# A Transformational Approach to VHDL and CDFG Based High-Level Synthesis: a Case Study

Peter F.A. Middelhoek, Gerhard E. Mekenkamp, Bert E. Molenkamp and Thijs Krol

University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
e-mail: pfam@cs.utwente.nl

**Abstract**

In this paper, a novel multi-target design methodology based on the concepts of transformational design, and its application to the interlaced-to-progressive scan conversion (IPSC) problem, are discussed. Starting from a single high-level behavioral specification in VHDL a direction detector used in IPSC algorithms is mapped onto both a custom implementation and a programmable video signal processor. Results are compared with those previously obtained using different tools and methodologies.

## 1. Introduction

Currently, a wide variety of tools for high-level synthesis is available. All are specialized towards specific application and target domains. While this specialization was initially introduced to allow construction of efficient synthesis tools it is now more and more becoming a limitation. Because of changing requirements of a design in its life cycle, different synthesis tools and design flows are needed. Initially time-to-market or rapid prototyping are often the driving forces, which require the use of general-purpose reusable and programmable hardware. Later on in its life cycle other considerations become important. High-end features move down towards the domain of mass production where implementation cost becomes the dominant driving force. Other requirements, such as low power consumption, are also likely to appear. These different requirements, while maintaining the same functional behavior, call for completely different target architectures. Many examples of such retargeted designs can be found in both the computer as well as the consumer electronics market. This paper discusses how the transformational design methodology implemented in our design system TRADES (1) can be used to tackle these problems while at the same time offering a solution to both the correctness problem in high-level synthesis and the problem of interfacing different specification languages and tools.

## 2. Design Methodology

Most research in high-level synthesis concentrates on the synthesis of efficient implementations of well-defined behavioral specifications using a parameterized target architecture consisting of functional, memory, and interconnection units. This approach requires a large number of (incompatible) synthesis systems to cover the complete spectrum of digital designs. At the same time it is observed that many synthesis systems incorporate, in principle, the same kinds of synthesis steps. A flexible design methodology capable of generalizing and reusing these steps in different target specific design flows is desirable. Therefore our design methodology focuses on the synthesis of correct implementations using generalized, reusable design steps while at the same time being independent of a specific target architecture or specification language.

We believe that efficient designs can only be obtained by exploiting the flexibility and creativity of the designer. Therefore we have chosen to implement a user-centered design methodology which in addition should ease acceptance by designers. Within TRADES this implies that the selection of design steps (i.e. transformations) is done manually by the designer aided by a rule-based system. Obviously some well understood parts of design flows can and will be automated.

The objective of high-level synthesis is to both *optimize* and *refine* behavioral descriptions. Optimizations are used to reduce the implementation cost while refinement is used to reduce the abstraction level of a behavioral specification to allow direct implementation in hardware. The additional objective of formal design methods is to minimize the combined cost of both synthesis and verification. We have selected a transformation-based method which guarantees design correctness because all design steps are correct: i.e. the behavior of the design before and after the application of a transformation are equivalent. The reduced design speed due to the manual selection of design steps is partially compensated for by avoiding post-verification and debugging steps. Because all transformations are *generalized*, *primitive*, *small*, *local*, and *behavior-preserving*, reusability is maximized and the construction of correctness proofs simplified. Guaranteeing correctness requires a representation format with formal semantics to allow the construction of proofs for transformations. Currently within TRADES proofs are constructed manually (2). Results of using the automated proof system PVS from SRI have been reported in (3).

In practice different specification languages (e.g. VHDL, Silage, C) are used. If our methodology is to be applied to a wide variety of applications it must offer support for these languages. This requires the use of an intermediate language.

SFG languages have proven to be very useful as a backbone language in high-level synthesis (4). For the above reasons a data-dependency graph-like language which integrates both data and control flow (CDFG), SIL[1] (5), was developed. SIL has formal semantics (2) and includes support for the translation of functional, applicative, as well as imperative specification languages including implicit implementation suggestions (e.g., ordering in sequential languages). In (6) it was demonstrated how transformations on SIL can be used to transform a graph with a sequential implementation suggestion into a functional style graph. This makes SIL suitable as an intermediate language. The use of SIL as a language backbone for transformational design requires not only possession of the usual *abstraction* and *composition* properties but in addition both syntax as well as semantics should be easy to *manipulate*. Our experience shows

[1] SIL was developed in cooperation with Philips Research and IMEC as part of the ESPRIT/SPRITE project 2260.



