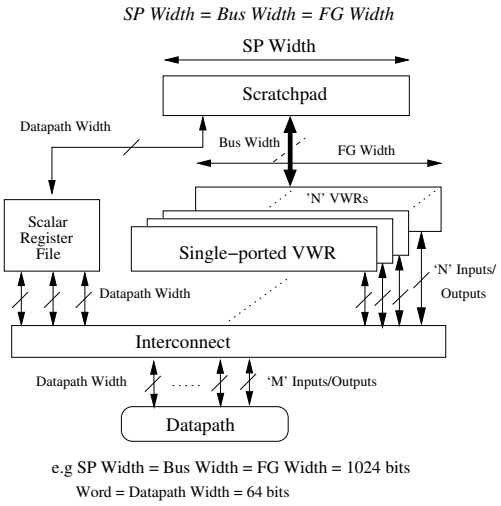


**Figure 3. VWR and Scalar Register File connectivity to the datapath and the L1 Memory**

### 3.3 Connectivity Between VWR and Datapath

The foreground memory consisting of VWRs and SRF can be connected to any datapath organization (consisting of multipliers, adders, accumulators, shifters, comparators, branch-unit etc.) by replacing the register file. Figure 3 shows the connectivity between the VWRs, SRF and the datapath. The datapath may or may not support sub-word parallelism similar to state of the art processor engines like Altivec, MMX or SSE2.

Once the appropriate data are available in the foreground memory, the decoded instruction steers the read and write operations from and to the foreground memory and the datapath. At a given cycle, one word (consisting of subwords) will be read from the VWR to the datapath and the result will be written back to a VWR. The foreground VWR organization along with the datapath is shown in Figure 3.

## 4 Example Operation

Figure 4 presents the operation of the VWR on simplified example code, assuming a 32-bit processor datapath and a 256-bit line-size. This means that one VWR at any given point in time can store 8 words. For the sake of simplicity no subword parallelism or vector parallelism is used in this example, of which Figure 4 shows the C code (with intrinsics). The asymmetric interface of the VWR, having a wide connection to the memory (width is complete row of the scratchpad) and a narrow connection of one word wide to the datapath, results in the following mode of operation: a complete row of the scratchpad is copied to the VWR at once, using a *LOAD_row*. In this example operands from arrays b and c are allocated to two different rows in the scratchpad and to two different VWRs (VWR 1 and 2).

paper.

```
Original Code:

for(i=0; i<64; i++) {
  a[i] = b[i] * c[i];
}
```

```
Modified Code with VWR:

for(i=0; i<8; i++) {
  LOAD_row VWR2, b[i*8];
  LOAD_row VWR1, c[i*8];
  for(j=0; j<8; i++) {
        VWR3[j] = VWR2[j] * VWR1[j];
  }
  STORE_row VWR3, a[i*8];
}
```

**Figure 4. Re-written C code with Very Wide Registers and load/store operations**

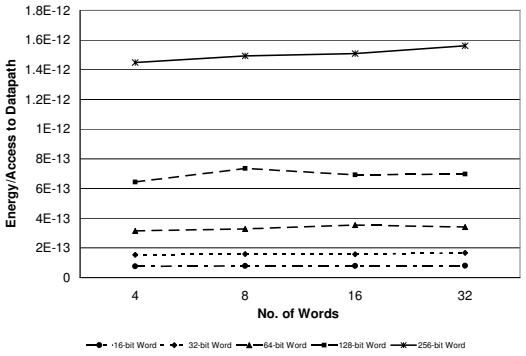


**Figure 5. 1-ported VWR Energy/Access to datapath**

Therefore two rows are loaded. In the next phase these operands are consumed one by one by the inner loop and the results are stored in a third VWR (VWR 3). Only when all computations of the inner loop are finished, the complete VWR 3 is stored back to the scratchpad.

