High-level synthesis

From Wikipedia, the free encyclopedia

High-level synthesis (**HLS**), sometimes referred to as **C synthesis**, **electronic system-level (ESL) synthesis**, **algorithmic synthesis**, or **behavioral synthesis**, is an automated design process that interprets an algorithmic description of a desired behavior and creates digital hardware that implements that behavior.[1] Synthesis begins with a high-level specification of the problem, where behavior is generally decoupled from e.g. clock-level timing. Early HLS explored a variety of input specification languages.,[2] although recent research and commercial applications generally accept synthesizable subsets of ANSI C/C++/SystemC/MATLAB. The code is analyzed, architecturally constrained, and scheduled to create a register-transfer level (RTL) hardware description language (HDL), which is then in turn commonly synthesized to the gate level by the use of a logic synthesis tool. The goal of HLS is to let hardware designers efficiently build and verify hardware, by giving them better control over optimization of their design architecture, and through the nature of allowing the designer to describe the design at a higher level of abstraction while the tool does the RTL implementation. Verification of the RTL is an important part of the process.[3]

Hardware design can be created at a variety of levels of abstraction. The commonly used levels of abstraction are gate level, register-transfer level (RTL), and algorithmic level.

While logic synthesis uses an RTL description of the design, high-level synthesis works at a higher level of abstraction, starting with an algorithmic description in a high-level language such as SystemC and ANSI C/C++. The designer typically develops the module functionality and the interconnect protocol. The high-level synthesis tools handle the micro-architecture and transform untimed or partially timed functional code into fully timed RTL implementations, automatically creating cycle-by-cycle detail for hardware implementation.[4] The (RTL) implementations are then used directly in a conventional logic synthesis flow to create a gate-level implementation.

Contents

History

Early academic work extracted scheduling, allocation, and binding as the basic steps for high-level-synthesis. Scheduling partitions the algorithm in control steps that are used to define the states in the finite-state machine. Each control step contains one small section of the algorithm that can be performed in a single clock cycle in the hardware. Allocation and binding maps the instructions and variables to the hardware components, multiplexers, registers and wires of the data path.

First generation behavioral synthesis was introduced by Synopsys in 1994 as Behavioral Compiler[5] and used Verilog or VHDL as input languages. The abstraction level used was partially timed (clocked) processes. Tools based on behavioral Verilog or VHDL were not widely adopted in part because neither languages nor the partially timed abstraction were well suited to modeling behavior at a high level. 10 years later, in early 2004, Synopsys end-of-lifed Behavioral Compiler.[6]

In 2004, there emerged a number of next generation commercial high-level synthesis products (also called behavioral synthesis or algorithmic synthesis at the time) which provided synthesis of circuits specified at C level to a register transfer level (RTL) specification.[7] Synthesizing from the popular C language offered accrued abstraction, expressive power and coding flexibility while tying with existing flows and legacy models. This language shift, combined with other technical advances was a key enabler for successful industrial usage. High-level synthesis tools are used for complex ASIC and FPGA design.

High-level synthesis was primarily adopted in Japan and Europe in the early years. As of late 2008, there was an emerging adoption in the United States.[8]

Source input

The most common source inputs for high-level synthesis are based on standard languages such as ANSI C/C++, SystemC and MATLAB.

High-level synthesis typically also includes a bit-accurate executable specification as input, since to derive an efficient hardware implementation, additional information is needed on what is an acceptable Mean-Square Error or Bit-Error Rate etc. For example, if the designer starts with an FIR filter written using the "double" floating type, before he or she can derive an efficient hardware implementation, they need to perform numerical refinement to arrive at a fixed-point implementation. The refinement requires additional information on the level of quantization noise that can be tolerated, the valid input ranges etc. This bit-accurate specification makes the high level synthesis source specification functionally complete.[9] Normally the tools infer from the high level code a Finite State Machine and a Datapath that implement arithmetic operations.

Process stages

The high-level synthesis process consists of a number of activities. Various high-level synthesis tools perform these activities in different orders using different algorithms. Some high-level synthesis tools combine some of these activities or perform them iteratively to converge on the desired solution.[10]

- Lexical processing
- Algorithm optimization
- Control/Dataflow analysis
- Library processing
- Resource allocation
- Scheduling
- Functional unit binding
- Register binding
- Output processing
- Input Rebundling

Functionality

In general, an algorithm can be performed over many clock cycles with few hardware resources, or over fewer clock cycles using a larger number of ALUs, registers and memories. Correspondingly, from one algorithmic description, a variety of hardware microarchitectures can be generated by an HLS compiler according to the directives given to the tool. This is the same trade off of execution speed for hardware complexity as seen when a given program is run on conventional processors of differing performance, yet all running at roughly the same clock frequency.