0-7803-2584-2/95 $3.00 

that this requires a one-to-one mapping between syntactic and semantic components.

Other important considerations were the ability to integrate the methodology with existing synthesis systems; in particular the PHIDEO silicon compiler developed at Philips Research (7).

Fig. 1 shows two possible design flows of our multi-target design methodology that were used to map an edge direction detector for IPSC algorithms to both a programmable video signal processor, the VSP-2 (8) and custom hardware. High-level VHDL was used for the specification of the behavior of the algorithm. For the mapping onto the VSP-2 the methodology was used in combination with existing scheduling and allocation tools specifically designed for the VSP-1 & 2. For the custom implementation our work focused on the design of an efficient processing unit for the PHIDEO silicon compiler. In a previous implementation the direction detector accounted for 40% of the size of the data paths. Scheduling, allocation, mapping, generation of on-chip communication, memory, and addressing hardware were left to PHIDEO. Although our methodology can also be used for the above transformation steps, existing tools were used to show its ability to function in a hybrid environment, which is often a requirement in industrial settings. Disadvantage of this mixed approach is that correctness-by-construction can only be guaranteed in the section of the design path where transformations on SIL are used.

Besides abovementioned transformations such as scheduling we have defined and partially implemented within TRADES the usual transformations known from software compiler literature, such as for instance strength reduction (e.g. '*' to '+' and '<<'), and algebraic and loop transformations. Also defined are pipe–lining, retiming, and the similar time folding, as known from hardware design and in particular array synthesis. To deal with abstraction, transformations for the removal and introduction of hierarchy and transformations between data types are used

The application of hierarchy for abstraction, in contrast to its use as a hiding mechanism, requires a powerful mechanism that allows the manipulation of these hierarchies by means of transformations in the same way that primitive constructs can be manipulated.

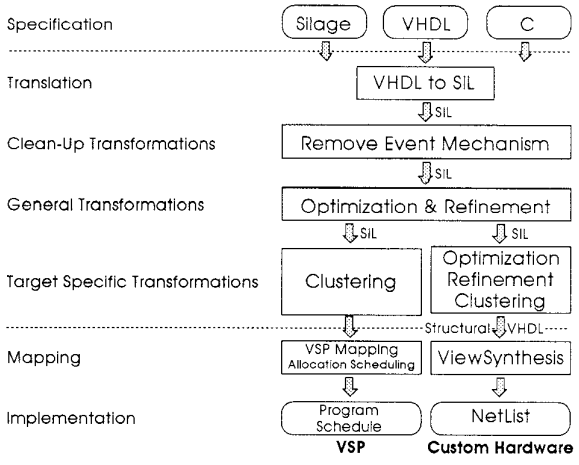


Fig. 1. Multi-target design methodology

## 3. Design of an IPSC Processor

As a test case for our transformational design methodology we have selected the design of the direction detection section of IPSC algorithms. IPSC algorithms are used to double the screen retrace frequency by interpolating intermediate scan lines of a field of an interlaced frame. If an edge is present in a field, interpolation takes place in the direction of the edge. Detection of edges is based on the gradient in the luminance. Three gradients are calculated based on the difference in luminance of three pairs of opposing pixels in the line before the estimated line and in the next. The smallest gradient indicates the direction of the edge. This algorithm in combination with linear interpolation of the luminance signal is known as the *edge-based line average* (ELA) algorithm. A diagram of the direction detector is shown in Fig. 2. A more detailed description can be found in (9). The direction detector was used in combination with an extended form of the ELA algorithm which first feeds the input signal through a low-pass filter and uses median interpolation instead of linear interpolation. This last modification requires the luminance of the pixel from the previous field with the same spatial location as the pixel to be estimated. Extensions to this filter have recently been proposed in (10), which also contains a more complete overview of the algorithm and its history.

We have selected the direction detector algorithm because it is relevant in the industry and currently used in television systems. Furthermore, the example has been studied extensively both at Philips Research (7) and IMEC (11) as part of the ESPRIT/SPRITE project. This allows us to compare our results both with respect to efficiency and flexibility with those obtained previously using different methodologies and tools.