Because in the embedded signal processing systems domain (including the benchmarks used here), most data is streaming and continuous, it is reasonable to assume that most of the time complete lines of the scratchpad can be loaded with relevant data. It is still possible to load partial rows if not enough independent data words can be found to fill a complete row (for instance at the end of a loop).

Currently the allocation of arrays to the VWR is done using *intrinsics* (like *LOAD_row*, *STORE_row*, etc.). To perform register allocation to the VWR, the benchmarks arrays need to have *affine linear index expressions*. The compiler phase required for the VWR allocation is being studied and preliminary results are promising. A detailed discussion of the compilation falls outside the scope of this paper.

## 5 Experimental Setup and Results

Both the proposed and the base line architecture are simulated using the CRISP [15] framework. CRISP is a cycle accurate instruction set simulator extension of the Trimaran [2] framework. Different architectures like VLIWs, RISC and VWR-based architectures are modeled in the CRISP framework, which generates the activation trace for the different components of the processor.

The different components are implemented in VHDL and synthesized using *Synopsys Physical Compiler* using *DesignWare* components. All the components are synthesized to a clock frequency of *200*MHz (typical frequency for embedded processor) using a TSMC *90nm* general purpose technology library running at *1V* Vdd, worst case design corner. The area and energy consumption of the 90 nm memories was taken from Cacti 4.1 [7]. The sizes of the memories used in the experiment are described in Section 5.2. The area of the different modules generated from *Physical Compiler* and taken from Cacti are ported to *Magma Fusion Blast*, where placement and routing is done[5]. Memories from SRAM generators like Artisan could also be used instead of Cacti. The capacitance of the routing for each component is back-annotated and re-synthesized in *Physical Compiler* and the energy/access of each component is computed. Since wide L1 memories and VWR are used here, the energy overhead due to interconnect bus could become large if the interface bus is incorrectly routed. The interconnect has been modeled (in both the VLIW and the VWR based register files) accurately using the detailed results from place and route and is taken into account in the energy estimations. The energy consumption of the clock tree network of every component is included in the energy consumption of the component itself. Due to space constraints a detailed discussion on the layout is not included in this paper [19]. Accurate energy/access numbers are obtained from the physical design and are combined with the activation trace for each component from the instruction set simulation, providing the net energy consumption of each component for any simulation run.

## 5.1 VWR Parameter Exploration

A Very Wide Register or VWR is a parameterizable foreground memory organization, replacing the traditional register file. Therefore different parameters like number of words, size of the word can be varied for each VWR. Figure 5 shows the variation in energy/access of the VWR from the datapath, when both the number of words as well as the size of the word are varied. Figure 6 shows the same variation in case of the access toward the memory (when all the words in the VWR are read from or written to).

Figure 5 shows that as the size of the word is changed from 16-bit to 256-bit, the energy/access towards the datapath changes linearly. When the number of words is increased however, there is only a slight increase in the energy/access. This increase is more apparent for larger sizes of words due to the fact that the MUX/DEMUX structure for larger word sizes is more complex. Figure 6 shows that the energy/access also increases linearly for increasing word sizes and increasing number of words.

The experiments show that the energy/access of the

[5]For Place and Route tools, internal proprietary memories were used as Cacti provides only area estimates.