Architectural constraints

Synthesis constraints for the architecture can automatically be applied based on the design analysis.[3] These constraints can be broken into

- Hierarchy
- Interface
- Memory
- Loop
- Low-level timing constraints
- iteration

Interface synthesis

Interface Synthesis refers to the ability to accept pure C/C++ description as its input, then use automated interface synthesis technology to control the timing and communications protocol on the design interface. This enables interface analysis and exploration of a full range of hardware interface options such as streaming, single- or dual-port RAM plus various handshaking mechanisms. With interface synthesis the designer does not embed interface protocols in the source description. Examples might be: direct connection, one line, 2 line handshake, FIFO.[11]

Vendors

Data reported on recent Survey[12]

Status	Compiler	Owner	License	Input	Output	Year	Domain	TestBench	FP	FixP
Announced	A++ (https://web.archive.org/web/20161010221724/https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/spectra-q.html)	Altera	Commercial	C/C++	VHDL/Verilog	2016	All	?	?	?
	AUGH (http://tima.imag.fr/sls/research-projects/augh/)	TIMA Lab.	Academic	C subset	VHDL	2012	All	Yes	No	No
	eXCite (http://www.yxi.com)	Y Explorations	Commercial	C	VHDL/Verilog	2001	All	Yes	No	Yes
	Bambu (http://panda.dei.polimi.it/?page_id=81)	PoliMi	Academic	C	VHDL/Verilog	2012	All	Yes	Yes	No
	Bluespec	BlueSpec Inc.	Commercial	BSV	SystemVerilog	2007	All	No	No	No
	CHC	Altium	Commercial	C subset	VHDL/Verilog	2008	All	No	Yes	Yes
	CoDeveloper	Impulse Accelerated	Commercial	Impulse-C	VHDL	2003	Image Streaming	Yes	Yes	No
	Stratus	Cadence	Commercial	C/C++ SystemC	RTL	2015	All	Yes	No	Yes
	Cyber-Workbench	NEC	Commercial	BDL, SystemC	VHDL/Verilog	2011	All	Cycle/Formal	Yes	Yes
	Catapult	Mentor (Siemens	Commercial	C, C++,	VHDL/Verilog	2004	Streaming	No	No	Yes

		business)		SystemC						
	DWARV	TU. Delft	Academic	C subset	VHDL	2012	All	Yes	Yes	Yes
	GAUT (http://www.gaut.fr)	U. Bretagne	Academic	C/C++	VHDL	2010	DSP	Yes	No	Yes
	Hastlayer (https://hastlayer.com/)	Lombiq Technologies	Commercial	.NET (C#/C++/F#...)	VHDL	2015	.NET	Yes	No	Yes
	LegUp HLS (https://www.legupcomputing.com)	LegUp Computing	Commercial	C	Verilog	2017	All	Yes	Yes	No
In Use	LegUp (http://legup.eecg.utoronto.ca)	U. Toronto	Academic	C	Verilog	2011	All	Yes	Yes	No
	MaxCompiler	Maxeler	Commercial	MaxJ	RTL	2010	DataFlow	No	Yes	No
	ROCCC (http://roccc.cs.ucr.edu/)	Jacquard Comp.	Commercial	C subset	VHDL	2010	Streaming	No	Yes	No
	Symphony C	Synopsys	Commercial	C/C++	VHDL/Verilog/SystemC	2010	All	Yes	No	Yes
	VivadoHLS (http://www.xilinx.com/products/design-tools/vivado/index.htm) (formerly AutoPilot from AutoESL[13])	Xilinx	Commercial	C/C++/SystemC	VHDL/Verilog/SystemC	2013	All	Yes	Yes	Yes
	Kiwi (http://www.cl.cam.ac.uk/research/srg/han/hprls/orangepath/kiwic.html)	U. Cambridge	Academic	C#	Verilog	2008	.NET	No	Yes	Yes
	CHiMPS	U. Washington	Academic	C	VHDL	2008	All	No	No	No
	gcc2verilog	U. Korea	Academic	C	Verilog	2011	All	No	No	No
	HercuLeS	Ajax Compilers	Commercial	C/NAC	VHDL	2012	All	Yes	Yes	Yes
	Shang (https://github.com/etherzhhb/Shang)	?	U. Illinois	C	Verilog	2013	All	Yes	?	?
	Trident	Los Alamos NL	Academic	C subset	VHDL	2007	Scientific	No	Yes	No
	AccelDSP	Xilinx	Commercial	MATLAB	VHDL/Verilog	2006	DSP	Yes	Yes	Yes
	C2H	Altera	Commercial	C	VHDL/Verilog	2006	All	No	No	No
	CtoVerilog	U. Haifa	Academic	C	Verilog	2008	All	No	No	No

