Master thesis proposal

go2async: A high-level synthesis tool for asynchronous circuits based on click-elements

Sebastian Michael Wiedemann
Matr. No.: 01425647

Institute of Computer Engineering
Embedded Computing System Group
TU Wien

Advisors
Univ.Ass. Jürgen Maier
Prof. Andreas Steininger

November 28, 2021
Vienna

1 Problem Description

The everlasting need for more and more computer performance while maintaining power, resource and cost efficiency will always be a problem for the computer industry. One way to tackle this problem is by *simply* getting rid of the huge overhead that is created by computer systems that support software execution for rather easy or highly parallel tasks. This can be done by using specific hardware accelerators, e.g. dynamically loaded into an FPGA, to run specific tasks very efficiently in hardware.

However, creating hardware introduces its own huge overhead, especially regarding verification. It is known that the design flow from the specification to an ASIC is risky in regards to design, time to market, and market adoption. Overall, ASIC designs are in general rather cumbersome and very expensive [7]. This makes custom hardware designs only feasible if large quantities are demanded.

One way to reduce these risks is to create ASICs using hardware description languages (HDLs; e.g. VHDL, Verilog). This way, virtual prototypes can be simulated. Additionally, prototypes can be loaded into an FPGA and the result can be physically tested. This also provides a more dynamic workflow. Nevertheless, the complexity of the design flow of prototypes is not to be underestimated. These languages still require an extensive verification effort.

This is especially the case for asynchronous circuits, which use dedicated signal lines to indicate when new data is ready or when it can be processed. Compared to their synchronous counterparts, which are controlled by a central clock, they have the advantage that clock related problems such as clock skews are eliminated. Asynchronous circuits also do not need to make worst-case assumptions (internal and external) about calculation speeds since these circuits basically *detect* when calculations are ready by design (e.g. handshake, coding). Additionally, asynchronous circuits generally operate on lower power because they save much by omitting the toggling of clock lines and having transitions only in areas where active computations take place.

On the flipside, asynchronous logic needs more delicate care and much more attention to detail to the dynamic state of the circuit to avoid potential hazards. Asynchronous circuits also need control logic to ensure the operations are ordered which is simply solved by the placement of latches in its synchronous counterpart.

This makes unassisted designs of large asynchronous circuits almost unbearable.

2 Expected Results

The overall goal of this project is to decrease the complexity of asynchronous hardware generation such that even untrained personal is quickly able to achieve satisfying results. A high-level synthesis tool does exactly this. Such tools take standard software code (mostly subsets of a programming languages) and try to generate hardware (mostly in form of an HDL) which behaves just like its software counterpart would.

The expected result of this thesis is a high-level synthesis tool (go2async) written in the *go* programming language [4] which parses a subset of *go* itself with the built-in go parser. Initially, the supported go subset consists of standard program flow structures (e.g. for, if, jumps), arithmetic, and binary operators. Only binary and integer variable types are allowed. Fixed sized arrays can be used.

Go2async takes a *go* function as an input and generates synthesizable VHDL code based on click-elements [6] representing an asynchronous circuit. The result mimics the functionality of the parsed function. The function's parameters and return values are the inputs and outputs of a component.

The evaluation of the tool shall be executed by comparing resulting circuits to manually created ones and other projects that are based on click-elements (e.g. RISC-V processor [5]). Additionally, I want to compare the results to their synchronous counterparts in regards to performance and area usage.

At the end of the thesis, the following research questions shall be answered:

RQ1: How can the generation of asynchronous circuits based on click-elements be automated?

RQ2: How much time in development and verification of asynchronous circuits can be saved by using go2async?

RQ3: How do the generated circuits compare to functional equivalent synchronous solutions?

3 Methodological Approach

The thesis is executed in the following steps. These occur in iterations, starting with a simpler lightweight generator, which will be gradually improved. This yields the advantage of a more structured approach and thus simpler verification.

Literature study: As a first step, I will take a look into literature to explore other high-level synthesis tools and get different ideas of VHDL code generation. Additionally, exploring

other projects done with click-elements might be useful for comparisons or might even affect implementation decisions.

Implementation: In this step, the tool is actually implemented. The initial plan is to start small and iteratively make the generator better and more powerful. This implies that at the start, a smaller subset of the *go* language will be supported rather than implementing everything at once.

Simulation, Analysis and Comparison: Here the results of the generator get tested. Simulations are used to identify problems and to find possible optimizations. Results can be compared to different versions and implementations from other papers. Results will also be tested on FPGAs.

Optimization: This step can be done during or after the implementation. Especially hardware usage of FPGAs (registers, logic elements, etc.) will be considered in this step. For instance, in software, variables have a lifespan and are only relevant in certain scopes. Instead of reserving latches and wires for every piece of data everywhere in hardware it would be very beneficial to the area usage of the FPGA to introduce similar scopes in the hardware design too.

