Silicon Compilation and Rapid Prototyping of Microprogrammed VLSI-Circuits with MIMOLA and SOLO 1400

N. Hendrich, J. Lohse*, R. Rauscher
University of Hamburg, Dept. of Computer Science
Troplowitzstr. 7, 2000 Hamburg 54, Germany
e-mail: hendrich@tech7.informatik.uni-hamburg.de (Internet)

Abstract
We describe our MIM2SOLO Silicon Compiler which integrates the MIMOLA high-level synthesis system and the SOLO 1400 standard-cell IC design system. Key features of the system are synthesis from an algorithmic description, full simulation support with automatic test pattern generation, and support of hybrid analogue/digital designs.
We present a CISC processor designed both traditionally and using MIM2SOLO and compare design efforts and layout. As an example of hybrid designs, we show a digital filter with on-chip AD and DA converters.

1 Introduction

With ever increasing demands on processing speed, data throughput, and reliability, software solutions are inadequate for several new application fields (e.g. digital signal processing, image processing, neural networks), and hardware solutions are required. Application specific integrated circuits (ASICs) offer the possibility to realize unusual architectures customized to meet performance requirements. More and more, the integrated circuits have to be designed as fast as possible and the *time to market* often is more crucial than the quality of the final layout.

Therefore the need arises for high-level synthesis to cut design-time and enable rapid prototyping. The design process starts with a system specification and ends with the chip layout and a set of test patterns. Only sophisticated tools give the designer the freedom to evaluate design decisions—that is, to carry out the process from behavioural description to chip layout several times.

In this paper we present a system overview of our MIM2SOLO silicon compiler which integrates both the design and simulation tools of the MIMOLA high-level synthesis system [10] and the standard-cell VLSI design software SOLO 1400 [4].

The MIM2SOLO system allows automatic high-level synthesis from an algorithmic specification will full access to the standard-cell libraries of SOLO 1400, including the analogue components. The mapping of RT-level generic operators to standard-cell structures is done with a novel method based on parametrizable cell libraries. Additionally, an EDIF netlist may be used to interface to other VLSI design systems.

*present address: IBM T.J.Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA

The vertical integration of the RT-level behavioural and structural simulations with the pre- and post-layout logic level simulations allows fast verification of the synthesis process and is indispensable for rapid prototyping.

Last but not least, the MIM2SOLO environment is very user-friendly. Both MIMOLA and SOLO 1400 offer interactive help facilities and on-line documentation, and can be put to good use by inexperienced designers too.

2 Related Work

High-level synthesis systems, e.g. [10], [2], [1], [12], as a rule, transform the algorithmic system specification into a RT-level structure with generic operators and the corresponding state transition tables. Early synthesis systems relied on manual transformation of this RT-level description into a system built from TTL and MSI components.

Most general silicon compilers use module-generators or datapath-compilers, combined with logic optimization, to generate reasonable dense layouts from the RT-level descriptions [1], [12]. However, the module-generators usually introduce a severe technology dependence and interface badly with industrial standard environments.

To ease acceptance of high-level synthesis tools in industrial applications, it is crucial that the tools can be integrated with existing VLSI design systems and their cell libraries. The Olympus system [8] uses module-generators to produce gate-level descriptions and its technology mapper to map to individual cell libraries.

An approach similar to ours to map generic RT-level operators to standard cell libraries using an VHDL environment [3] was proposed recently.

Very few existing silicon compilers support vertically integrated simulations from RT-level down to post-layout logic level.

