FADIC: Architectural Synthesis applied in IC Design.

J. Huisken, F. Welten

Philips Research Laboratories, Eindhoven, The Netherlands

Abstract

This paper discusses the design of a chip using architectural synthesis. The chip, FADIC, is applied in Digital Audio Broadcasting (DAB) receivers. It shows that architectural synthesis tools are used for the design of new complex applications and that it supports the evolutionary development of challenging applications like DAB. It was found that the success of such tools in the design community depends on the way user interaction is supported and stimulated. Fast and accurate feedback from the synthesis tools in combination with a rich set of hints for the compiler to guide the architecture exploration are the key issues. It is shown that short time to market is possible for implementations which are an order of magnitude more efficient than alternative implementations on commercially available DSP processors.

1 Introduction

The increasing emphasis on short time-to-market in combination with an increased complexity necessitates a high degree of design automation. One of the main challenges in IC-design automation is to simultaneously obtain a short design time and an area and power efficient product. Architectural synthesis [12], [3], [1], [2], [13], is a potential solution but has not yet found widespread use in industry today. Scepticism is particularly encountered in case of high volume design which is typically the case in consumer applications. The successful application of high level synthesis tools for the design of a commercial product for the consumer IC market can therefore be regarded as a milestone.

In section 2 we will introduce shortly the application at hand. Section 3 describes shortly a prototype built with general purpose DSPs. This is followed by the design strategy used in the specification and implementation of the FADIC chips in section 4. It shows the effective use of high level synthesis during the design.

2 The DAB system

The DAB system is designed to provide reliable, multi service digital sound broadcasting for reception by mobile and stationary receivers, using a simple non-directional antenna. It is meant to replace the existing AM/FM networks, and to provide easy program selection and additional data services. The introduction to the public has occurred at the “Internationale Funk Ausstellung” in Berlin September 1995. The same month the BBC started a regular service containing all five radio programs in Great London using a single frequency network. The single frequency mode is the first of the three operating modes in DAB, which is typically used for nation-wide coverage. The other two modes are meant for local and satellite broadcasting. DAB can be used from about 50 MHz until 2 GHz.

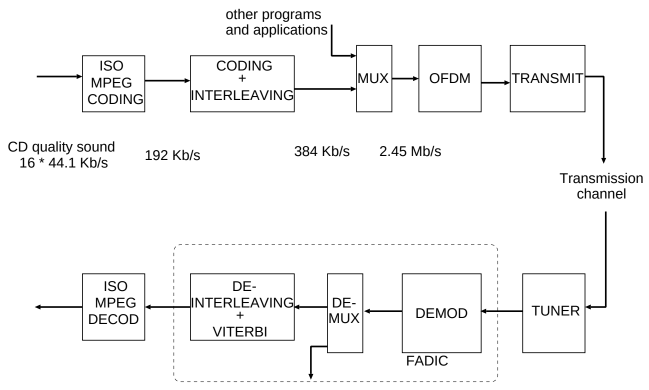

Figure 1: DAB system overview, the shaded area is the channel decoder

The system is designed in Eureka project 147 which has lead to an European Telecommunication Standard [6], to which FADIC fully complies. Development of FADIC was part of a JESSI application project. The DAB system is based on the technique of Coded Orthogonal Frequency Division Multiplex [10], in which demodulation is done using a Fast Fourier Transform. FADIC performs the demodulation function in which the FFT plays an important rôle. Another function which is also part of FADIC is the differential demodulation caused by the fact that QPSK is applied. The demodulator architecture is depicted in figure 2. The shaded part is subject of our high level synthesis approach.

A more elaborate description of the DAB system can be found in [9] and [15] where experimental DAB receivers and integration issues are shown. It is important to note that DAB is a new application which requires new architectures. This paper shows that high

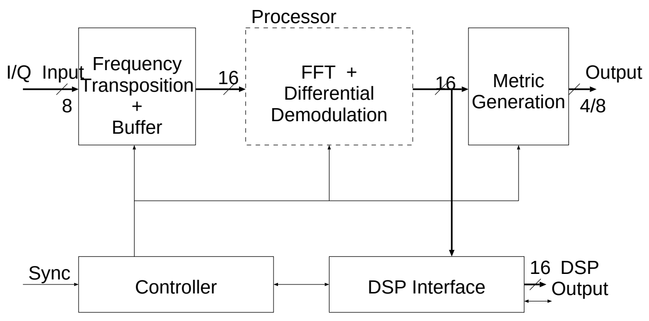

Figure 2: Birds eye view of the FADIC architecture: Besides the presence of a controller and a DSP-interface frequency transposition and metric quantization is performed.

level synthesis can play an important rôle in bringing these types of systems to the customer. The FADIC is used in the first consumer DAB receivers.

