Case Study on DiaHDL: A Web-based Electronic Design Automation Tool for Education Purpose

Muhammad Shoaib Iqbal Ansari, Thomas Schumann
Faculty of Electrical Engineering
h_da – University of Applied Sciences Darmstadt, Germany
iqbal.ing@gmail.com
schumann@eit.h-da.de

Abstract: This paper describes a case study on DiaHDL, a web-based tool we have developed. The tool generates a VHDL code of a digital circuit and its VHDL test bench code. For this case study we show the design flow of a 32-bit adder from component diagram to VHDL code and its test bench.

1 Introduction

Today Field Programmable Gate Arrays (FPGAs) are replacing Application Specific Integrated Circuits (ASICs) in many application areas, so the development of electronic circuits on FPGAs is an important topic for the education in electronic engineering. Many university labs offer FPGA-boards for the students to finally implement their developed circuits. But the development of those circuits requires the knowledge of a Hardware Description Language (HDL) such as Verilog or VHDL. We have developed the web-based tool DiaHDL [CS10], which not only generates the VHDL code out of component diagram of the circuit but also its test bench. In this paper we present a case study on this tool. We show that the generated VHDL code of a 32-bit adder circuit is functional and synthesizable.

2 Related work

2.1 Simulink HDL Coder

Simulink HDL Coder™ is a MATLAB toolbox extension from The MathWorks, Inc. to generate HDL code from Simulink models, Embedded MATLAB code, and Stateflow charts. Simulink HDL Coder generates bit-true, cycle-accurate, synthesizable Verilog and VHDL code and test benches [Mat06].

2.2 Xilinx System Generator™

Xilinx System Generator™ is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs. It generates synthesizable HDL code mapped to Xilinx pre-optimized algorithms. Additionally, it provides automatic generation of a HDL test bench [Xil08].

3 DiaHDL tool

We have developed DiaHDL, a tool which generates VHDL code and its test bench from component diagrams [CS10]. This tool runs on any web browser with Java Runtime Environment. Fig. 1 shows the design flow of VHDL and test bench generation using DiaHDL.

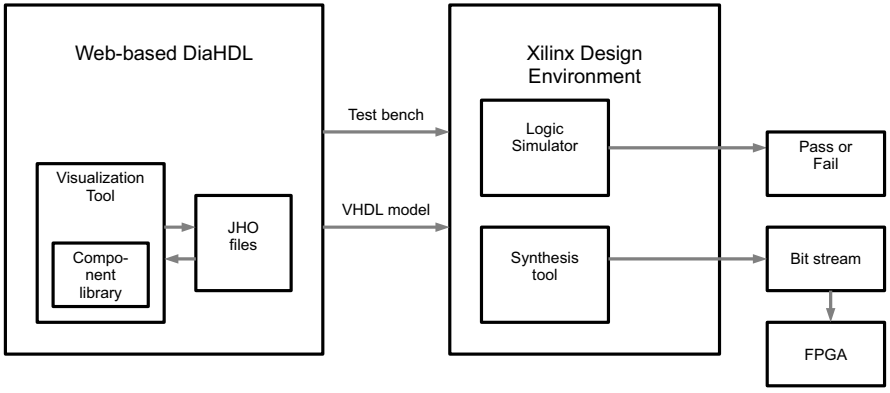

Figure 1: Design flow to generate VHDL code and test bench using DiaHDL

JHO files in DiaHDL contain parameterized VHDL and test bench models as well as java script to create VHDL code dynamically. The visualization tool in DiaHDL converts the component diagram finally to VHDL and test bench code [CS09].

4 Case Study

4.1 VHDL model generation

DiaHDL provides a graphical user interface that allows the user to select a digital component customize their parameters and generate the VHDL code and its test bench. For a case study of this tool we will design a Ripple Carry Adder (RCA).

4.2 VHDL test bench generation

The test bench includes the VHDL model of the design as well as appropriate input test vectors. ModelSim is used for simulation of the test bench. Figure 2 shows the simulation result of a 32-bit RCA using the test bench. The output changes at time (0, 1800, 3600, 5400n) as defined in the test bench.

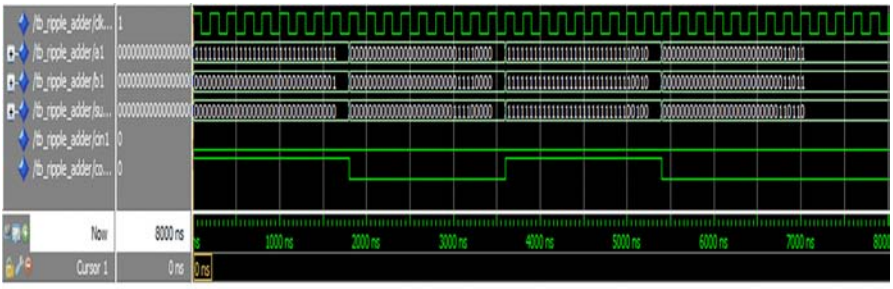

Figure 2: Simulation result of Ripple Carry Adder

4.3 Synthesis

We have synthesized the generated VHDL code with the help of Xilinx ISE tool to target a Xilinx FPGA board.