Abandoned	DEFACTO	U. South Cailf.	Academic	C	RTL	1999	DSE	No	No	No
	Garp	U. Berkeley	Academic	C subset	bitstream	2000	Loop	No	No	No
	MATCH	U. Northwest	Academic	MATLAB	VHDL	2000	Image	No	No	No
	Napa-C	Sarnoff Corp.	Academic	C subset	VHDL/Verilog	1998	Loop	No	No	No
	PipeRench	U.Carnegie M.	Academic	DIL	bistream	2000	Stream	No	No	No
	SA-C	U. Colorado	Academic	SA-C	VHDL	2003	Image	No	No	No
	SeaCucumber	U. Brigham Y.	Academic	Java	EDIF	2002	All	No	Yes	Yes
	SPARK	U. Cal. Irvine	Academic	C	VHDL	2003	Control	No	No	No

- MATLAB HDL Coder from Mathworks[14]
- HLS-QSP from CircuitSutra Technologies[15]
- C-to-Silicon from Cadence Design Systems
- Concurrent Acceleration from Concurrent EDA
- Symphony C Compiler from Synopsys
- QuickPlay from PLDA[16]
- PowerOpt from ChipVision[17]
- Cynthesizer from Forte Design Systems, acquired by Cadence Design Systems on 2014, February 14
- Catapult C from Calypto Design Systems, part of Mentor Graphics as of 2015, September 16
- CyberWorkBench from NEC[18]
- Mega Hardware [19]
- C2R from CebaTech[20]
- CoDeveloper from Impulse Accelerated Technologies
- HercuLeS by Nikolaos Kavvadias[21]
- PICO from Synfora, acquired by Synopsys in June 2010[22] (PICO = Program In/Code Out)
- xPilot from University of California, Los Angeles[23]
- Vsyn from vsyn.ru[24]
- ngDesign from SynFlow[25]

See also

- C to HDL
- Electronic design automation (EDA)
- Electronic system-level (ESL)
- Logic synthesis
- High-level verification (HLV)
- SystemVerilog

References

1. "High-Level Synthesis - Springer" (http://www.springerlink.com/content/978-1-4020-8587-1). *Springerlink.com*. Retrieved 2016-10-03.
2. IEEE Xplore High-Level Synthesis: Past, Present, and Future (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5209959) DOI 10.1109/MDT.2009.83
3. "The 'why' and 'what' of algorithmic synthesis" (http://www.eetimes.com/news/design/showArticle.jhtml?articleID=162100329). EE Times. Retrieved 2016-10-03.