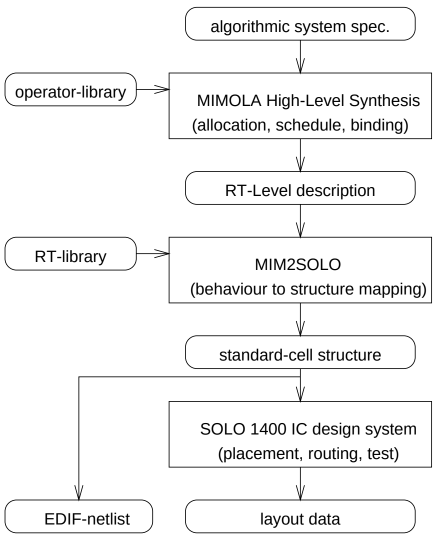

Figure 1: System architecture

3 System Overview

Figure 1 shows the system architecture of MIM2SOLO. The algorithmic-level specification of the desired behaviour is parsed by the MIMOLA frontend, then operations are scheduled, hardware modules allocated and bound. The output of the MIMOLA system is a structure on RT-level with the corresponding microcode.

The MIM2SOLO compiler parses the MIMOLA output and converts it to a standard-cell description where the mapping of generic RT-level operators to standard cells is done using a parametrizable library of RT-level constructs, called RTLIB. The MIMOLA microcode is optimized and a SOLO 1400 ROM-description is generated. Finally, the SOLO 1400 design tools are started. Floorplanning, placement and routing may execute fully automatically, and generate a chip layout (in standard CIF format).

Parallel to the synthesis, the behaviour specification and the synthesized design are simulated on the RT-level. These simulation results are converted into stimuli and expected values for the SOLO 1400 logic-level simulations. The results are checked to verify the synthesis.

To support analogue cells with MIMOLA, we extended the method used by ES2 in the SOLO 1400 system. Analogue wires are modelled as buses and analogue voltages are represented by digital values on this bus. As MIMOLA allows user-specified operator libraries, we were able to add behaviour descriptions of op-amps and converters to the standard MIMOLA operator libraries. However, the timing specifications are rather coarse. This allows to integrate and simulate analogue components within the high-level (algorithmic or structural) MIMOLA description.

The MIM2SOLO tools were realized using C with Lex/Yacc parsers with approximately 15.000 lines of code.

3.1 Overview of the MIMOLA synthesis system

The MIMOLA synthesis system [10], [11], [7] supports a high-level synthesis starting from an algorithmic behavioural description in the Pascal-like language MIMOLA (Machine Independent MicrOprogramming LAnguage).

The MIMOLA system synthesizes a hardware structure on the register-transfer-level with corresponding microcode, or generates a microcode for a given structure (retargetable microcode generator), alternatively. The use of user-defined operator libraries and replacement rules gives the designer considerable freedom in the synthesis and code-generation phase.

The system itself is partitioned into several individual programs. These are front-end compiler, behaviour and structure simulators, synthesis tools, microcode generation, automatic test pattern generation, and

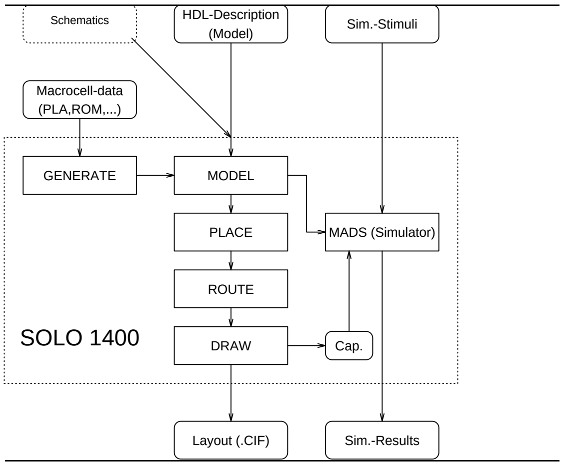

Figure 2: Design flow using the MIM2SOLO silicon compiler

schematic exit, etc.

An important option of the MIMOLA synthesis tools is to specify costs for each component of the operator library. However, while costs are quite easy to determine for SSI and MSI designs (e.g. TTL-components), they are unknown in VLSI designs because of wiring costs. Therefore MIM2SOLO offers an user-library with roughly estimated operator costs.

Output of the system (see also Fig. 1) is the register-transfer level structure and the horizontal microcode. See [7] for further details.

