VECTOR PROCESSING AS AN ENABLER FOR SOFTWARE-DEFINED RADIO IN HANDSETS FROM 3G+WLAN ONWARDS

C.H. (Kees) van Berkel[1, 2], Frank Heinle[3], Patrick P.E. Meuwissen[1], Kees Moerman[3], Matthias Weiss[3]
[1] Philips Research, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
[2] Technical University Eindhoven [3] Philips Semiconductors
E-mail: kees.van.berkel@philips.com

ABSTRACT

A major challenge of software-defined radio (SDR) is to realize many GIPS of flexible baseband processing within a power budget of only a few hundred mW. A heterogeneous hardware architecture with a programmable vector processor as key component can support WLAN, UMTS, and other standards. For handsets SDR baseband is feasible today, has many flexibility advantages, and saves silicon area.

1. INTRODUCTION

Future mobile handsets will need to support multiple wireless communication links, potentially including 2G cellular, 3G cellular, wireless local-area network (WLAN), personal-area network (PAN), broadcast, and positioning. A layered structure of such a future network, adapted from [1], is shown in Figure 1 and Table 1.

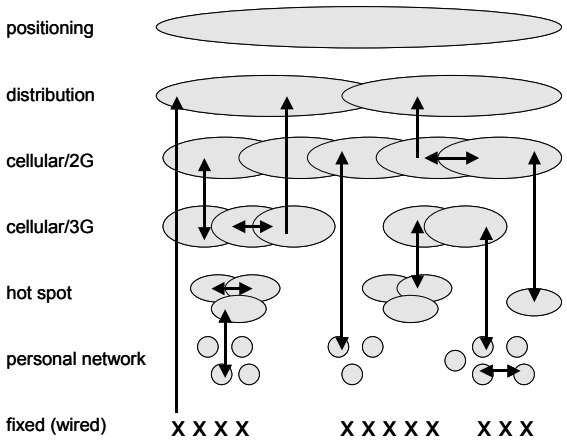

Figure 1 Layered structure of a seamless future network

These layers are to be integrated in a common, flexible, and seamless IP core network, supporting global roaming and a single access number per user. This requires both horizontal (intra system) and vertical (inter system) handover, as indicated by the arrows.

For each of these layers there exists a multitude of, often regional, standards. Some handsets may have to support multiple standards per layer, e.g. in a world phone.

layer	link range [$\log_{10}$ m]	up/down	mobility	examples
positioning	6-7	d	full	GPS, Galileo
distribution	5-6	d	full	DAB, DVB-T/H
2G cellular	4-5	d,u	full	GSM, IS95, PHS
3G cellular	3-4	d,u	full	UMTS, CDMA2000 TD-SCDMA
hot-spot	2-3	d,u	local	802.11 a,b,g, wifi
personal	1-2	d,u	local	BT, DECT, UWB
fixed	0-1	d,u	none	POTS

Table 1 Layers of a future seamless network.

Individual standards typically evolve over the years towards higher bit rates, more features, and more services. For example, 3G cellular standards will need to support high-speed downlink packet access (HSDPA), and for WLAN multiple-antenna schemes are being studied (IEEE 802.11n).

For a given standard, new algorithms are continuously developed to improve performance (lower bit-error rate, more efficient spectrum usage). Upgrading handsets by software would then be attractive, possibly by down-loading of new software versions over the air interface.

In a typical user case, multiple standards have to be supported in standby mode, plus one standard active. In a high-end scenario, however, several links may be active simultaneously, e.g. DVB-T (data downlink) UMTS (matching uplink), GSM (standby), BT, and GPS.

The combination of the above trends is sometimes referred to as "4G" wireless. They form a powerful argument for so-called software-defined radio (SDR) [2].

2. ARCHITECTURES FOR SDR BASEBAND

The digital baseband processing for SDR can be split into three stages: a filter stage, a modem stage, and a codec stage, as shown in Figure 2. Estimates for the computational load [GHz] for baseband processing are indicated in Figure 3. Interestingly, the numbers roughly apply to both compiled

programs on an X86 and to optimized assembly running on current GSM DSPs.