9). EE Times. Retrieved 2016-10-03.
4. "C-Based Rapid Prototyping for Digital Signal Processing" (http://www.ee.bilkent.edu.tr/~signal/defevent/papers/cr1179.pdf) (PDF). UBS University, France. Retrieved 2016-10-03.
5. "Publications and Presentations" (https://web.archive.org/web/20080426075303/http://www.bdti.com/articles/info_dspx95asics.htm). *Bdti.com*. Archived from the original (http://www.bdti.com/articles/info_dspx95asics.htm) on 2008-04-26. Retrieved 2016-10-03.
6. "Behavioral synthesis crossroad" (http://www.eetimes.com/news/design/columns/tool_talk/showArticle.jhtml?articleID=18700196). EE Times. Retrieved 2016-10-03.
7. [1] (http://archives.eetimes.com/high-level-synthesis-rollouts-enable-esl/110436.html)
8. [2] (https://web.archive.org/web/20111003070615/http://www.scdsource.com/article.php?id=322)
9. Multiple Word-Length High-Level Synthesis (http://www.hindawi.com/GetArticle.aspx?doi=10.1155/2008/916867&e=html) EURASIP Journal on Embedded Systems
10. "A look inside behavioral synthesis" (http://www.eetimes.com/document.asp?doc_id=1217645). EE Times. Retrieved 2016-10-03.
11. [3] (https://web.archive.org/web/20100925034609/http://www.designcon.com/infovault/paper.asp?PAPER_ID=407)
12. Nane, R.; Sima, V. M.; Pilato, C.; Choi, J.; Fort, B.; Canis, A.; Chen, Y. T.; Hsiao, H.; Brown, S. (2016-01-01). "A Survey and Evaluation of FPGA High-Level Synthesis Tools" (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7368920). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. **PP** (99): 1–1. doi:10.1109/TCAD.2015.2513673 (https://doi.org/10.1109%2FTCAD.2015.2513673). ISSN 0278-0070 (https://www.worldcat.org/issn/0278-0070).
13. "Xilinx buys high-level synthesis EDA vendor" (http://www.eetimes.com/electronics-news/4212668/Xilinx-buys-high-level-synthesis-EDA-vendor). EE Times. 2011-02-05. Retrieved 2016-10-03.
14. "MathWorks – Makers of MATLAB and Simulink" (http://mathworks.com/). *Mathworks.com*. Retrieved 2016-10-03.
15. "SystemC based ESL methodologies - SystemC based ESL methodologies" (http://www.circuitsutra.com). *Circuitsutra.com*. Retrieved 2016-10-03.
16. John M. at a major ERP & DBMS Corporation (2016-08-29). "QuickPlay: Bringing FPGA Computing to the Masses" (http://www.quickplay.io). *Quickplay.io*. Retrieved 2016-10-03.
17. [4] (https://web.archive.org/web/20020530041720/http://www.chipvision.com/)
18. "CyberWorkBench: Products" (http://www.nec.com/en/global/prod/cwb/index.html). NEC. Retrieved 2016-10-03.
19. [5] (https://web.archive.org/web/20040115020254/http://www.mega-hardware.com/)
20. [6] (https://web.archive.org/web/20050507101111/http://www.cebatech.com/)
21. "Nikolaos Kavvadias - HercuLeS high-level synthesis tool" (http://www.nkavvadias.com/hercules/). *Nkavvadias.com*. Retrieved 2016-10-03.
22. "Synopsys buys Synfora assets" (http://www.eetimes.com/electronics-news/4200083/Synopsys-buys-Synfora-assets). EE Times. Retrieved 2016-10-03.
23. "The xPilot System" (http://cadlab.cs.ucla.edu/soc/). *Cadlab.cs.ucla.edu*. Retrieved 2016-10-03.
24. "vSyn.ru" (http://www.vsyn.ru). *vSyn.ru*. 2016-06-16. Retrieved 2016-10-03.
25. "Hardware design for all" (https://www.synflow.com/). Synflow. Retrieved 2016-10-03.

Further reading

- Michael Fingeroff (2010). *High-Level Synthesis Blue Book*. Xlibris Corporation. ISBN 978-1-4500-9724-6.
- Coussy, P.; Gajski, D. D.; Meredith, M.; Takach, A. (2009). "An Introduction to High-Level Synthesis". *IEEE Design & Test of Computers*. **26** (4): 8–17. doi:10.1109/MDT.2009.69 (https://doi.org/10.1109%2FMDT.2009.69).
- Ewout S. J. Martens; Georges Gielen (2008). *High-level modeling and synthesis of analog integrated systems*. Springer. ISBN 978-1-4020-6801-0.
- Saraju Mohanty, N. Ranganathan, E. Kougianos, and P. Patra (2008). *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*. Springer. ISBN 978-0387764733.
- Alice C. Parker; Yosef Tirat-Gefen; Suhrid A. Wadekar (2007). "System-Level Design". In Wai-Kai Chen. *The VLSI handbook* (2nd ed.). CRC Press. ISBN 978-0-8493-4199-1. chapter 76.
- Shahrzad Mirkhani; Zainalabedin Navabi (2007). "System Level Design Languages". In Wai-Kai Chen. *The VLSI handbook* (2nd ed.). CRC Press. ISBN 978-0-8493-4199-1. chapter 86. covers the use of C/C++, SystemC, TML and even UML
- Liming Xiu (2007). *VLSI circuit design methodology demystified: a conceptual taxonomy*. Wiley-IEEE. ISBN 978-0-470-12742-1.
- John P. Elliott (1999). *Understanding behavioral synthesis: a practical guide to high-level design*. Springer. ISBN 978-0-7923-8542-4.

- Razvan Nane, Vlad-Mihai Sima, Christian Pilato, Jongsok Choi, Blair Fort, Andrew Canis, Yu Ting Chen, Hsuan Hsiao, Stephen Brown, Fabrizio Ferrandi, Jason Anderson, Koen Bertels. "*A Survey and Evaluation of FPGA High-Level Synthesis Tools*". *In* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume:PP, Issue: 99). ISSN 0278-0070.

External links

- Vivado HLS course on Youtube (https://www.youtube.com/watch?v=kgae3Wzqngs&list=PLo7bVbJhQ6qzK6ELKCm8H_WEzzcr5YXHC)
- Deepchip Discussion Forum (http://www.deepchip.com/posts/0480.html)