3.2 Overview of the SOLO 1400 system

The SOLO 1400 system [4] by European Silicon Structures Ltd. is a VLSI design system based on standard and macro cells. It allows the specification of a design using the proprietary hardware description language MODEL, without the necessity for schematic entry. MODEL supports a hierarchical hardware description of parametrizable *Parts* with C-like control structures.

This and the possibility of fully automatic operation of the design system make the SOLO 1400 system an ideal candidate for integration into a silicon compiler.

Currently SOLO 1400 uses the standard-cell libraries for ES2's 1.5μm and 1.2μm CMOS processes. Generic process-independent libraries built from the basic standard-cells offer a rich choice of MSI cells, e.g. all 74xx circuits. Macro-cell generators are used to produce RAMs, ROMs, PLAs, and parallel multipliers with full-custom layout quality. Other tools allow automatic synthesis of finite-state machines.

Hybrid (analogue and digital) designs are possible, using the cells from ES2's analogue libraries (several op-amps, analogue-digital and digital-analogue converters, reference voltage generators). To achieve analogue designs with a pure digital tool, analogue wires are modelled as 8-bit buses (giving a voltage resolution of about 0.02V) with corresponding simulation models for the analogue cells.

Usually, floorplanning, placement, and routing are done automatically. Naturally, some interactive modification of parameters is possible and recommended to obtain more compact layouts.

3.3 The MIM2SOLO-Compiler: Mapping MIMOLA to MODEL

The MIM2SOLO compiler implements most of the tasks that allow a fully automatic conversion of MIMOLA synthesis results to complete SOLO 1400 standard-cell structures.

It realizes the following functions:

- Parsing of the MIMOLA RT-level description into an internal graph-representation
- Compilation of multi-function MIMOLA modules (described via their RT-level behaviour) into matching operators from the RTLIB or into elementary operators with multiplexers
- Compilation of both the required multiplexers and the remaining elementary MIMOLA operators (e.g. "+", "INCR", "SHIFTLL", "SELECTBIT") into standard-cell realizations from the RTLIB
- Compilation of analogue operator modules into standard-cell realizations from ES2's analogue standard cells. The analogue wires are modelled by 8-bit buses as required by SOLO 1400
- Decision whether memories (e.g. register files) are to be realized on the chip or not
- Inclusion of pad cells for all primary inputs, outputs, and power supply at the top-level of the MODEL description

  If a memory was moved off-chip during the previous step, the necessary interface pads are allocated automatically
- Optimization of the microcode (don't care elimination) and conversion to the input format of the SOLO 1400 ROM generator

3.4 The RTLIB: Technology mapping

The RT-level structure from MIMOLA uses generic multiplexers (e.g. n bit, $m:1$) and complex operators, described via their behaviour (e.g. n bit addition, $n \cdot n$ bit multiplication). MIMOLA multi-function modules may combine several of the individual operators.

Such complex operators are not included in standard-cell libraries. Therefore, a step of mapping from behaviour to standard-cell structures is necessary.

To overcome this problem, we use a special library, called RTLIB, which contains parametrizable MODEL descriptions of all

`registers` (n bit):
  `latches`
  `DFF registers` (pos./neg. edge triggered)
  `scan-latches`
  `scan-DFFs`
  `shift-registers`
`memories` ($n \cdot m$ bit):
  `ROM` (macrocell)
  `RAM` (macrocell)
  `multiport-RAM` (macrocell)
  `stack, FIFO` (RAM macrocell)
`ALUs` (n bit):
  `adder, subtractor`,
  `incrementer, decrementer`
    (carry-look-ahead or ripple-carry)
  `comparison relations` (n bit)
    (`eq, neq, gt, gte, lt, lte, zero`)
  `multiplier` ($n \cdot m$ bit)
    (parallel (macrocell) or serial)
  `multi function ALUs` (e.g. 74181)
`logic:`
  `multi-level not,and,nand,or,nor`
  `PLA` (macrocell)
  `multiplexer` (n bit, $m : 1$)
    (as primitive muxes or buses)
  `enocders, decoders`
  `clock generator`
`interface:`
  `all ES2 Padcells`

Figure 3: RTLIB digital operators

the primitive operators and generic multiplexers up to a fairly large size. Additionally, multi-function operators (e.g. adder/incrementer, ..., 74181 ALU) are included in the RTLIB. A list of the digital operators currently supported by the RTLIB is shown in figure 3, the corresponding analogue operators are shown in figure 4.