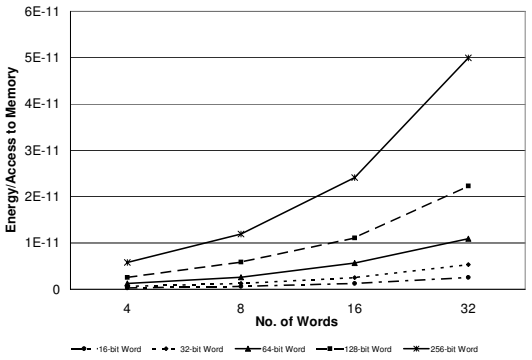


**Figure 6. 1-ported VWR Energy/Access to memory**

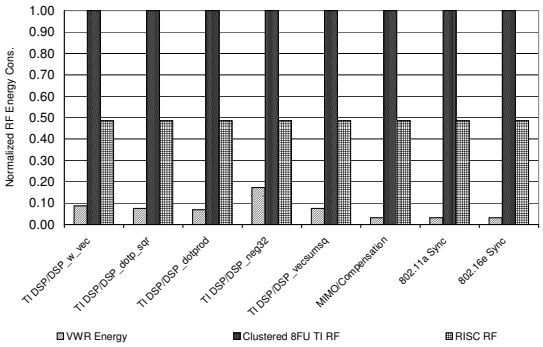


**Figure 7. VWR Energy comparison over TI DSP Benchmarks and Wireless Benchmarks**

VWR towards the datapath and memory is always lower than that of a register file with 3 or more ports with the same storage space and the same memory footprint. The gains of the VWR with respect to register files increase dramatically as the number of ports is increased, as additional ports increase the energy/access of the register file dramatically.

## 5.2 Benchmarks and Energy Savings

For comparison with other register file configurations, a clustered 8FU VLIW register file (similar to TI's TMS320C64x [18] 12 ports/cluster, 32 registers deep/cluster) and a single issue RISC (3 ports, 16 deep) are used as the base line processors. Since the VWRs are single ported, 4 VWRs are used to allow 4 (R/W) accesses in parallel. Each VWR contains 8 words of 32-bit each. A 32KB of memory is used in all the three architectures, but the output is 256-bit for the VWR case and 32-bits for the RISC and the VLIW.

The TI DSP benchmark suite [1] and realistic benchmark kernels from Software Defined Radio (SDR): 802.11a synchronization, 802.16e synchronization, MIMO feedback loop compensation, are used to evaluate the three different architectures described above. Figure 7 shows the energy gains of using a VWR for each of the different benchmarks. The energy consumption has been normalized to the energy consumption of the 8FU VLIW's register file for each

benchmark. The VLIW register file's energy consumption is not much higher than that of the RISC, because it has been clustered. It can be seen that over all the benchmarks, the VWR based architecture saves on average of about *10X* in energy with respect to the register files of the 8FU clustered VLIW based architecture. For the benchmarks used, about 5% of the total energy spent in the VWR is spent on loads and stores from the memory. This cost is reduced by a factor of *14X*, because the activity is substantially reduced by loading/storing complete rows to the VWR and because the access cost is also lower for the wide load/store compared to the register file. The remaining 95% of the VWR energy is spent in accessing it from the datapath. This part of the cost is reduced by a factor of *10X*, and hence the total energy consumption of the register file is reduced just over a factor *10X*. Note that the VLIW, RISC Register Files as well as the VWR can be sub-banked and microarchitecturally optimized furthur, thereby giving a different relative gain of the VWR with respect to the multi-ported register file. Such optimizations are outside the scope of this paper and are complementary to the architecture presented. Complete rows of data could not always be loaded from the wide memory to the VWR for the data layout of the benchmarks, but the overhead of the partial loads was observed to be low. Since Energy/Access of the VWR has a much lower cost, large energy savings exist compared to a multi-ported VLIW.

Figure 8 shows the energy breakdown between the memories, buffers to drive the bus and the register file for the VLIW and the VWR based architectures. The pins of the VWR are aligned and matched to the pitch of the sense amplifier of the memories to reduce the net capacitance of the bus and the energy required to drive them. The VWRs consist of cells that are connected to only one bitline and one wordline each, which results in a smaller net-capacitance that has to be driven. Therefore VWR designs can be clocked faster than multi-ported register files. The larger buffer cost in case of the VLIW register file is the result of the unified interface and multi-ported nature of a traditional register file to both the datapath and the memory. The extra cost of the bus that connects this port to the memory has to be paid every time the register file is accessed from the datapath. The high access frequency of the VLIW ports towards the datapath results therefore in a high energy penalty. Due to the wide nature of the SRAMs, the energy consumption in the SRAMs is also reduced by about 40% for the same amount of data transfered. This results in a total reduction of more than 60% in energy consumption in the data memory and the register file, which form a major part of the total energy consumption of a VLIW.