3 Prototype channel-decoder with DSPs

A major part of the demodulator is the calculation of a 2K complex point FFT. In figure 3 an early implementation [8] is shown using 4 Motorola 56156 DSPs.

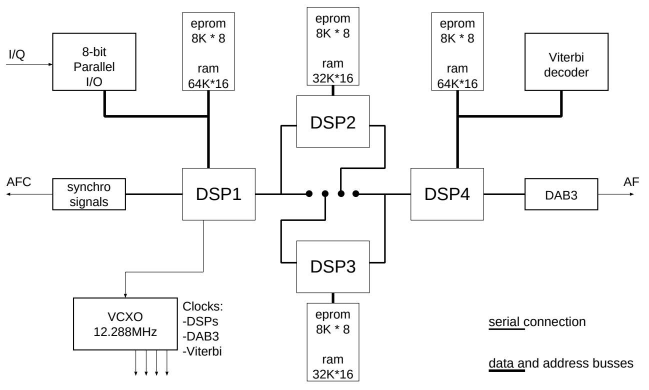

Figure 3: Prototype DSP board for DAB channel decoding

DSP1 Synchronization, clock control and program selection. For these purposes a parallel baseband I/O and a serial AFC output interface was added.

DSP2/3 FFT, differential demodulation and metric generation. The processors are switched on a frame basis.

DSP4 De-interleaving, interfacing to the sound decoder and Viterbi decoding.
One of the serial interfaces is used for data transfer to the sound decoder.

Two DSPs are needed to obtain enough performance for receiving an audio and a data service, i.e. about one third of the required performance. The performance is mainly determined by the number of FFT calculations. In assembly language the in-place FFT code amounts to ca. 200 lines of assembly code comprising 3 nested loops. In the inner loop the radix-2 butterfly calculations are done.

In assembly language the macro `butterfly()` contains 10 instructions, so the calculation of a complex butterfly takes about 20 clock cycles assuming a mean instruction duration of 2 cycles. For a 2K FFT $11 \times 1024 = 11264$ butterflies need to be computed. The board runs at 48 MHz leading to a measured 6 ms to perform a 2K complex point FFT including overhead in loop control. See

```
n_blocks = 1024;
n_butts = 1;
for (stage=0; stage<11; stage++)
  for (block=1; block<=n_blocks; block++)
    for (butt=1; butts<=n_butts; butt++ )
      butterfly();
    endfor;
  endfor;
  n_blocks = n_blocks / 2;
  n_butts = 2 * n_butts;
endfor;
```

Figure 4: Pseudo in-place FFT code

table 1 and realize that a radix-4 butterfly equals 4 radix-2 butterflies. The total assembly language code of such a DSP computing

FFT	DSP	FADIC
clock	48 MHz	12 MHz
radix-2 butterfly	20 cc	–
radix-4 butterfly	–	4 cc
2K complex FFT	6 ms	1 ms
power dissipation	>1 W	750 mW

Table 1: 2K complex FFT on DSP and FADIC

the FFT with differential demodulation and other required functions is about 1K instructions. Power dissipation is in the order of 1 Watt per DSP and the size of the board is 22×23 cm.

For DAB demodulation it is required that a FFT is performed in about 1 ms, with a preferred clock frequency of 12.288 MHz, and substantial less power consumption. It is obvious that, given the physical size, further integration is essential.

4 FADIC Design Strategy

From the above it is clear that there is a need for an application specific implementation because it can become much more efficient. The problem with application specific design is the long design time. Therefore high-level synthesis was investigated as a possibility to reduce design time and to manage design complexity. First the initial design strategy is described. After an evaluation it was found that a major change in the strategy was needed.

4.1 Initial design strategy

When we started this project High Level Synthesis was seen as a method for design where designers would specify the behavior (i.e. WHAT must be designed) at a high level of abstraction and where the tools were responsible for an efficient implementation (i.e. HOW the design is done). This would have the following advantages.