Individual cells from the RTLIB are instanciated with their corresponding bit-widths by the hierarchical MODEL description produced by the MIM2SOLO compiler.

`supply:`
  `agnd, apwr` (analog ground, power)
  `anref, apref` (reference voltages)
`op-amps:`
  `opamp` (standard operation amplifier)
  `comp` (digital output comparator)
  `amux` (analog multiplexer)
  `dop` (digital controlled gain opamp)
`converters:`
  `dac` (digital-analog converter)
  `adc` (analog-digital converter)

Figure 4: RTLIB analog operators

For most of the operators the RTLIB offers several realizations. This gives the designer a choice to optimize for either performance or area. For example, carry-look-ahead or ripple-carry adders may be used to implement the MIMOLA "+" operator. Similarly, multiplexers may be implemented with primitive multiplexers or tri-state buses.

As an example we show in figure 5 the MIMOLA description of a 3-to-1 multiplexer (5 bit width), the corresponding MODEL description, and the RTLIB realization with primitive multiplexers. The alternative realization as a tri-state bus is not shown.

All RTLIB module output drivers are carefully dimensioned to balance sufficient driving strength with minimal chip area.

3.5 The MIM2WDL-Compiler: Multi-level simulation support

Though most silicon compilers claim *correctness by construction*, simulations *must* be done. Not only formal verification is not yet possible for software systems as complex as high-level synthesis tools. The main reason is the large number of bottom-up effects in VLSI design. For example, the chip delays and therefore the performance of the design cannot be estimated before place-

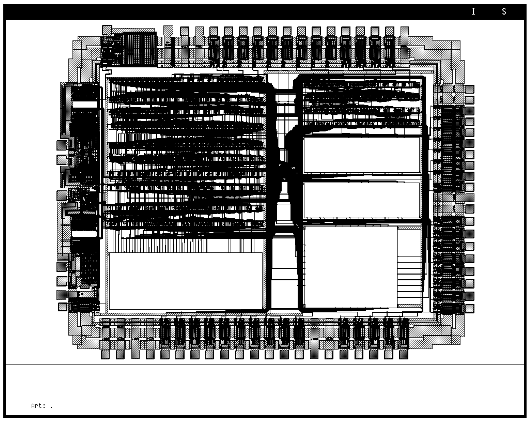

Figure 6: Layout of the hybrid digital filter circuit with on-chip ADC/DAC

ment and routing are done.

Therefore, simulations are necessary at all design-levels to verify the correct specification and implementation of the design. To be useful, a silicon compiler has to support simulations at all levels.

The MIMOLA synthesis system includes two simulators (MSSU and MSSS) which allow fast RT-level simulations of the specified algorithms on the synthesized structure. The interactive structure simulator MSSU is very easy to use and especially well suited for debugging. For example, all register contents can be modified and monitored at every time step while most logic simulators permit modification of primary inputs only.

The MIM2WDL compiler allows to convert the stimuli from the MIMOLA simulation runs to the format required by the SOLO 1400 simulators. Therefore, the stimuli used for the RT-level simulations can be reused for the logic-level simulations and the post-layout simulations to estimate chip delays. Comparison of the simulation results allows to verify the correct function of the synthesis tools.

4 Design Examples

The MIM2SOLO system has been tested on all standard MIMOLA examples, including the PDP8 processor and an industry standard 8-bit microcontroller (8051), and several of the usual benchmark circuits [13] for high-level synthesis.

In the following section we present our experiences with the design of two larger circuits. The first is a complex CISC processor and the second a digital filter circuit with on-chip analogue components.