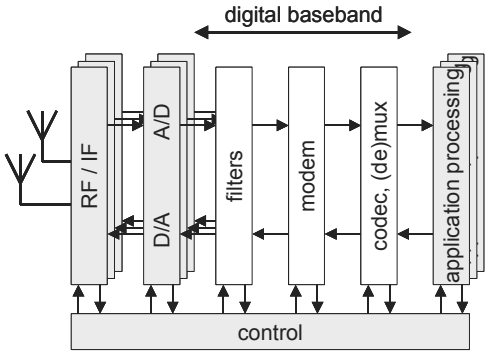

Figure 2 A crude SDR architecture, with the baseband section split into filters, modem, and channel codec.

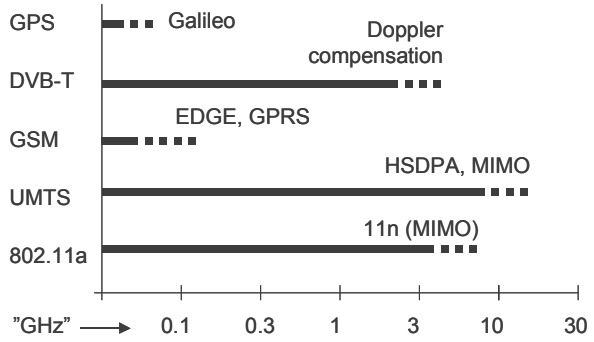

Figure 3. Load estimates for various SDR standards.

These loads are more or less evenly distributed across the three baseband stages. Nevertheless, the stages have very different characteristics.

2.1. Filter stage

Various transmitter and receiver filters are required for band limitation, e.g. (root) raised cosine filters and sample-rate conversion. Given their high computational load (e.g. 2-5 billion multiplications and additions per second for UMTS), their regularity, and commonality among the algorithms involved, full programmability would add little value. Moreover, a generic DSP would consume too much power. A configurable filter is more appropriate.

2.2. Modem stage

The modem stage, sometimes called "inner transceiver" or "signal conditioner" appears to be the most diverse across the different standards. It includes functions such as rake reception, correlation, synchronization, joint detection, equalization, FFT, OFDM (de)mapping, cordic, matrix multiplication and inversion, etc. Furthermore, new modulation schemes are proposed within the continuous evolution of standards to improve throughput and performance. Also manufacturers are challenged to differentiate their products by improving algorithms to reduce BERs or transmit power for the same BER. This is the stage where programmability offers most value!

2.3. Codec stage

The codec stage, sometimes called "outer transceiver" involves a variety of functions: (de) multiplexing, (de) puncturing, (de) interleaving, and a variety of channel codecs (e.g. convolution, Turbo, Reed-Solomon). The performance of these functions is determined by standard algorithms, and allows little differentiation among manufacturers. Given the considerable similarities among standards and algorithms, and given the considerable processing requirements for higher bit rates (> 100 Mbps), a fully programmable solution does not appear to be justifiable. See also Sections 4.4 and 5.

2.4. Baseband hardware architecture

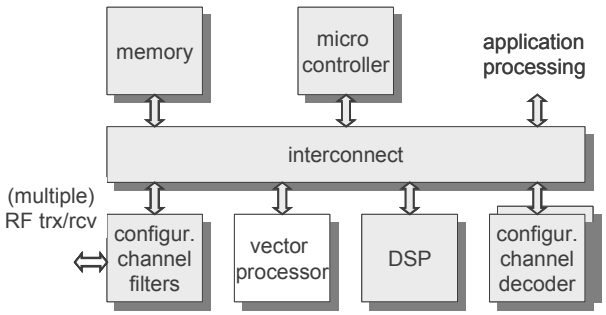

Figure 4 Schematic hardware architecture for SDR/BB.

The above observations on the different baseband stages result in a proposal for a multi-standard hardware architecture (Figure 4), comprising:

- a general purpose microcontroller for link/MAC layer processing and for controlling baseband and RF tasks;
- a conventional DSP for intrinsically "scalar" algorithms and legacy code, e.g. speech codecs;
- one or more multi-standard weakly configurable channel decoders, e.g. Viterbi, Turbo;
- a programmable vector processor for number crunching, mostly in the modem stage;
- a configurable filter processor.