1. High level specifications are easy to understand, to manipulate, and to verify. This opens up the possibility for fast and reliable design and short time to market. Furthermore they are well suited as a starting point for derivative designs which always occur in practice.

2. This way it must be possible to relieve the designer from time-consuming implementation details while the efficiency of the implementation comes from the target architecture, the optimized module library and intelligent synthesis.

The first design exercises indicated that this initial strategy was insufficient. The following conclusions were drawn.

1. It was possible to design first time right silicon.
2. It was possible to start from specifications at a high abstract level. The Silage language from UC Berkeley was used [7].
3. But is was found that the strategy with respect to the implementation was insufficient. The generated layouts were too large (a factor of 2) to be acceptable.

The conclusion was that the strategy with respect to the specifications (the WHAT) was sufficient but that the strategy with respect to the implementation (the HOW) needed reconsideration. A more detailed analysis showed that the problems were related to data storage and transport and not to the computation units which were used in an efficient way. [4] Furthermore it became evident that the approach taken did not encourage user interaction in the design process. The major problem was that the effect of user interaction was not always predictable in terms of efficiency of the result.

4.2 New design strategy

Based on these results the design strategy was adopted as follows. The designer is still responsible for the specification but also for the global aspects of the implementation. An extensive set of pragmas (hints for the compiler) was defined such that the user is always in control. Examples of pragmas will be given later. The synthesis tools became transparent for the designer. The implementation of all the details remained a task for the tools. The new strategy was called: *What you write is what you get*. The design method became fast and interactive. Typically a designer uses the compiler as follows. Starting from a first version of the algorithm a mapping is performed using the tools. Feedback is provided on cost for storage, transport of data and computations. Based on this information the designer can optimize the algorithm and add information to improve the implementation. The design flow is shown in figure 5. This new strategy will now be discussed for the design of FADIC. The Mistral 2 tools of DSP Station are used.

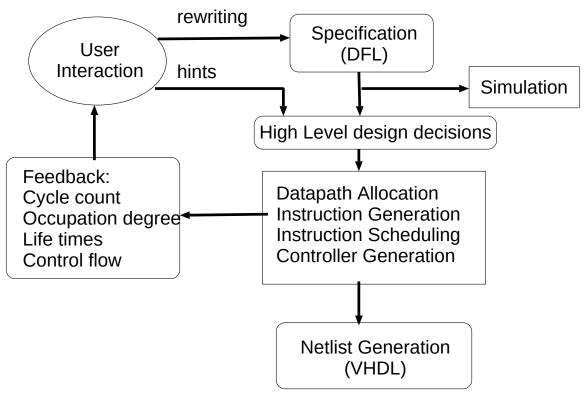

Figure 5: Mistral 2 design flow

4.3 FADIC design exploration

The major steps in the design method shown in figure 5 are now discussed in more detail.

2K complex FFT	radix-2	radix-4,2
# butterflies	11×1024	$5 \times 512 + 1024$
	$= 11264$	$= 3584$
# complex $+/-$	22528	22528
# complex $\times$	10240	7680
# ram accesses	45056	24576

Table 2: Number of operations in a radix-2 and radix-4 2K FFT

Specification The first action the user takes is the writing of a specification. The specification language DFL^{TM} from DSP Station^{TM} allows compact descriptions[1]. DFL is based on the Silage language from UC-Berkeley [7] but contains important procedural extensions which were needed, for example, to describe the different DAB operation modes of the chip. For the final FADIC description merely 450 lines of code were needed.

High level design decisions In the following step the designer can take a number of high level decisions which are related to the global architectural issues. This is typically the level at which the user is reasoning about his design. Each time such a decision has been taken the synthesis tools are used afterwards to evaluate the cost of the corresponding implementation. In this section a number of such high level decisions are discussed as examples.

First a clock frequency is selected. For reasons of power dissipation and EMC the clock frequency is not chosen too high. For FADIC 12.288 MHz is selected, also because it is an exact multiple of 48 kHz, a sampling frequency of ISO/MPEG digital audio [11]. Next the design space exploration can start. The computation bottleneck lies clearly in the calculation of a complex 2K point FFT with an overall time budget of 1.246 ms (or 15312 clock cycles). A first attempt was made using radix-2 FFT schemes. The number of butterflies to calculate is $1024 \times 11 = 11264$ which leads to the constraint of performing about one butterfly per clock cycle. It turned out that there was a bottleneck in the memory accesses.