The next sections describe the different phases in the design flow. First the VHDL to SIL translation and 'clean up' phase are discussed. This is followed by sections on target-independent optimizing transformations, and two target-specific back ends to the design flow.

### *3.1. VHDL to SIL*

Different approaches to the translation of VHDL to SIL have

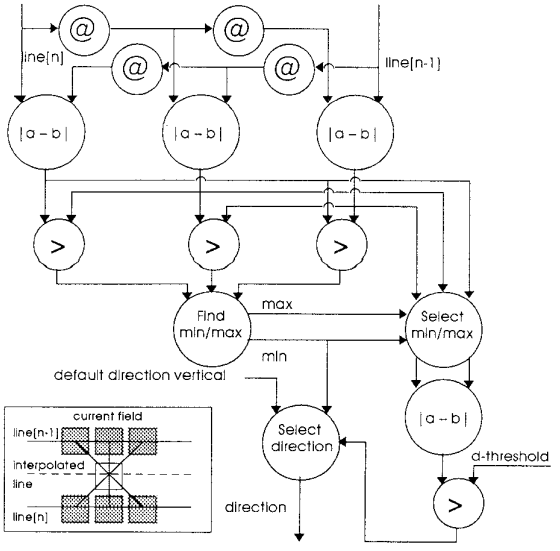


Fig. 2. Architecture of a direction detector for IPSC algorithms

been investigated. Within both Philips and the University of Twente experimental tools for the semantics-based translation of a subset of VHDL to SIL have been implemented. Such a semantics-based approach has, however, some significant drawbacks. The translation process is complicated because it requires the interpretation of the meaning of a VHDL description. Furthermore it is limited to a subset of VHDL, requiring complex synthesis guidelines. Within TRADES we are therefore developing an alternative syntax-based approach towards the translation of full VHDL (with the exception of the explicit time constructs AFTER and FOR). It requires only a relatively simple straightforward statement-by-statement translation. Afterwards the resulting SIL description is 'cleaned up' and optimized using the same behavior-preserving transformations as are used in the rest of the design flow. The main advantage is that the complexity of the translation process is now moved to the SIL domain where correctness can be guaranteed and transformations shared.

To evaluate our synthesis results with respect to flexibility and efficiency we derived two alternative custom implementations starting from VHDL. First, as a reference to the current state of synthesis tools, a direct implementation of the high-level description of the edge detection algorithm was synthesized with ViewSynthesis (VS) version 2 by ViewLogic. Next, an implementation was derived using VS starting from a lower-level, manually optimized structural VHDL specification developed at Philips Research, which is illustrated in Fig. 2. At IMEC this same structural specification, although specified in ELLA, was further optimized towards area and speed. The results obtained at IMEC are scaled towards the VS system by using the original Philips implementation as a reference. We refer to these latter two respectively as the 'structural' and the 'optimized structural' specifications. Results are compared to those obtained using TRADES and VS as a back end for low-level synthesis. We used simulation to verify the correctness of our results. Both implementations obtained using TRADES were indeed correct-by-construction even though TRADES is not yet fully implemented and verified.

As a starting point the most straightforward high-level VHDL specification of the direction detection algorithm was used. A first attempt to synthesize this description directly using VS failed because it violated the synthesis guidelines for VS. After some tuning of the original specification the VS tool accepted the specification. Note that every manual rewrite of the specification requires a validation step of the specification by means of simulation. Our VHDL-to-SIL compiler, however, without any problem accepted the initial high-level specification and generated useful SIL. This demonstrates the flexibility of the syntax-based translation method. The low-level structural VHDL description used by Philips proved to be no problem for VS. The efficiency of the three alternative paths is discussed in section 3.4.

For the mapping onto the VSP we compared our results with an implementation of the algorithm obtained previously at Philips. Results are discussed in section 3.5.

### *3.2. 'Clean Up' Transformations*

An obvious disadvantage of the syntax-based method is the large size of the SIL graphs that result after the translation. This is somewhat compensated for by the fact that the straightforwardness and transparency of the translation process preserve the structure of the original VHDL specification, which is especially important in a user-centered methodology.