3. THE ONDSP AND EVP VECTOR PROCESSORS

Vector processing can be used to exploit the abundant and often regular parallelism encountered in many baseband-processing algorithms. Using SIMD instructions (Single Instruction Multiple Data) arithmetic operations or load/store operations can be applied to P (e.g. P=16) samples in parallel. For several DSP algorithms this will be explored in Section 4.

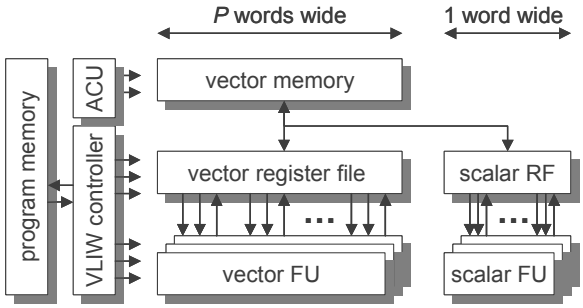

Figure 5 A generic vector-processor architecture.

The basic features of a vector-processor suitable for SDR are listed below, with reference to Figure 5:

- The dominant data size is 16 bits as in conventional DSPs, with some support for 8-bit and 32-bit data. Hence a single SIMD vector comprises P 16-bit elements, $2P$ 8-bit elements, or $P/2$ elements of 32 bits.
- The main data types are integer and fixed point, with support for complex numbers (2×8 or 2×16 bits).
- The vector memory (VM) supports one vector read or vector write (P words) per clock cycle.
- The VLIW execution model (Very Long Instruction Word) supports parallelism among multiple vector functional units (FUs), e.g. MAC, ALU. This VLIW parallelism comes *in addition to* vector parallelism.
- *On top of that* a VLIW instruction may *also* specify several operations on scalar functional units.
- To keep many functional units busy, there is extensive support for address calculations (ACUs, e.g. post-increment, modulo) and for zero-overhead looping.

3.1. OnDSP

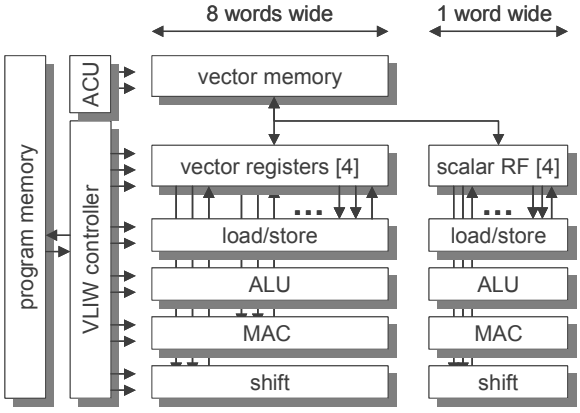

Figure 6 The OnDSP architecture.

The OnDSP vector processor is a key component of several multi-standard programmable Wireless LAN baseband ICs [3]. The application of vector processing to WLAN will be addressed in some detail in Section 5.1.

The OnDSP architecture is depicted in Figure 6. The vector size equals 128 bits (P=8). A single VLIW instruction can specify a number of vector operations, e.g. load/store, ALU, MAC, address calculations, and loop-control. OnDSP supports a couple of specific vector instructions, including word insertion/deletion, sliding, and gray coding/decoding. VM addresses must be multiple of P. Program code is compressed vertically ("Tagged VLIW").

3.2. EVP

The EVP (Embedded Vector Processor) is a productized version of the CVP [4]. Although originally developed to support 3G standards, the current architecture proves to be highly versatile. Care has been taken to cover the OnDSP capabilities for OFDM standards and also many media algorithms perform well.