## 6 Conclusion

In this paper we have presented a novel register file architecture with asymmetric interfaces to the memory and the datapath. Different application kernels were mapped on the VWR. Compared to a clustered VLIW register file, about 10x reduction in energy consumption on different benchmarks was demonstrated. The interconnection between the memory and the VWR was optimized and the achieved energy gains remove most of the overhead. We are currently working a compiler with automated register allocation of arrays on the VWR and the design of the wide memories.

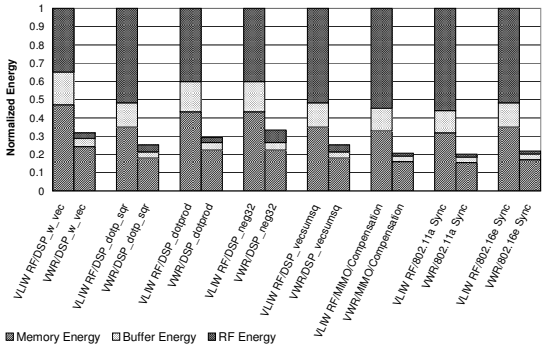


**Figure 8. Energy consumption Split between register file, buffers and register file**

## References


[1] *TI DSP Benchmark Suite* http://focus.ti.com/docs/toolsw/folders/print/sprc092.html.

[2] *Trimaran: An Infrastructure for Research in Instruction-Level Parallelism*. http://www.trimaran.org, 1999.

[3] B. Amrutur et al. Speed and power scaling of SRAM's. In *IEEE JSSC*, vol. 35, Feb 2000.

[4] K. Asanović. *Vector Microprocessors*. PhD thesis, University of California Berkeley, 1998.

[5] H. Corporaal. *Microprocessor Architectures : From VLIW to TTA*. John Wiley & Sons, 1998.

[6] P. Evans, et al. Energy consumption modeling and optimization for SRAM's. In *IEEE JSSC*, vol 30, May 1995.

[7] HP Research, http://quid.hpl.hp.com:9081/cacti/index.y. *Cacti 4.1*.

[8] Improv Systems, Inc, http://www.improvsys.com. *Jazz DSP processor: Product Brief*, 1999.

[9] J. Janssen and H. Corporaal. Partitioned register file for TTAs. In *Proc. of Micro*, pages 303–312, 1995.

[10] M. Kandemir, et al. Compiler-directed scratch pad memory optimization for embedded multiprocessors. In *IEEE Trans on VLSI*, pages 281–287, March 2004.

[11] U. J. Kapasi, et al. Programmable stream processors. *IEEE Computer*, Aug. 2003.

[12] C. E. Kozyrakis et al. Scalable vector processors for embedded systems. *IEEE Micro*, 23(6):36–45, 2003.

[13] A. Lambrechts, et al. Power breakdown analysis for a heter. NoC platform running a video application. *Proc of ASAP*, Jul 2005.

[14] V. Lapinskii, et al. Application-specific clustered VLIW datapaths: Early exploration on a parameterized design space. *IEEE TCAD*, 21(8):889–903, Aug 2002.

[15] P. OpDeBeeck, et al. CRISP: A template for reconfigurable instruction set processors. In *Proc of FPL*, Aug 2001.

[16] S. Rixner, et al. Register organization for media processing. In *HPCA*, Jan 2000.

[17] J. Sánchez et al Modulo scheduling for a fully-distributed clustered VLIW architectures. In *Proc of MICRO*, Dec 2001.

[18] Texas Instruments, Inc, http://www.ti.com. *TMS320C6000 CPU and Instruction Set Reference Guide*, Oct 2000.

[19] J-B.Domelevo, Working on the Design of a Customizable Ultra-Low Power Processor: A Few Experiments Masters Thesis, ENS Cachan Bretange/IMEC, Sep 2005.