Therefore we looked into schemes for radix-4 FFTs. A radix-4 butterfly needs to be executed in 4 cycles and 8 memory accesses to read and write are necessary. Figure 6 shows such a butterfly in more detail.

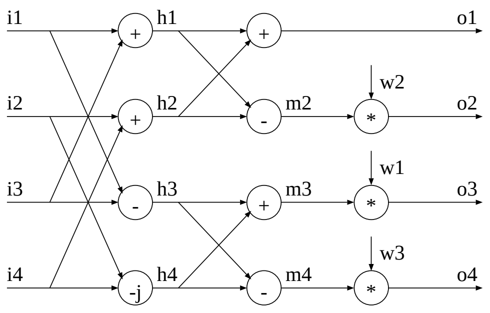

Figure 6: A complex radix-4 butterfly.

It shows that one complex radix-4 butterfly corresponds to $16 + 6 = 22$ real additions, $3 \times 4 = 12$ real multiplications and 16 RAM accesses. Since all these operations must be executed every 4 clock cycles this corresponds to a large parallelism. This parallelism is responsible for the complexity of this design. Even the scheduling of a single butterfly operation turned out to be extremely difficult when tried manually.

[1]DFL, Mistral 2, and DSP Station are registered trademarks of Mentor Graphics Corp.

time slot	ACU1	RAM1	RAM2	ACU2
54	decr	write(o1)	i4=read()	add
55	decr	write(o2)	i1=read()	incr
56	decr	write(o3)	i2=read()	add
57	add	write(o4)	i3=read()	add
time slot		ALU1	ALU2	
54		sub(h4,h3)	add(i3,i1)	
55		add(i4,i2)	sub(i3,i1)	
56		subj(i4,i2)	sub(h2,h1)	
57		add(h4,h3)	add(h2,h1)	
time slot	ACU3	ROM	MPY	
54			mul(m3,w1)	
55	k1=ag()	w3=read()	mul(m4,w3)	
56	k3=ag()	w2=read()		
57	k2=ag()	w1=read()	mul(m2,w2)	

Table 3: Four cycle schedule of a radix-4 complex butterfly –folded– loop, the MPY only contains a pipeline.

If we choose a radix-4 in-place scheme for the FFT we could use read-modify-write on our memory. That this is not trivial is given by the fact that a butterfly calculation needs several cycles, and thus the calculation of 2 butterflies need to be performed concurrently. An option is to store the results of a previous butterfly on the locations of the inputs of the current butterfly. In combination with a varying addressing scheme per stage this option became less interesting with respect to area.

Another way to obtain an in-place solution is to split the memory in 4 parts, i.e. the number of operands for a butterfly. This option is not very interesting as well because addressing becomes more complex. By splitting the memory area efficiency is reduced as well due to extra routing and address generation hardware.

The option we have chosen is a scheme found by Singleton [14]. This scheme uses a linear addressing, which is also identical over all stages. A disadvantage is however that more memory is required. Surprisingly this option delivered the most area efficient solution, therefore this scheme was chosen. The final schedule of the inner, folded, loop is shown in table 3. This schedule is already strongly simplified due to the use of a complex data path and the omission of control and addressing initialization details. The architecture is shown in figure 7.

Feedback from the synthesis tools Feedback from the compiler to the user includes measures such as cycle count, occupation degrees, and life times of variables. From the tools it is shown graphically but too detailed to be shown here. The occupation degree of the units shown in table 3 is generally above 75% for the complete algorithm.

User interaction Based on this feedback the user can take the appropriate action if he wants to steer the design space exploration. Two ways of interaction are possible.

First the user can add pragmas to the specification. This are implementation hints for the compiler. A rich pragma set is available in Mistral 2. It is not possible to discuss all pragmas within the scope of this paper. Most of the pragmas are related to data storage, data transport, data computation and to the controller. A first group is related to storage of variables in foreground register files or background RAMs. In both cases variables can be forced to share the same memory location. If desired, the user can control the memory organization by defining memory sectors and circular buffer structures. A second group is related to data transport. The user can decide which busses should be merged to minimize area. A third group is related to the computation units. Arbitrary computation units can be defined, including any arbitrary number of pipelines. In the case of FADIC this is used to define complex ALU and complex multiplier operations. Examples of controller pragmas are related to loop folding and loop unrolling.

