

Evaluating the applicability of current VHDL synthesis tools to an industrial top-down development procedure

Lars Lindqvist

NKT Elektronik A/S,
NKT Alle 85,
DK-2605 Brøndby, Denmark

and

Center of Integrated Electronics,
Technical University of Denmark,
Building 345, DK-2800 Lyngby, Denmark

E-mail: llq@ne.dk

E-mail: llq@id.dth.dk

Abstract

This paper reports on an evaluation of commercial synthesis systems from Racal-Redac, Mentor Graphics and Synopsys. The tools have been evaluated with a circuit example from a previous, commercial ASIC design as reference. The conclusion is that automated synthesis from RTL VHDL to a final design is possible and feasible. Furthermore, the synthesized designs are of a quality comparable to what is seen in a gate-level design from a commercial project, and the design time is much shorter.

1 Introduction

Time to market is a key parameter for staying competitive, and thus a very important measure for products developed in the industry. Also, driven by market factors is the ever increasing complexity of electronics systems. Therefore, development procedures and tools, which would reduce the time to market and ease the handling of complex designs, are very important research areas. The work [1] on which this paper is based is part of a research project in top-down development procedures for use in the industry. The objective of this part of the project was to establish which tools already existed and how they performed. This paper reports on the performance of a set of commercially available synthesis tools.

Previous work related to this has been reported in [2]. Lim et al. concluded, that automatic HDL based synthesis, with the present synthesis technology was not possible. Specifically, they reported that they had to make many iterations in the design process and that

it was necessary to detail the design constraints several times. It was also necessary to hand-edit some parts of the design.

In [3] Gajski and Dutt report on the difficulties of evaluating synthesis tools. Some of the conclusions were that realistic designs with related test-benches are needed to evaluate synthesis tools.

The evaluation presented in this paper differs from other benchmarks in the following aspects: Firstly, it evaluates a number of commercially available synthesis tools with emphasis on the applicability to an industrial top-down development procedure. This is in contrast with the normal comparison with only measurable criteria such as pure gate count and no restrictions on the time and effort needed to become productive. Secondly, the evaluation was done with a circuit example from a previous ASIC design. This ensures that the circuit example is well understood as regards functionality as well as performance criteria. Furthermore, the existing design, which was made with a traditional schematic-entry-based development procedure, serves as a reference for the synthesized designs. Thirdly, the designs were targeted towards a real technology. That is, the synthesized designs were completed to a degree where the netlists were ready for design input to the ASIC vendor.

2 Applicability criteria

The synthesis tools are intended to be used in a top-down development procedure using VHDL as specification language (see figure 1). The procedure implies the use of an executable specification written in VHDL at an early stage