By providing different values of bit width (8,16,32 bit) and number of pipeline stages (1,2,4,8 stages) for the RCA, DiaHDL generates different VHDL codes. Fig. 3 shows area usage, frequency, and power consumption of this Ripple Carry Adder when the synthesis is done on a VirtexII-Pro FPGA.

	ripple adder 8 bits				ripple adder 16 bits				ripple adder 32 bits			
	1 state	2 states	4 states	8 states	1 state	2 states	4 states	8 states	1 state	2 states	4 states	8 states
IOs	27	27	27	27	51	51	51	51	99	99	99	99
BELS	16	16	16	16	32	32	32	32	64	64	64	64
FlipFlops / Latches	9	10	1	16	17	18	20	24	33	34	36	40
Clock Buffers	1	1	1	1	1	1	1	1	1	1	1	1
IO Buffers	26	26	26	26	50	50	50	50	98	98	98	98
Num of Slices	8	8	8	8	16	16	16	16	32	32	32	32
Num of Slice Flip Flops	9	10	12	16	17	18	20	24	33	34	36	40
Num of 4 input LUTs	16	16	16	16	32	32	32	32	64	64	64	64
Num of IOs	27	27	27	27	51	51	51	51	99	99	99	99
Num of bonded IOBs:	27	27	27	27	51	51	51	51	99	99	99	99
Num of GCLKs:	1	1	1	1	1	1	1	1	1	1	1	1
Max Frequency (MHz)		298.329	470.588	717.36		172.263	298.329	470.588		93.336	172.236	298.329
Minimum period (ns)		3.352	2.125	1.394		5.806	3.352	2.125		10.714	5.806	3.352
Min input arrival time (ns)	8.172	4.676	2.928	2.054	15.164	8.172	4.676	2.928	29.148	15.164	8.172	4.676
Max output req time (ns)	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615
Levels of Logic	10	4	2	1	18	8	4	2	34	16	8	4
Data Path Delay (ns)	8.172	2.6	1.654	1.595	15.164	4.783	2.915	1.982	29.148	12.153	5.33	3.465
Estimated power (mW)	33	32	32	32	37	39	37	36	49	50	48	47

Figure 3: Synthesis result of Ripple Carry Adder

As shown in Fig. 3 the frequency will increase in line with the addition of the number of pipeline stages. Power consumption is almost stable, but area increases slightly.

The disadvantage of the RCA is that it is slow when the bit width is high. The Carry Look-Ahead Adder (CLA) solves this problem by pre-calculation the carry signals, based on the input signals. Fig. 4 shows area usage, frequency, and power consumption of a CLA when the synthesis is done on the same FPGA. Obviously, RCA has the least chip area because of its simplicity in structure. But CLA is faster than RCA for the same bit width and the same number of pipeline stages. Regarding power consumption, RCA is the least power consuming structure because of less hardware usage.

	ripple adder 8 bits				ripple adder 16 bits				ripple adder 32 bits			
	1 state	2 states	4 states	8 states	1 state	2 states	4 states	8 states	1 state	2 states	4 states	8 states
IOs	27	27	27	27	51	51	51	51	99	99	99	99
BELS	16	24	22	16	32	64	56	46	64	140	160	120
FlipFlops / Latches	17	18	20	24	33	34	36	40	65	66	68	72
Clock Buffers	1	1	1	1	1	1	1	1	1	1	1	1
IO Buffers	26	26	26	26	50	50	50	50	98	98	98	98
Num of Slices	9	14	13	14	18	36	32	26	37	81	89	69
Num of Slice Flip Flops	16	18	20	24	32	34	36	40	64	66	68	72
Num of 4 input LUTs	16	24	22	16	32	64	56	46	64	140	160	120
Num of IOs	27	27	27	27	51	51	51	51	99	99	99	99
Num of bonded IOBs:	27	27	27	27	51	51	51	51	99	99	99	99
Num of GCLKs:	1	1	1	1	1	1	1	1	1	1	1	1
Max Frequency (MHz)	746.826	688.705	706.215	717.36	746.83	651.042	688.705	706.215	746.826	609.756	651.042	688.705
Minimum period (ns)	1.339	1.452	1.416	1.394	1.339	1.536	1.452	1.339	1.64	1.536	1.452	
Min input arrival time (ns)	8.172	4.676	2.928	2.054	15.164	8.172	4.676	2.928	29.148	15.164	8.172	4.676
Max output req time (ns)	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615	3.615
Levels of Logic	1	1	1	1	1	1	1	1	1	1	1	1
Data Path Delay (ns)	1.339	1.452	1.416	1.394	1.339	1.536	1.452	1.416	1.581	2.988	2.201	2.088
Estimated power (mW)	33	34	33	32	37	45	40	37	46	70	63	52

Figure 4: Synthesis result of Carry Look-Ahead Adder

5 Conclusion

In this paper we have discussed a web-based electronic engineering tool, DiaHDL, using a case study. We have shown that with the help of DiaHDL a synthesizable VHDL model of a digital circuit can be generated. As this tool is web-based, no local installation is needed. Therefore, this tool ensures an easy access to students at any place in the world. There are many tools on the market which can generate VHDL and test bench code but to our best knowledge they are not online tools. DiaHDL has proven its ability to generate synthesizable VHDL code for simple arithmetic units. But further research is needed for building complex DSP systems out of component diagrams.

References

[CS09] Christyowidiasmoro and T. Schumann. VHDL Code Generator based on Component Diagrams with JavaScript and Object Serialization. In *Proc. SITIA 2009*, Surabaya, Indonesia, 2009.

[CS10] Christyowidiasmoro and T. Schumann. DiaHDL: A Web-Based VHDL Code Generator. accepted paper at: ECUMICT 2010, Gent, Belgium, 2010.

[Mat06] Simulink HDL Coder. Technical report, The MathWorks Inc., 2006.

[Xil08] System Generator for DSP. User guide, Xilinx Inc., 2008.