A second possibility is to rewrite the DFL specification. For example, the decision to use a radix 4 butterfly, as discussed above, requires a rewriting of the specification. In general, different specifications that have the same behavior can have different implementations. Currently a lot of research effort is spent to become independent from such syntactical differences in the specification. This is interesting from an academic point of view but it is not realistic to expect results on a short term for complex designs such as FADIC. Therefore we accept that the way the specification is written influences the mapping result. Once accepted, it becomes a mechanism used to control the implementation. This is specially the case for the new procedural constructs in DFL. Typical examples are the assignment of variables to registers or the control of data routing.

5 Results

The synthesis finally led to the architecture shown in figure 7. The architecture makes use of complex additions and complex multiplications as basic processing units. These units are described in VHDL for register transfer level synthesis and imported in Mistral 2 as Application Specific Units (ASUs). The architecture consists of 2 instances of the complex adder, one complex multiplier, 3 data RAMs and 1 ROM with the twiddle factors. Most RAMs and ROMs have their own address calculation unit. A complete micro-coded control unit with program contents is produced as well by Mistral 2. Analysis of the micro-program gives a peak performance of 16 register transfers per clock-cycle, leading to a 196 MOPS machine.

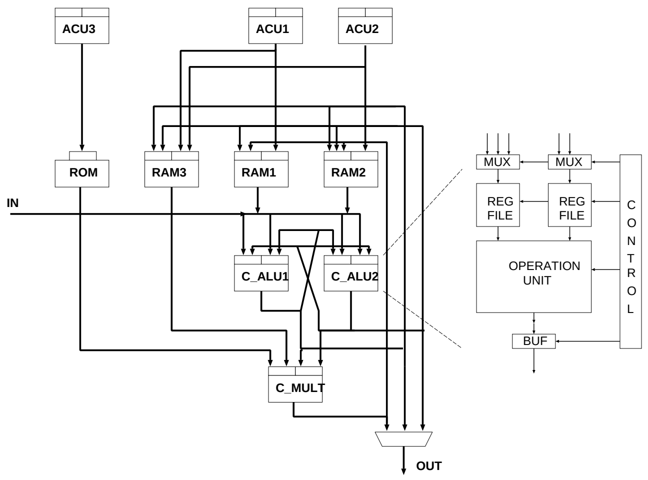

Figure 7: FADIC architecture generated with Mistral 2 (data path only).

After a number of iterations an acceptable solution was reached. At this point a complete VHDL description is generated. This result is simulated and verified against the simulation at the DFL level. When it is correct the register transfer level VHDL description is synthesized and mapped into standard cells except for the ROM, RAM, and register file modules for which parameterized module generators are available. The resulting layout is shown in figure 8. Table 4 summarizes the main results.

The first version of FADIC only supported part of the DAB system. Only a single operating mode was implemented, due to pressure on silicon delivery for field tests. While FADIC was applied in field tests, a second version [5] was made supporting all 3 DAB