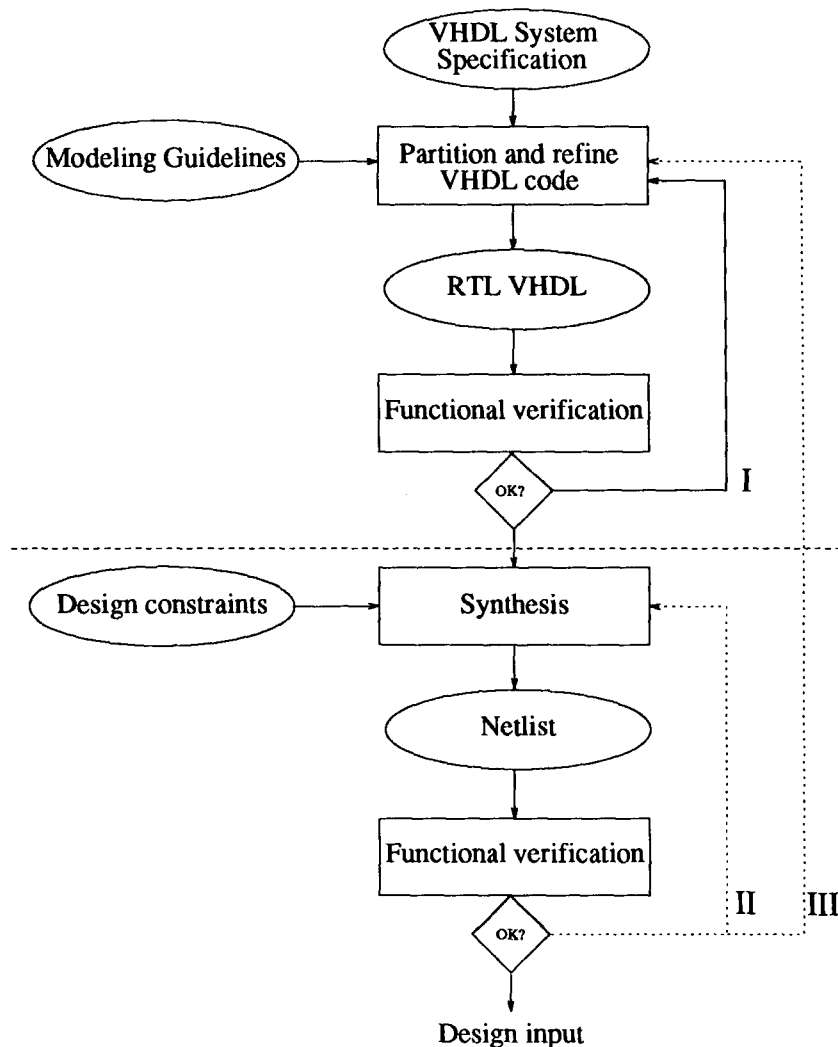


Figure 1: The use of synthesis in the design flow.

of the development process. After verifying the system specification, it is refined until a point where the synthesis tools can take over. The level of description where this takes place is in the proximity of the RT-level. Assuming a given time frame for the development, it is the goal to be able to spend as large a fraction of the time frame as possible above the dashed line depicted in figure 1, i.e. making the VHDL system specification and iterating in loop I.

Important questions are which VHDL subsets are supported by the tools, and how efficient designs they will create from the supported VHDL subset. The support of the synthesizable VHDL subset requires a detailed study of the VHDL modeling guidelines. These guidelines tell how to write efficient VHDL code targeted at a particular tool. Establishing the

degree of weirdness of these modeling guidelines, as seen from a designer's viewpoint, is important to an evaluation with focus on development procedures.

Another topic of the evaluation is the route from VHDL code, describing the RTL design using the appropriate modeling guidelines, to the final design, i.e. the steps below the dashed line in figure 1. In this area, aspects such as the maturity and availability of the tools should be considered. Support of the tools is also important, for example libraries must be available in order to produce any designs.

When the tool works it is important that the design constraints could be fulfilled, i.e. the constraints could be specified and obeyed. Telecommunications circuits, such as the one selected in this benchmark, are normally con-

strained by timing requirements. Other problems are for example handling of divided clocks. Before a design is usable it has to be fabricated, which implies that the selected test methodology has to be supported by the tool. Further, the design also has to be tolerant towards changes in environmental conditions such as temperature and supply voltage.

3 Circuit example

A single circuit has been selected as benchmark for evaluation of all the tools. The circuit example is taken from the SDH¹ and SONET² domain since this is the application domain of interest to my research in top-down development. The reason for selecting only one circuit is the desire to use as large a circuit example as possible, given the time frame for the evaluation.

SDH and SONET are the new emerging standards for high speed telecommunications using optical fibres. A thorough discussion of SDH and SONET is given in [4]. The SDH terminology will be used in the following.

The information about SDH needed to understand this benchmark circuit is given in the following. High speed optical telecommunications systems are used for transporting data in the network operator's trunk networks. The bits transported in optical fibres are conceptually divided into frames. The basic structure in SDH is the STM-1 frame as shown in figure 2. It contains 9 rows each containing 270 columns. Each position in the STM-1 frame contains one byte (8 bits). The bits are sent in the fibre row by row from top to bottom and column by column from left to right. The most significant bit of a byte is sent first. The frame is divided into three regions the RSOH³, MSOH⁴, and AUG⁵. Simplifying the network a little, one could say that payload data are transported from multiplexer to multiplexer in the AUG. The multiplexers at each end of the multiplexer section use the MSOH for monitoring and communication. If the physical distance between the multiplexers are too long, regenerators are inserted on the line. These subsections of the line are called regenerator sections. The regenerator sections use the RSOH for monitoring and communication. The RSOH consists of 27 bytes and the MSOH consists of 45 bytes.