Due to the syntax-based translation, the VHDL event mechanism appears in SIL and manifests itself as two nested loop levels. The inner loops model the continued execution of processes. The outer loop triggers computation if internal events occur. These loops can be removed by unfolding them, depending on the wait-statements followed by constant propagation, and dead code elimination. In addition hierarchy expansion and SIL-specific transformations are used. 'Clean up' transformations reduced the number of operations in the SIL description to 30% of the original.

### *3.3. General Optimizations*

Our primary design objective for the direction detector was the minimization of area. From Fig. 2 it will be clear that the top three comparators which can be implemented as subtractors can be combined with the bottommost absolute difference function. Further optimizations are possible by integrating the top three absolute difference functions with the comparators below and not calculating the *absolute* difference in all cases. As a side effect the size of the select min/max function can also be halved because only the correct max–min pair need be selected.

In general, area is reduced by minimizing the number of operations and implementation cost per operation. The former can be reduced by increasing the amount of sharing or reducing redundancy. If we decompose operations into more primitive operations the chances for sharing and redundancy reduction increase. For instance, the absolute difference functions can be decomposed into a subtraction and multiplexers. The subtraction can be further decomposed into an addition and a multiplication with –1. As it turns out we can use tail merging and common subexpression elimination in combination with many small transformations (like those based on the algebraic identity relation) to effectively halve the required implementation cost.

### *3.4. Custom Implementation*

The most important transformations are a combination of strength reduction and type transformations which allow the mapping of abstract operations onto lower-level operations which, after clustering, can be mapped efficiently onto hardware structures.

First, the results with respect to area in terms of gate count will be investigated. Table I gives an overview of the gate counts resulting from the four different design flows. The third row shows the maximum reduction in area obtained at IMEC. These resulted from using transformations based on *stuck-at* detection (i.e. dead code elimination and constant propagation) and applying them to the original structural specification. Other transformations that were tried, such as logic optimization using SIS, actually increased the size of the circuit, probably because they interfered with regularity in the data path. Their best re-

TABLE I
Gate count after different design trajectories

| | # of gates | % gates |
|---|---|---|
| VS | 2613 | 100 % |
| Structural | 1237 | 47 % |
| Optimized Structural | 1076 * | 41 % * |
| TRADES | 685 | 26 % |

* results are translated and scaled relative to structural implementation

sults have been translated to gate counts obtained by VS. The large size of the direct implementation using VS clearly indicates its limitations. More interesting is understanding the large difference between the direct implementation of the structural specification and the TRADES results and the relatively small gain obtained in the implementation of the optimized structural specification. The optimizations used for the optimized structural implementation were too low-level to result in a large area gain. The TRADES approach of optimizing the algorithm at the data path level is much more effective in this example.

To what extend should these improvements be attributed to the design methodology or to the designer? Although we can not quantify it, our belief is that the transparency of the design process and use of small design steps significantly improved understanding and hence quality of the design. Another important advantage is that our methodology allows the algorithm to be specified as it is most natural to the designer. Optimizing the algorithm results in a more complex description which would be difficult to notate correctly without the use of behavior-preserving transformations.

The implementation derived from the structural specification required 13 pipeline stages to reach the 27 MHz throughput requirement. (12) Claims that the optimized structural specification resulted in a speed gain of 25%. It does however not seem very likely that transformations based on stuck-at detection (i.e. constant signal detection) could result in any speed gain other than that obtained because of fan-out reduction. It seems more likely that the reported gain was due to the inability of the timing tools to calculate data dependent critical paths. That this is indeed the case has been confirmed by the authors of (11). Different optimizations used at IMEC resulted in a speed increase of up to 41% at the expense of area. Delay estimation using a carry-ripple model indicates a speed gain between 25% and 50% for the implementation obtained using TRADES.

If we look at power consumption the results obtained with TRADES seem even more favorable because the improvements in area and speed accumulate. The capacitance component in the power equation is half that of the implementation obtained from the structural specification. In addition the increase in speed of at least 25% results in fewer pipeline stages than the structural implementation, again reducing the capacitance component. Alternatively we could also maintain the same number of stages but reduce the supply voltage. Of course this would require the other sections of the design to be speeded up equally. This needs to be further investigated.