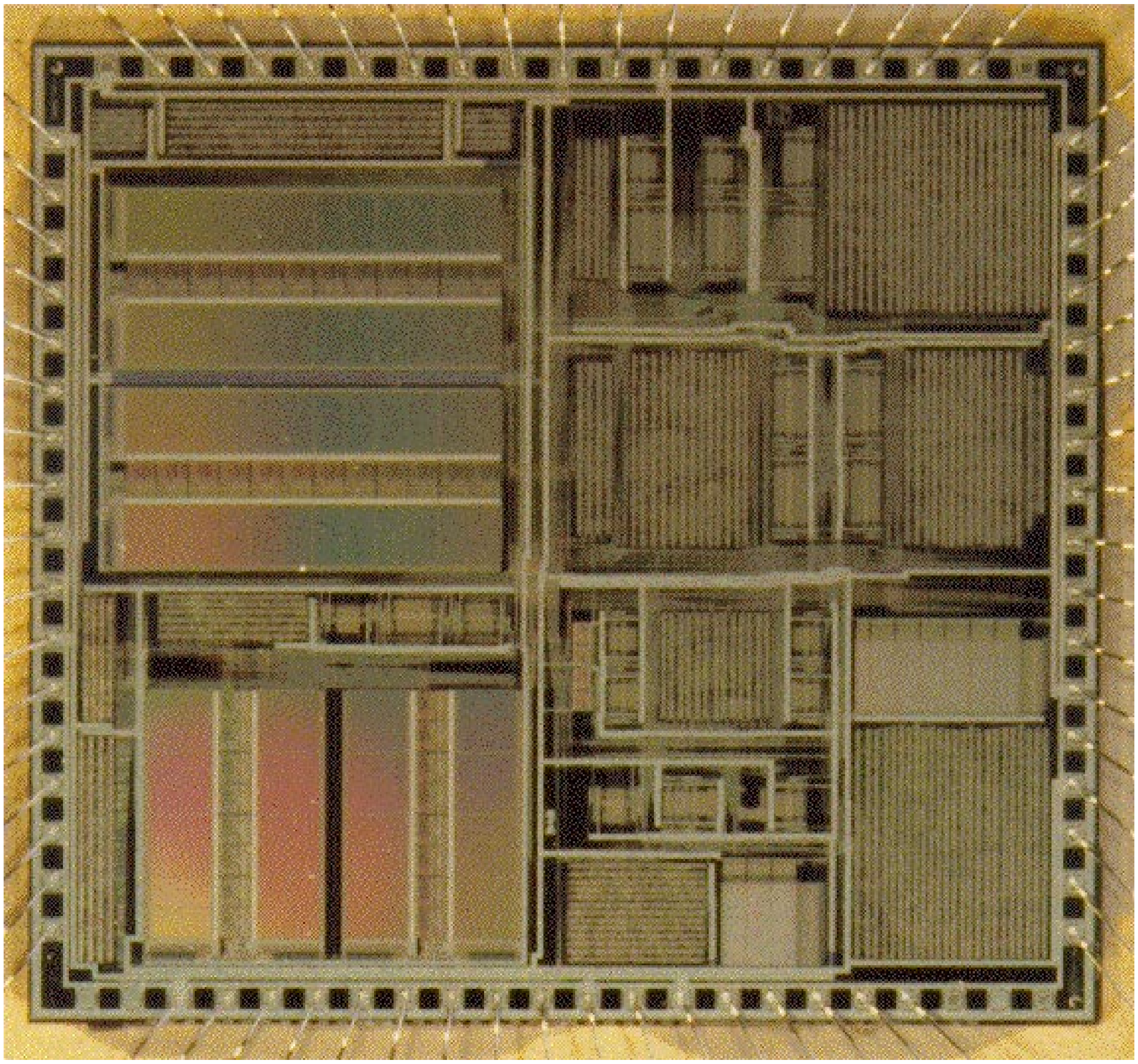

Figure 8: FADIC chip foto

	FADIC1	FADIC123
Functions	2K FFT + Diff.Demod.	256, 512, 2K FFT + DD, IFFT, AFC
Technology	1 µm	0.8 µm
# transistors	466700	519500
Area	111 mm^2	75 mm^2
# ROM	2	2
# RAM	3	3
# complex ALU	2	2
# complex MUL	1	1
# ACUs	5	4
program rom	85 by 183	330 by 160
program cycles	14899	1647, 3133, 14397

Table 4: FADIC high level synthesis results.

operating modes[2]. It was crucial to have HL-synthesis tools available since we were able to optimize our architecture even further and add more functions as well. The second design was finished within 4 months elapsed time, including layout optimizations and verification. The microprogram more than tripled in size and the IC technology changed to 0.8µm. Also built-in self test is applied for memories and full boundary-scan test is implemented. A shift from dedicated building blocks in a 1µm process (ACUs) toward standard cell implementations in 0.8µm was observed.

6 Conclusions

The use of high level synthesis tools for the design of a complex chip has been shown. Future versions of this design will follow. For example new architectures are currently explored which aim at a significant reduction of the power dissipation. Therefore a new exploration of the design space is necessary and this starts from the

[2]Philips device OQ8873

same DFL description. Also integration of extra signal processing functions is considered.