The benchmark circuit is a frame aligner

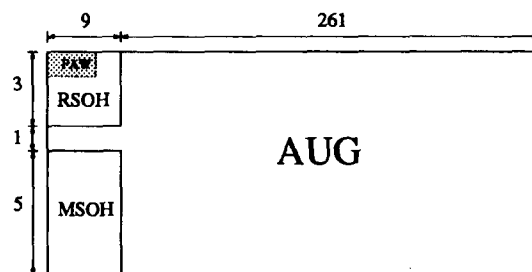


Figure 2: Frame structure for STM-1

for STM-1. Besides the information about the frame structure, the benchmark circuit uses the first 6 bytes in the RSOH. These bytes are allocated to a frame alignment word (FAW). The FAW is a byte sequence which is used to find the frame structure in a received bit stream. A frame alignment is needed because the STM-1 frame is transported without explicit position information, i.e. only the data are transported not the row and column values. The position information is needed to process the RSOH, MSOH, and AUG correctly.

The STM-1 is transported at a rate of 155.520 Mbits/sec which transforms to 125 μ sec per frame. The circuit used in this benchmark performs frame alignment of STM-1 input (see figure 3). The input is an STM-1 bit stream demultiplexed to 4-bit-width. The corresponding clock with a frequency of 38.88 MHz is also supplied to the circuit. The output of the circuit is a data signal 8 bits wide containing the individual bytes in the STM-1 frame. The input is scanned for a match of the FAW sequence. When the sequence is recognized, it is assumed that it is a FAW. In fact it could have been an emulated FAW in the payload data but this is not considered in this circuit. The clock output corresponds to the data output. The temporal relation between data output and clock output must remain the same, i.e. realignment might require gapping the output clock. Row and column values are also output from the circuit. An external signal controls whether the circuit is allowed to realign or not.

4 Target technology

The Fujitsu CG21 series has been selected as target technology. Fujitsu CG21 is a 0.8 μ m CMOS sea-of-gates family with up to 100k gates masters. Typical delays are 370ps/gate.

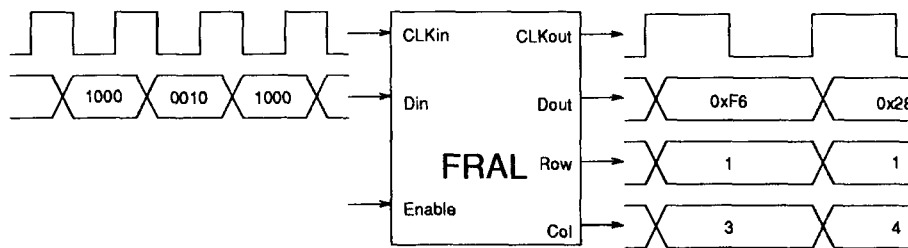


Figure 3: Frame alignment circuit for STM-1.

The test methodology used in the hand-crafted design is a Fujitsu specific FullScan design style. It is somewhat like an auxiliary clock LSSD test methodology. Basically, the requirement to the designer is that scan flip-flops must be used instead of normal flip-flops, and that three extra signals have to be added. There are some additional rules, but only the requirements for divided clocks influence this benchmark. It is required that clock inputs on all flip-flops clocked by the output of another flip-flop can be disabled via an external pin. To operate in the application environment the circuit should be designed to handle changes in temperature and supply voltage. Further the circuit should be designed to take care of variations in the production process. This implies the requirement that delay multipliers in the range from 0.35 to 1.85 should not affect the functionality of the circuit. A delay multiplier is a scaling factor used to reflect variations in the timing caused by different environmental conditions and production variations. The actual delays are found by multiplying the nominal delays with the delay multiplier.

The VHDL code for what is almost equivalent to the hand-crafted design is 470 lines of code including both behavioral and RTL descriptions and a test bench for reading a stimulus file and writing a response file. The VHDL code was written without targeting any specific tool, i.e. the code was written in the context of [5, 6] and the expectation that RTL is the description level from where synthesis could be done. The VHDL code was written in 2 days. The hand-crafted circuit was designed in 2 weeks as part of an ASIC.

5 Synthesis tools

The following sections describe the evaluated tools. The main focus of the evaluation was

the synthesis tools. Therefore, no performance data are given for the simulators. The mentioning of the simulators serves to indicate that a complete design environment existed.