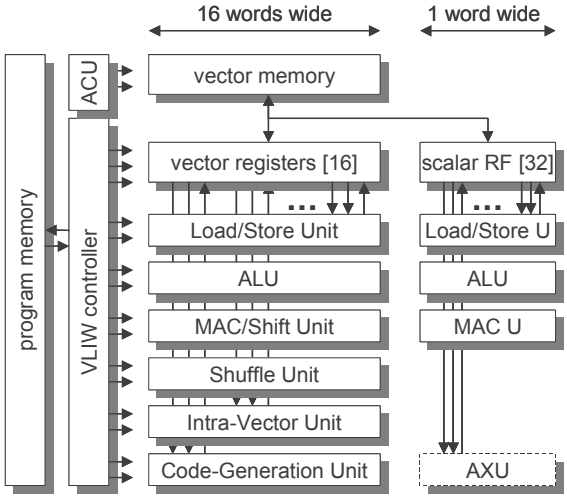

Figure 7 The EVP architecture.

The EVP architecture is depicted in Figure 7. The SIMD width is scalable, and has been set to 256 bits (P=16) for the first product instance. The maximum VLIW-parallelism available equals five vector operations *plus* four scalar operations *plus* three address updates *plus* loop-control. Specific FUs of the EVP include the following.

- The Shuffle Unit can be used to rearrange the elements of a single vector according to an arbitrary pattern.
- The Code Generation Unit supports CDMA-code generation: in a single clock cycle 16 successive complex code chips are generated. The unit can be configured for a variety of codes (UMTS, CDMA2000, GPS, etc.) and for cyclic redundancy checks (CRC).
- The Intra-Vector Unit supports operations such as add (or take the maximum of) the elements of a single vector in, e.g. in four segments of four elements each.

Programs are written in EVP-C, i.e. in terms of function intrinsics for vector operations, in a C-syntax. An assembler does register allocation and VLIW instruction scheduling.

In a 90 nm CMOS process, the EVP core measures about 2 mm^2 (450 k gates), runs 300 MHz (worst case), and dissipates about 0.5 mW/MHz (core only) and 1 mW/MHz including a typical memory configuration. These numbers are based on gate-level simulations of annotated netlists.

4. VECTORIZATION OF BASE-BAND KERNELS

Making vector parallelism explicit in program code is commonly called "vectorization". Depending on the algorithm at hand, this can be relatively straightforward, or it may require some ingenuity. Unfortunately, the state of the art of vectorizing compilation today cannot fully exploit the architectures discussed above *and* at the same time achieve *efficient* code for SDR. Although we have been able to generate code of acceptable efficiency for some algorithms, we rely on manual vectorization for the time being. The results for several key algorithms are presented below.

4.1. Fast Fourier Transform

The FFT is one of the fundamental DSP algorithms. Together with its inverse it is a key algorithm for OFDM standards such as 802.11a (cf. Section 5.1). The basic computation is the FFT butterfly applied to pairs of complex samples, say, 2×12 bits. On a vector processor $P/4$ butterflies can be computed in parallel conveniently. The vectorization challenge is how to rearrange the elements of a block between successive layers of butterflies. For OnDSP and EVP this required handcrafted shuffle instructions.

Processor	ref	code	clock cycles	SIMD
EVP		Opt	64	16×16
OnDSP		Opt	160	8×16
Tigersharc	[5]	Opt	174	2×8×16
VIRAM		Opt	357	16×32
TMS320C6203	[6]	Opt	646	N.A.
Altivec MPC7447	[6]	Opt	956	8×16
Carmel 10xx	[6]	Otb	5568	N.A.
AMD K6-2E+/ACR	[6]	Otb	10751	N.A.

Table 2 Cycle counts for a 64-point complex FFT.

Table 2 shows the cycle counts for a 64-point FFT (incl. bit-reversal) for a number of DSPs. The numbers from [6] have been scaled from a 256-point FFT in proportion to the number of butterflies. The column 'code' specifies whether the program has been compiled `out-of-the-box' ('otb'), i.e. without any manual intervention, or whether it has been hand optimized ('opt') at the assembly level. The EVP uses its 16 multipliers effectively during 48 / 64 clock cycles. The prototype EVP-C scheduler requires 79 cycles.

4.2. Golay correlator for UMTS

In an UMTS-FDD receiver, a Golay correlator is used for the acquisition of base-station signals. It is basically a filter designed specifically to detect correlation peaks of the 256-chip long primary synchronization code (PSC) [7]. The irregular FIR taps require non-aligned accesses to the vector memory. Vectorization is relatively straightforward: P successive output symbols can be computed in parallel. Automatic scheduling on the EVP requires 22 cycles for P symbols. Manual optimization of the memory accesses and schedule reduces this to 16 cycles for P symbols.