The following conclusions can be drawn. The design of a complex DSP chip such as FADIC is possible using the Mistral 2 high-level synthesis system. Short time to market is possible: typically 1 year (including specification time) for a first design and 4 months for a derivative IC. This can be done in combination with efficient implementations; typically 1 order of magnitude smaller than an implementation on commercial DSP processors. It was found that the success of such tools in the design community depends on the way user interaction is supported and stimulated. Fast and accurate feedback from the synthesis tools in combination with a rich set of hints for the compiler to guide the architecture exploration are the key issues.

Finally it is important to mention that the synthesis tool must be embedded in an environment including verification and test plan generation.

REFERENCES

[1] BRAYTON, R., CAMPOSANO, R., DEMICHELI, G., OTTEN, R., AND VAN EIJNDHOVEN, J. *Silicon Compilation.* Addison-Wesley, Reading, MA, 1988, ch. The Yorktown silicon compiler, pp. 204–311.

[2] DE MAN, H., CATTHOOR, F., GOOSSENS, G., VANHOOF, J., VAN MEERBERGEN, J., AND HUISKEN, J. Architecture-driven synthesis techniques for VLSI implementation of DSP algorithms. *Proceedings of the IEEE* (Feb. 1990), 319–335.

[3] DE MAN ET.AL., H. Cathedral II: a silicon compiler for digital signal processing. *IEEE Design and Test of Computers 3*, 6 (Dec. 1986), 13–25.

[4] DELARUELLE, A. The design of a syndrome generator chip using the piramid design system. In *Proc. European Solid-State Circuits Conf.* (Sept. 1988), pp. 256–259.

[5] DELARUELLE, A., HUISKEN, J., VAN LOON, J., AND WELTEN, F. A chip-set for a digital audio broadcasting channel decoder. In *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference* (345 E. 47 St. New York, NY 10017, May 1995), IEEE Electron Devices Society, IEEE Publishing Services, pp. 293 – 296. ISBN 0-7803-2584-2.

[6] EUROPEAN BROADCAST UNION. *Radio Broadcast Systems; Digital Audio Broadcasting to Mobile Portable and Fixed Receivers*, vol. Final Draft. European Transmission Standards Institute, 06921 Sophia Antipolis Cedex, France, Nov. 1994. prETS 300 401.

[7] HILFINGER, P. A high-level language and silicon compiler for digital signal processing. In *Proc. IEEE CICC* (May 1985), pp. 213–216.

[8] HUISKEN, J., VAN DE LAAR, F., DELARUELLE, A., AND PHILIPS, N. Specification, partitioning and design of a DAB channel decoder. In *VLSI Signal Processing* (October 1993), L. Eggermont, P. Dewilde, E. Deprettere, and J. van Meerbergen, Eds., vol. VI, IEEE Signal Processing Society, IEEE Special Publications, pp. 21 – 29. IEEE Workshop on VLSI Signal Processing, Veldhoven, The Netherlands.

[9] LANGEN, E. The Philips DAB 452 test receiver. In *DAB Newsletter*, F. Kozamernik, Ed., no. 6. European Broadcasting Union, autumn 1994, pp. 6–10.

[10] LE FLOCH, B., HALBERT-LASSALLE, R., AND CASTELAIN, D. Digital sound broadcasting to mobile receivers. *IEEE Transactions on Consumer Electronics 35*, 3 (August 1989).

[11] LOKHOFF, G. Precision adaptive subband coding for the Digital Compact Cassette. *IEEE Transaction on Consumer Electronics 38*, 4 (November 1991), 784 – 789.

[12] MCFARLAND, M., PARKER, A., AND CAMPOSANO, R. The high-level synthesis of digital systems. *Proceedings of the IEEE 78*, 2 (Feb. 1990), 301–318.

[13] SHUNG, C. An integrated CAD system for algorithm-specific IC design. *IEEE Trans. on CAD 10*, 4 (Apr. 1991), 447–482.

[14] SINGLETON, R. A method for computing the fast fourier transform with auxiliary memory and limited high-speed storage. *IEEE Transactions on Audio Electroacoustics AU-15* (June 1967).

[15] VAN DE LAAR, F., PHILIPS, N., AND OLDE DUBBELINK, R. General-purpose and application-specific design of a DAB channel decoder. *EBU Technical Review*, 258 (Winter 1993), 25 – 35. ISSN 1019-6587.