The synthesis was performed as follows: First the behavioral VHDL code and the RTL VHDL code were simulated. Then the RTL VHDL code was synthesized. All tools required changes to the RTL VHDL code, so some iterations were done before the code was accepted by the tools. The modified code was then synthesized, and based on the result some iterations were done with the synthesis scripts for the tools. The completed designs were transformed into an FLDL⁶ netlist. LDRC⁷ was used to check the designs for design rule violations. Due to unavailability of the SDRC⁸ software the designs have not been passed by SDRC.

With a Fujitsu tool, called *shmaker*, Mentor version 7 schematics were created. The evaluation data were then obtained by simulating and inspecting the designs using Mentor version 7 tools. Therefore, all the designs were analyzed using the same CG21 simulation library as the hand-crafted design. Instead of running SDRC, the schematics have been inspected.

The results of the evaluations are summarized in figure 4. The following subsections present the information condensed in the table. Please note at this point that an absolute performance comparison of the tools would hardly be precise using only this benchmark. The benchmark is taken from a specific application domain that might favour one tool to another. Also note that the data in the table describe the situation at the time of the evaluation. More recent releases of the tools might offer a better solution for some of the items in the table.

5.1 Racal-Redac

The Racal-Redac (RR) tools were evaluated with the aid of an applications engineer. The VHDL code was sent to the applications engineer on May 17, 1992. On July 6 and 7 the RR tools were demonstrated at RR's domicile in Reading, UK. On that occasion the tools would only synthesize FullScan design for a single block. This problem meant that only a design without scan was synthesized for the entire FRAL circuit. RR and Fujitsu promised to look into the problem. A new version of the FRAL design was received on August 24, 1992.

At the time of the evaluation the tools as a unity ran only on SUN Sparc. The version of the synthesis tool demonstrated was SilcSyn 2.3. It ran on a SUN Sparc station 2 with 32 MB RAM, an 1.3 GB disk and 250 MB swap space.

The VHDL subset supported by SilcSyn requires that attributes are included for all clock and reset signals. SilcSyn does not permit signal declarations in packages. This kind of signal declaration may be useful for clock and reset signals. The limitation that no ports can be assigned in clocked processes is annoying.

The VHDL modeling guidelines for SilcSyn state that the best synthesis results are obtained when combining logic and registers in the same block. Although it need not be in the same processes, it would have been preferable with a flattening command instead. The guidelines recommend the usage of the `std_logic` bit type, which includes an unknown state.

The application engineer, demonstrating the tools, changed the design so that all flip-flops were clocked by *CLKin*. This removes the problem of synthesizing divided clocks, but reduces the performance of the circuit. After validating the VHDL source code with the simulator, SilcSyn is used to synthesize a netlist from the VHDL code. SilcSyn operates in three steps:

- Analyze
- Link
- Build

In the analyze step the VHDL code is parsed via VTIP⁹. VTIP finds syntax errors or in case of no errors it prepares a synthesis view of the code. Synthesis view is a special database for representing VHDL code in synthesis tools.

In the link step SilcSyn maps the synthesis view to a generic library. This library consists of n-bit registers, adders, logic, etc. After the link step, it is possible to simulate the generic design. This simulation can be used to find initialization problems in the design. These problems may pass the initial VHDL simulation because of the way VHDL operates. A variable or signal in VHDL may be initialized in its declaration. In real hardware a reset has to be used to get the circuit into a defined state. The generic design is still technology independent.

The third step is to map the generic netlist from the link step to the target library. In the build step signals and variables are encoded and the combinational logic is optimized. As input to the builder, an implementation file is used to define constraints on the synthesis. The builder performs timing analysis of the design to verify the fulfilment of the constraints. If some of the constraints are not fulfilled, the builder tries some more efficacious and time consuming synthesis procedures. If this is not enough, an error report list is generated. The synthesis took about 15 minutes from VHDL code to a Fujitsu netlist. The synthesized design uses 1227 gate-equivalents. This count could have been reduced by 110 using a builder constraint to avoid block clock buffers and using normal clock buffers instead. The gate count to compare with other solutions is thus 1117 gate-equivalents. The minimum clock period for the entire FRAL circuit was 16.90 ns. This corresponds to a frequency of 59.2 MHz and a delay multiplier of 1.52 over the required frequency of 38.88 MHz. This is too small a margin, since 1.85 is required. That is, the synthesis has to be redone or the RTL VHDL has to be changed.

The design contains 13 slightly overloaded nets¹⁰. These overloads are permitted and are merely displayed as warnings about nets that might be severely overloaded after place and route.