4.3. Rake receiver for UMTS

A rake receiver is commonly used to combine the resolved multipath signals in CDMA standards. For a UMTS handset a rake receiver is used for multiple data and control channels, with up to six rake fingers per channel. A basic UMTS rake finger involves code generation (scrambling + channelization), descrambling, despreading, dual weighting, and STTD combining.

On the EVP the inner loop of this rake finger keeps most FUs busy. A single EVP VLIW instruction specifies:

- load and align a sample vector of P data chips,
- generate a vector of P code chips,
- correlate a vector of data and a vector of code chips,
- intra-add result to one or few symbols ($SF \leq 16$) or to one partial symbol ($SF > 16$).

Table 3 summarizes the load of a rake finger on various programmable DSPs.

processor	ref	load [MHz]	arithmetic resources (*complex arithmetic*)
EVP	[4]	1	16×(*MAC+ ALU + PN gen.*)
4 UMTS DP	[8]	25	4×(*MAC+ ALU + PN gen.*)
UMTS DP	[8]	25	1×(*MAC+ ALU + PN gen.*)
TI C62	[9]	40	
Carmel	[8]	125	2 MAC + ALU
TI C54x	[8]	300	1 MAC/ALU

Table 3 Load [MHz] of a UMTS-FDD rake finger.

4.4. Viterbi and Turbo Decoding

Viterbi decoders are probably the most common channel decoders. They are quite computationally intensive and comprise two distinct types of computation: trellis construction and trace back. The former allows fairly straightforward vectorization, making effective use of vector shuffles. Trace back is inherently a sequential operation. The EVP allows both types of computation to be scheduled in parallel on the vector and scalar paths respectively [10]. Table 4 benchmarks several DSPs for a 12 kbps UMTS AMR voice channel (constraint length K=9). The 3.5 butterfly operations/clock cycle for EVP translates to a ten cycles/symbol for a decoder with constraint length 7.

Table 4 also provides load numbers for turbo decoding (3GPP, with 6 iterations). The EVP cycle count is an estimate, based on hand schedules.

Processor	ref	Viterbi cycles /symbol	Viterbi butterflies /cycle	Turbo cycles /symbol
EVP	[10][4]	37	3.5	55
TigerSharc	[5]	70	1.8	70
TMS320C6400	[11]	170	0.8	440
Altivec	[12]	260	0.5	

Table 4 Viterbi and Turbo decoding on several DSPs.

SDR RESULTS

5.1. Wireless LAN

The Philips 802.11 a/b/g baseband implementation is based on the OnDSP vector processor [3]. Below we focus on the IEEE 802.11a standard. From our discussion on Viterbi decoders (Section 4.4), we can conclude that for the symbol rates at hand (up to 54 MHz) it is not practical to map Viterbi decoding on a vector processor. Hence, the OnDSP is employed for signal conditioning tasks, while hardware accelerators support Viterbi decoding, (de)interleaving, and de(scrambling). The main tasks for the vector processor are:

- preparing transmission data (TX),
- equalizing and tracking when receiving data (RX),
- burst detection and acquisition.

OnDSP load (de)modulator task	TX [cycles]	RX [cycles]
symbol (de)mapping	35	35
pilot generation	55	
pre (post) scaling	35	35
tracking		36
channel correction		39
OFDM (de-) mapping	35	35
afc(I) FFT	160	160
TS frequency shift	40	
phase-error correction		39
CP insertion/removal	35	0
control code	40	40
overall peak load	*435*	*414*

Table 5 OnDSP load for the critical loop of the 802.11a (de)modulator [cycles per OFDM symbol of 4 µs each]

The OnDSP cycle counts of Table 5 yield an OnDSP load of 100 MHz. For the EVP, with twice the parallelism, this will go down to approximately 50 MHz. In addition to meeting the OnDSP load constraint, the OnDSP can also cope with the tight real-time constraints for synchronization.