4 State of the Art

Many approaches for high-level synthesis of asynchronous circuits, mainly developed by universities, have been proposed in the past. Large-scale asynchronous circuits are based on a technique called *syntax directed translation* [10]. This is essentially a one-to-one mapping of the syntax-tree of the source program into a corresponding structure of handshake components.

Another class of research described in [10] focuses on converting clocked circuits into asynchronous ones by using a set of conversion rules (de-synchronization). This does not enable designers to use the full potential of asynchronous circuits but it yields other benefits. For instance, this method tackles problems that occur from dynamic voltage drops and power grid noises caused by synchronous circuits.

Notably, there are recent motivations to introduce HLS tools in education for undergraduate computer engineers [8].

One of the classic asynchronous pipelines is *Micropipelines* [11]. Their backbone control circuit is the well-known Muller pipeline based on C-elements. This is already one of the big disadvantages of this design, since its components are not in standard-cell libraries, thus not ideal for FPGA implementations.

MOUSETRAP [9] was introduced to be faster and more efficient than its predecessor *Micropipelines*. An additional goal was to use mostly simple structures (XOR, latches)

for control, and an efficient plus highly-concurrent event-driven protocol. Unfortunately, MOUSETRAP's goals were not completely met. It still needs C-elements for more complex pipelines such as fork and join components.

This is where click-elements really shine. They work entirely on standard-cell library components [6] and are thus very suitable for FPGA implementations. The paper also provides a github project [1] which uses the described hardware. Every example in this github project is manually programmed, which turns out to be rather bothersome and complex to do. However, with the help of the tool this thesis presents, these examples could be generated with simple *go* function inputs.

An example of a project based on click-elements is the low-power asynchronous RISC-V processor proposed by Li *et al.* [5].

A commercially active high-level synthesis tool is LegUp [2]. This tool takes C/C++ sources as input. A software programmer can mark parts that should be hardware accelerated (e.g. a loop).

To generate accelerators, the tool maps C code in LLVM intermediate representation (IR) instructions. These instructions can be synthesized in an HDL directly since one C statement might generate more than one IR instruction. This enables the usage of debugging tools which can highlight the generated HDL code lines to the corresponding C code [3].

Overall, LegUp synthesizes a processor/accelerator hybrid system. The program can be executed on an FPGA where the programmer's marked parts are hardware accelerated and the rest runs normally on a processor. This processor enables this tool to execute any C source, which also implies an extensive area usage.

5 Relation to Computer Engineering

Designing new hardware circuits is a typical Computer Engineering problem. In fact, this thesis is very heavily related to *Advanced Digital Design*: Firstly, it is directly inspired by a project on click-elements that I worked on during this class. Secondly, the fundamental knowledge about asynchronous circuits this thesis is built upon is taught there.

Further relations can be found to the lecture *Advanced Computer Architecture* due to the architecture features used in this thesis and *Advanced FPGA Design* in regards to smart hardware design decisions for FPGAs.

References

[1] *Async-Click-Library*. URL: `https://github.com/zuzkajelcicova/Async-Click-Library`. (accessed: 17.11.2021).

[2] Andrew Canis et al. “From software to accelerators with LegUp high-level synthesis”. In: *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*. 2013, pp. 1–9. DOI: 10.1109/CASES.2013.6662524.

[3] Andrew Canis et al. “LegUp: An Open-Source High-Level Synthesis Tool for FPGA-Based Processor/Accelerator Systems”. In: *ACM Transactions on Embedded Computing Systems (TECS)* 13 (Sept. 2013). DOI: 10.1145/2514740.

[4] *Go programming language*. URL: https://golang.org/. (accessed: 17.11.2021).

[5] Zhiyu Li et al. “A Low-Power Asynchronous RISC-V Processor With Propagated Timing Constraints Method”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 68.9 (2021), pp. 3153–3157. DOI: 10.1109/TCSII.2021.3100524.

[6] Adrian Mardari, Zuzana Jelcicova, and Jens Sparso. “Design and FPGA-implementation of Asynchronous Circuits Using Two-Phase Handshaking”. In: May 2019, pp. 9–18. DOI: 10.1109/ASYNC.2019.00010.

[7] Olga Melnikova, Irina Hahanova, and Karina Mostovaya. “Using multi-FPGA systems for ASIC prototyping”. In: *2009 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics*. 2009, pp. 237–239.

[8] Isaac Nelson et al. “Is It Time to Include High-Level Synthesis Design in Digital System Education for Undergraduate Computer Engineers?” In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401774.

[9] Montek Singh and Steven M. Nowick. “MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15.6 (2007), pp. 684–698. DOI: 10.1109/TVLSI.2007.898732.

[10] Jens Sparsø. “Current trends in high-level synthesis of asynchronous circuits”. In: *2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)*. 2009, pp. 347–350. DOI: 10.1109/ICECS.2009.5411011.

[11] I. E. Sutherland. “Micropipelines”. In: *Commun. ACM* 32.6 (June 1989), pp. 720–738. ISSN: 0001-0782. DOI: 10.1145/63526.63532. URL: https://doi.org/10.1145/63526.63532.