5.2 Mentor

The evaluation of the Mentor Graphics software was performed at NKT during August and September 1992. A specialist assisted in the evaluation of AutoLogic, which is the Mentor synthesis tool.

The software is available on HP/Apollo and Sun. The evaluation was performed on an HP 9000-425 with 32 MB RAM. The version of the

software used was 8.1.

The synthesis tool consists of two products. AutoLogic VHDL and AutoLogic. AutoLogic VHDL synthesizes a generic netlist from the VHDL source code. This step corresponds to the analyze and link steps of SilcSyn. AutoLogic performs the optimizations and mapping to the target library. This corresponds to the build step of SilcSyn. The VHDL subset supported by AutoLogic VHDL does not permit global signal declarations in packages. The handling of VHDL default binding was erroneous. An important limitation is that integers are not allowed in ports.

The VHDL modeling guidelines recommend the usage of `qsim_state` as bit type. This is a Mentor specific type, but the package with the declarations is available to the user.

AutoLogic VHDL maps the VHDL source to the generic netlist without any script files provided by the user. AutoLogic, on the other hand, needs a script in order to work. The script determines which optimization algorithms should be used and in what order. Making this script requires some knowledge about logic optimization and experience with the tool definitely improves the synthesized designs.

The entire synthesis from VHDL to end of AutoLogic took a couple of hours. The design is a Fujitsu CG21 FullScan design. It uses 989 gate-equivalents of which 500 gate-equivalents are 50 single bit flip-flops. LDRC only reported that one net was slightly overloaded. Only one SDRC violation immediately showed by the visual inspection. The clock signal *CLKout* comes from another flip-flop and therefore must be gated with XTST. This gating is missing in the design.

The minimum clock period with which the design simulated correctly was 14.20 ns. This corresponds to a maximum delay multiplier of 1.81 over the required frequency of 38.88 MHz. One has to note that hold time errors were reported during the simulations. They were caused by insufficient delay corresponding to signals coming from the *CLKin* domain and ending in the *CLKout* domain. The design is thus very close to fulfilling the design constraints.

5.3 Synopsys

The evaluation of the Synopsys software packages was done at the Technical University

of Denmark in the period from November 30, 1992 to January 14, 1993.

The Synopsys tools were all version 3.0-beta. The tools were installed on an Sun Sparcstation 1+ with 56 MB RAM and 128 MB swap space. The Fujitsu support tools are at present targeted at version 2.x of the Synopsys tools but for the purpose of this evaluation only minor problems arose on that account. It is worth noting that even though it was a beta release, no crashes occurred during the evaluation. The synthesis tool Design Analyzer consists of several tools. In fact, it encapsulates all the Synopsys tools available for synthesis, which could also include test synthesis and other HDL interfaces. The evaluated suite of tools were VHDL Compiler and Design Compiler. The VHDL subset supported by VHDL Compiler allows signal declarations in packages, but the signals are not global. This means that each entity using the signal declared in a package will cause a new signal.

The VHDL modeling guidelines specify how to place assignments to avoid inferring unnecessary registers in the design. This guideline required rewriting of the RTL code in much the same way as SilcSyn required. Reading the VHDL source generates a generic design. After loading the target library design constraints are specified. For the FRAL design three commands were needed to specify environment conditions, wireload model and clock frequency. The constrained design is then compiled which includes resource allocation, mapping, and optimization. After the compilation various reports could be generated. Based on these, more constraints could be specified for the design and a new iteration done.

The FRAL design was synthesized in 30 minutes. The synthesized design is a HiScan design, which is another Fujitsu test methodology, and was manually converted to a FullScan design. The conversion was done by replacing all HiScan flip-flops with their FullScan equivalent. After the conversion the design used 1021 gate-equivalents. LDRC reported that no nets are overloaded. The converted design respects the scan design rules except for the divided clock.

Inspection of the design shows that block clock buffers are used in the design. One of the buffers is unconnected, which might be caused by the HiScan design style, but since the design is converted to FullScan this should not be included in the gate count. Reduction of the

gate count according to this assumption results in the design using 998 gate-equivalents.

The minimum clock period at which the design simulates without any errors was 12.80 ns. This corresponds to a delay multiplier of 2.01 over the required frequency of 38.88 MHz. This means that the design constraints are fulfilled.