4.1 A 16-bit CISC microprocessor

To test the MIM2SOLO system on a larger example we redesigned a 16-bit CISC microprocessor. The processor—described in detail in [9]—consists of a register file, 2 ALUs, a shifter, program counter/incrementer and the microcode ROM. This processor was previously manually designed using the SOLO 1400 software.

It took only three days to describe and to simulate the processor instruction set using MIMOLA. Additionally, we specified a behaviour-model for the two-port register

MIMOLA description:

```
MODULE mux03x05 (IN e1,e2,e3: (4:0);
          FCT sel: (1:0); OUT f:(4:0));
   CONBEGIN
    f <- CASE sel OF
       0: e1;
       1: e2;
       2: e3;
       3: X;
   CONEND;
```

MODEL description:

```
rtmux3 (5) [ e1(0:4), e2(0:4), e3(0:4),
            sel(0:1) ] -> f(0:4)
```

RTLIB realization (primitive multiplexer):

```
Part rtmux3 (n) [ i00(0:n-1), i01(0:n-1),
       i02(0:n-1), c(0:1) ] -> o(0:n-1)
{ simple mux realization }
Integer k
For k=0:n-1 Cycle
  es2mux21[ c(1), i02(k), es2mux21[ c(0),
            i01(k), i00(k)]] -> o(k)
Repeat
End
```

Figure 5: Example of 3-to-1 multiplexer

file and the status-register flags. Starting with this behavioural description (approx. 1000 lines of MIMOLA source), the high-level synthesis to register-transfer level with MIMOLA took 5 minutes. The conversion to standard cell structure with MIM2SOLO needed 1 minute, and layout with SOLO 1400 less than 1 hour of CPU time on a standard workstation (Apollo DN 4500). The resulting standard cell description consists of 800 lines of MODEL code (plus RTLIB and ROM code). Details of this procedure are given in [5].

The size of the die area of the synthesized processor is 78.0 mm^2 using ES2's 1.5μm CMOS process. It is shown in figure 7. The chip contains about 70.000 transistors. The microprogram ROM (splitted into two smaller ROMs) and the RAMs used to implement the register file can be seen. On the other hand, the area of the manually designed processor is 72.8 mm^2 (see figure 8). Both processors were simulated thoroughly both at RT-level and at logic-level. The simulations show that both designs nearly reach identical performance.

The comparison shows that the synthesized design is only about 7% larger than the hand design. However, note that placement and routing parameters of the hand design were optimized incrementally during weeks. Considering the upper area of the die, it seems possible to improve the layout of the synthesized processor by 5% as well.

It may be surprising that the synthesized version of the processor is competitive both in terms of performance and chip area. Even more, as only the behaviour and no initial structure was used for synthesis.

This, however, reflects the advantages of high-level synthesis and rapid prototyping. As a complete design cycle—with generation of complete and fault-free microcode—only takes some minutes, we were free to play with several design decisions and the corresponding alternative architectures. The decision to implement the register file using RAMs (which have full-custom quality) instead of the discrete latches used in the hand design was especially useful.

4.2 A hybrid digital filter circuit

To demonstrate the possibility of synthesis of an analogue/digital circuit with MIM2SOLO, we designed a digital FIR filter with on-chip analogue-digital and digital-analogue converters and loadable filter coefficients. To meet performance contraints a parallel multiplier had to be used.

As MIMOLA allows a Pascal-like description of the desired behaviour, it took only one day to specify the digital-filter algo-

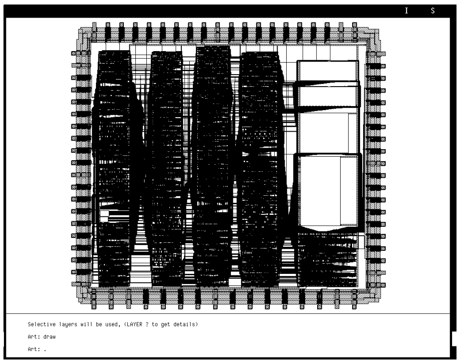

Figure 7: Layout of the automatically synthesized 16-bit processor. The right column of the core area contains (from above) the two RAMs to model the register file and the two microprogram ROMs.