On the other hand it is expected that stuck-at based optimizations will result in a power reduction which is relatively less than the reduction in area. Also, power reductions as a result of the originally reported speed gains are unlikely.

### *3.5. VSP Implementation*

Type transformations are applied to scale all operations to the word width of the processor. Furthermore, SIL operations are clustered to make the behavior of the clusters correspond with that of processor instructions.

The mapping of the transformed direction detection algorithm onto the VSP shows a reduction in operations from 15 to 13. The main reason for this smaller gain, compared to the custom implementation, is that execution of control operations is relatively expensive on a processor. These can only be performed on word-wide signals and are just as expensive as operations in the data path, whereas the custom implementation can use efficient single-bit signals. Within the small algorithm this control aspect becomes dominant.

The correctness-by-construction property proved to be useful in discovering a small bug in the original implementation of the algorithm which had not yet been discovered by simulation. This illustrates the usefulness of our approach for obtaining correct implementations.

## Conclusions

The design of a very efficient direction detector used in IPSC algorithms from a high-level VHDL specification has demonstrated the feasibility of full-scale transformational design, including its 'first time right' property. The obtained results are due to the use of small behavior-preserving design steps. In addition we have shown the method's ability, through reuse of transformations, to efficiently integrate different design flows, as well as its ability to function in a hybrid multi-tool environment. We believe these capabilities, in combination with the interactiveness of the method, will make transformational design the methodology of choice for high-level synthesis.

Future work focuses on developing more efficient methods for the implementation and verification of transformations.

## Acknowledgments

The authors wish to thank their colleagues at the University of Twente, Philips Research (in particular Gerben Essink), and IMEC for their contributions. This research was funded by Philips Research.

## References

(1) P.F.A. Middelhoek, "Transformational design of digital signal processing applications", *Proc. of the ProRISC/IEEE Workshop on CSSP*, pp. 176-180, Arnhem, March 1994.*

(2) C. Huijs, Th. Krol, "A formal semantic model to fit SIL for transformational design", *Proc. of 20th Euromicro Conference*, pp. 100-107, Liverpool, September 1994.

(3) P.S. Rajan, "Transformations in high-level synthesis: formal specification and efficient mechanical verification", *technical report, SRI-CSL-94-10*, SRI International, Menlo Park, October 1994.*

(4) R. Camposano, R.M. Tabet, "Design representation for the synthesis of behavioral VHDL models", *Proc. 9th Int. Conf. on CHDL*, pp. 49-58, eds. J.A. Darringer and F.J. Ramming, Elsevier Science Publishers B.V., North Holland, June 1989.

(5) Th. Krol, J. van Meerbergen, C. Niessen, W. Smits, J. Huisken, "The SPRITE input language: an intermediate format for high-level synthesis", *Proc. of EDAC 92*, pp. 186-192, Brussels, March 1992.

(6) C.Huijs, J. Hofstede, T. Krol, "SIL: a useful interface between specifications and silicon compilers", *Proc. of the ProRISC/IEEE Workshop on CSSP*, pp. 99-104, Houthalen, April 1992.

(7) P.E.R. Lippens, J.L. van Meerbergen, A. van der Werf, W.F.J. Verhaegh, B.T. McSweeney, J.O. Huisken, O.P. McArdle, "PHIDEO: A silicon compiler for high speed algorithms", *Proc. of EDAC 91*, pp. 436-441, Amsterdam, February 1991.

(8) F.J. Sluijs, "VSP Programming Tools, User's Guide & Reference Manual", Philips Electronics N.V., 1993.

(9) P.F.A. Middelhoek, "Transformational design of a direction detector for the progressive scan conversion processor", *CS Memorandum 94-64*, University of Twente, October 1994.*

(10) M.H. Lee, J.H. Kim, J.S. Lee, K.K. Ryu, D.I. Song, "A new algorithm for interlaced to progressive scan conversion based on directional correlations and its IC design", *IEEE Trans. Consumer Electronics*, vol. 40, No. 2, pp. 119-125, May 1994.

(11) Z. Sahraoui, L. Rijnders, "Report on the DIRDET Experiments", *internal report IMEC*, 1992.

* available at http://www.spa.cs.utwente.nl/aid/trades/