Interestingly, software flexibility does *not* increase the silicon area for this application. Unlike, e.g. [14] the same resources are used for both synchronization and FFT.

5.2. UMTS

Today's GSM handsets deploy programmable DSPs for all baseband signal processing, with all the associated flexibility benefits. Moreover, it has allowed concurrent and independent evolution of DSP architectures (following Moore's law) and algorithmic improvement. Different groups of designers, often in different companies used the DSP architecture and tools as interface between them.

For 3G standards, such as UMTS, this is not entirely practical, at least for the time being. From Section 4.4 we can see that Turbo decoding alone requires about 55 clock cycles per symbol on the EVP. For UMTS 3GPP R'99 this results in an acceptable 35 MIPS for a 640 kbps channel. For 3GPP release 5, however, a 14 Mbps data rate would result in an EVP load in excess of 700 MHz, and a power consumption close to 1 Watt in 90 nm CMOS. Furthermore, 3G+ standards show a much more dynamic computational load than for 2G standards, both due to the nature of the employed algorithms and the large number of different use cases. This can be illustrated for four UMTS scenarios [15]:

1. UMTS idle mode, only multi-cell synchronization.
2. UMTS R'99 connected mode with flat fading conditions (1 rake finger only), 3 dedicated channels (DCH), and neighbor-cell broadcast channel (BCH) monitoring.
3. UMTS R'99 connected mode in a scattering environment with multiple paths (six rake fingers) with the same transport channel configuration as before.
4. UMTS R'99 connected mode in a scattering environment with multiple paths (six rake fingers) with the same transport channel configuration plus an HSDPA (High Speed Downlink Packet Access) link (3GPP R5) with 15 downlink channelization codes.

The EVP load numbers for these scenarios are summarized in Table 6. Note the variation in load distributions!

	EVP load/scenario MIPS]			
UMTS task	1	2	3	4
PSCH search (Golay)	25	25	25	25
CPICH search	98	17	17	17
CPICH dispreading		4	22	33
CPICH symbol rate		1	1	2
DCH dispreading		3	16	16
DCH symbol rate		1	6	6
HS-SCCH dispreading				15
HS-SCCH symbol rate				1
HS-DSCH dispreading				12
HS-DSCH symbol rate				22
Overall peak load	*123*	*51*	*87*	*149*
Overall average load	*4*	*28*	*64*	*126*

Table 6 EVP loads for the modem stage; 4 UMTS scenarios.

More advanced receiver algorithms such as interference cancellation, chip-rate equalization, or joint detection will be required in the future, both to increase the system capacity

and to improve the reception quality. This is from our point of view one of the most compelling justifications for SDR.

5.3. Multi-standard considerations

EVP loads for the modem stage for various wireless standards are shown in Figure 8. Note the considerable headroom available on the EVP for most standards.

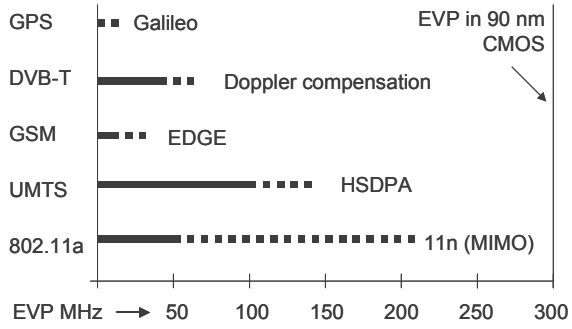

Figure 8 Estimated EVP loads for various modem stages.

This available headroom can in be used:

- to introduce improved but more demanding algorithms,
- to scale the supply voltage to reduce power consumption, and, in principle,
- to run multiple standards simultaneously.

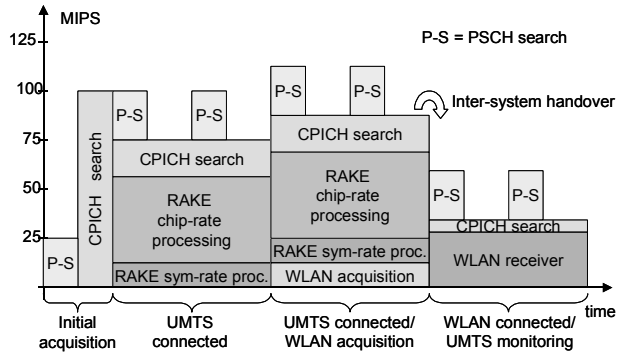

Figure 9 Handover from UMTS to WLAN, with load indications for the EVP.