6 Conclusions

This paper shows that it is indeed possible to synthesize from RTL VHDL to a design ready for design input. It is shown that tools exist which are capable of producing designs of a quality comparable to what is actually seen in commercial projects using gate-level design. The conclusion is that synthesis from RTL VHDL to a final design could be done with very few iterations in the synthesis and without requiring hand editing of the design to fulfill performance constraints. The synthesis time is short enough to allow effectively exploration of different architecture possibilities at the RTL level. It is also possible for the designer to become productive using reasonable time and effort. One tool was installed from scratch and 3 man-weeks later the final design was ready. Therefore, I consider the tools applicable to a top-down development procedure used for making commercial designs. The important lessons learned during the evaluation are that the major differences between the tools are in user interface and VHDL modeling guidelines.

Notes

- 1 Synchronous Digital Hierarchy, SDH is defined by ITU-T (former CCITT) recommendations.
- 2 Synchronous Optical NETWORK, SONET is defined by ANSI standards.
- 3 Regenerator Section OverHead.
- 4 Multiplexer Section OverHead.
- 5 Administrative Unit Group.
- 6 Fujitsu Logic Description Language.
- 7 Logic Design Rule Checker.
- 8 Scan Design Rule Checker.

- 9 VTIP is a VHDL parser from CLSI solutions.
- 10 Overloads marked with ** in the LDRC output. This kind of overloading is permitted but leads to slower operation.
- 11 Guidelines explicitly needed in this benchmark.
- 12 The sizes referred to are for FullScan designs comparable with the hand-crafted circuit.
- 13 The design contains heavily loaded nets, i.e. they are not overloaded but might be after layout.

References

- [1] Lars Lindqvist. VHDL synthesis benchmark. Technical Report DAG93-TR01, CIE, The Technical University of Denmark, 1993.
- [2] Stephen E. Lim, David C. Hendry, and Ping F Yeung. Experiences and issues in VHDL-based synthesis. In *Proceedings of EuroDAC'92 with EuroVHDL'92*, Hamburg 1992.
- [3] Daniel D. Gajski and Nikil D. Dutt. Benchmarking and the art of synthesis tool comparison. In *IFIP Workshop on Control Dominated Synthesis from a Register Transfer Level Description*, Grenoble 1992.
- [4] Mike Sexton and Andy Reid. *Transmission Networking: SONET and the Synchronous Digital Hierarchy*. Artech House, 1992.
- [5] IEEE. *IEEE Standard VHDL Language Reference Manual*, 1987 edition.
- [6] Roger Lipsett, Carl Schaefer, and Cary Ussery. *VHDL: Hardware Description and Design*. Kluwer Academic Publishers, 1989.

Item	Racal-Redac	Mentor Graphics	Synopsys	Human
VHDL simulator	VHDL2000	QuickSimII	VSS	
Synthesis	SilcSyn	AutoLogicVHDL, AutoLogic	VHDLCompiler, DesignCompiler	
Synthesis tool version	2.3	8.1-2	3.0-beta	
Gate simulator	CADAT	QuickSimII	VSS	QuickSim
Platforms	SUN	HP/Apollo, SUN	SUN	HP/Apollo
Evaluation period	Jul-Aug 92	Aug-Sep 92	Dec 92-Jan 93	
Evaluated on	SPARC station 2	HP9000-425	SPARC station 1+	
Notes on VHDL subset	no global signals, no assignments of ports in clocked processes, attributes for clock and reset signals	no global signals, no integers in ports	global signals not connected	
VHDL guidelines ¹¹	logic and register in same block, ranges important	ranges important	use assignments in clocked processes carefully, ranges important	
Recommended bit type	std_logic_1164	qsim_state		
Synthesis of divided clocks	?	yes	yes	
User interface speed	fast	slow	ok	
User interface quality	ok	acceptable	ok	
Documentation	ok	overwhelming	ok	
Learning		tedious	easy	
Design database	open	closed	open	
Time VHDL to design	15 min	2.5 hour	30 min	
Size of design ¹²	1117 BC	989 BC	998 BC	920 BC
Max delay multiplier	1.52	1.81	2.01	>1.85
Design rules	ok ¹³	ok ¹³	ok	ok
Test methodology	FullScan	FullScan	HiScan	FullScan
Design quality	too slow	hold time errors	good	it works!

Figure 4: Evaluation results summary.