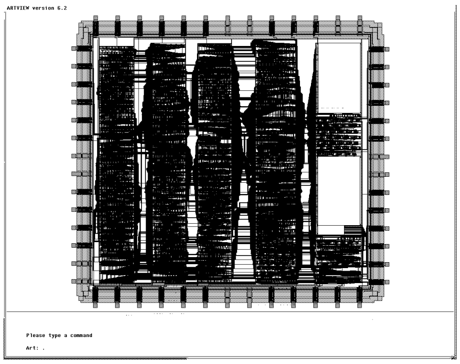

Figure 8: Layout of the manually designed 16-bit processor

rithm and to synthesize a first layout. Note that the analogue components are modelled using a special data type and an additional operator library in MIMOLA.

The final chip-layout—completed after only two days—is shown in figure 6. The chip measures 28.5 mm^2 using ES2's 1.5μm CMOS process. The left column in the core area contains the microprogram ROM and the right column, from above, the two RAMs for data-samples and filter-coefficients, and the parallel multiplier. The upper and left sides of the PAD ring contain the analogue components (supply, output buffer op-amp, DAC, ADC, supply).

5 Status and Future Directions

The 16-bit processor described above is currently going into fabrication at ES2. We plan to use MIM2SOLO for several future designs for rapid prototyping and to evaluate design decisions. Furthermore, we intend to synthesize the remaining of the standard high-level synthesis examples.

The system seems so easy to use that we will use it in students' projects.

6 Acknowledgements

The authors thank P. Marwedel and W. Schenk from the MIMOLA-group of the University of Dortmund for their support.

References

[1] R. K. Brayton, et al., The Yorktown Silicon Compiler System, in D. Gajski, *Silicon Compilation*, Addison Wesley, 1988

[2] R. Camposano, W. Rosenstiel, *Synthesizing Circuits from Behavioural Descriptions*, IEEE Transactions on CAD, 171–180, 1989

[3] N. D. Dutt, J. R. Kipps, *Bridging High-Level Synthesis to RTL technology libraries*, Proc. 28th Design Automation Conference, pp.526–529, San Francisco, 1991

[4] European Silicon Structures Ltd., *SOLO 1400 Release 3.1 Reference Manual*, Berkshire, 1991

[5] N. Hendrich, J. Lohse, R. Rauscher, *mim2solo: Automatischer Entwurf mikroprogrammierter VLSI Schaltungen mit MIMOLA und SOLO 1400*, Report 201/91, Univ. of Hamburg, Dept. of Computer Science, 1991

[6] N. Hendrich, *Silicon Compilation of Analogue/Digital Circuits with mim2solo*, Report, Univ. of Hamburg, Dept. of Computer Science, 1992, in preparation

[7] R. Jöhnk, P. Marwedel, *MIMOLA Reference Manual V. 3.45*, Report No. 8902, CAU Kiel, 1989

[8] D. Ku, G. De Micheli, *High Level Synthesis and Optimization Strategies in Hercules and Hebe*, Proc. European ASIC Conference, 124–129, Paris, 1990

[9] K. Lagemann, *Rechnerstrukturen*, Springer-Verlag, Berlin, 1987

[10] P. Marwedel, *The MIMOLA Design System: Tools for the Design of Digital Processors*, Proc. 21th Design Automation Conference, pp.587–593, Albuquerque, 1984

[11] P. Marwedel, *An Algorithm for the Synthesis of Processor Structures from Behavioural Specifications*, Proc. EUROMICRO 86, pp 251-262, Venice, Italy, 1986

[12] D. Thomas, et al., *Algorithmic and Register-Transfer Level Synthesis: The System Architect's Workbench*, Kluver Academic Publishers, 1990

[13] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0*, Microelectronics Center of North Carolina, USA, 1991