The latter introduces *intra*-standard resource sharing, substantially reducing the additional costs for adding another standard. For the combination of WLAN and UMTS, including inter-system handover, this is illustrated in Figure 9. Supporting multiple standards simultaneously requires a carefully coordinated use of the resources in an SDR hardware architecture like the one of Figure 4.

CONCLUSION

The modem stage of an SDR requires software flexibility to cope with the multitude of wireless standards, their evolution, and with algorithmic improvement (including bug fixes). Vector parallelism in combination with VLIW can offer the computation power required for this. The OnDSP has demonstrated this for several WLAN ICs. The EVP, with its powerful FUs (Shuffle, Intra-Vector, Code Generation) outperforms conventional DSPs by an order of magnitude or more, in a power-efficient way. Accordingly, the EVP can be a key component of an SDR, where it can save silicon area by both intra-standard and inter-standard reuse. With 300 MHz, the EVP can potentially handle multiple standards simultaneously.

Acknowledgements

The contributions of Srinivasan Balakrishnan, Nur Engin, Rick Nas, and Rob Takken (all with Philips Research Labs), and of Wim Kloosterhuis, Jean Paul Smeets, and Mahima Smriti (all with PS) are gratefully acknowledged.

REFERENCES

[1] Reinhard Becher et al, "Broad-Band Wireless Access and Future Communication Networks", Proc. of the IEEE, Vol. 89, No. 1, pp. 58-75, Jan 2001.

[2] Software Defined Radio Forum. www.sdrforum.org.

[3] J. Kneip et al, "Single Chip Programmable Base-band ASSP for 5GHz Wireless LAN Applications", in *IEICE Trans. Electron., Vol.E85-C, No.2*, pp 359-367, 2002.

[4] C.H. (Kees) van Berkel et al "CVP: A Programmable Co Vector Processor for 3G Mobile Base-band Processing", Proc. World Wireless Congress, May 2003.

[5] J. Friedman, Z. Greenfield, "The TigerSharc DSP architecture", *IEEE Micro*, pp. 66-76, Jan 2000.

[6] Embedded Microprocessor Benchmark Consortium. http://www.eembc.hotdesk.com/.

[7] 3GPP TS 25.213, "Spreading and modulation (FDD) – (Release 5)", v. 5.5.0 (2003-12).

[8] U. Walther et al, "New DSPs for Next Generation Mobile Communications", *Global Telecommunica-tions Conference - Globecom '99*, pp. 2615-19, 1999.

[9] P.R. Dent, "W-CDMA Reception with a DSP-Based Software Radio*"; 3G Mobile Communication Technologies, Conference Publication No. 471*, pp. 311-315, 2000.

[10] Nur Engin, et al, "Viterbi Decoding on a Co-processor Architecture with Vector Parallelism", IEEE Workshop on Signal Processing Systems (SIPS'03), Seoul, August 2003.

[11] Tom Horner, Jelena Nikolic-Popovic, "Application of TMS320C6400 in 3G Wireless Infrastructure Transceiver" 2000, http://focus.ti.com/pdfs/univ/01-Wireless.pdf.

[12] L. Gwennap, "G4 Is First PowerPC With Altivec", in *Microprocessor Rep.*, Nov. 16, 1998, pp 17-19.

[13] John Glossner et al, "A Software-Defined Communications Baseband Design", *IEEE Communications Magazine*, pp. 120-128, Jan. 2003.

[14] Teresa H. Meng et al, "Design and Implementation of an All-CMOS 802.11a Wireless LAN Chipset", IEEE Communications Magazine, pp. 160-168, Aug. 2003.

[15] 3GPP TS 25.211, "Physical channels and mapping of transport channels onto physical channels (FDD) – (Release 5)", v. 5.5.0 (2003-09).