Synthesis of Steel-ASIC, a RISC-V Core

Rafael da Silva[1], Vinícius dos Santos[2], Fábio Petkowicz[3], Rafael Calçada[4] and Ricardo Reis[5]

[1-5] PGMicro-Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazi
e-mail: rsilva@inf.ufrgs.br

Abstract— **It is presented the design flow of an ASIC version of STEEL, a RISC-V microprocessor developed at UFRGS. The microprocessor core called STEEL implements the RV32I and Zicsr instruction sets of the RISC-V specifications. The whole process entails logical and physical synthesis, using the X-Fab 180 nm, which relies on the Cadence EDA framework. The ASIC circuit operates with a maximum frequency of 19.61 MHz and the estimates obtained from the physical synthesis indicates a power consumption of 10.09 mW.**

Index Terms— **VLSI, RISC-V, Steel Core, ASIC, microprocessor, microelectronics**

I. Introduction

Integrated circuits (ICs) revolutionized electronics, and today they are present in most technology applications and in a great part of the industrial products [1]. For example, the microprocessors, almost every electronic device available today, are powered by a general-purpose microprocessor: computers, smartphones, digital televisions, smart speakers, etc. According to the Statista Research Department [2], the worldwide integrated circuit market reached 361.23 billion U.S. dollars in revenue, in 2020. Furthermore, the estimated market grow in 2021 is by over 20 percent to 436.37 billion U.S. dollars [2]. Application-Specific Integrated Circuits (ASICs) are ICs manufactured for two main purposes: (1) to meet user's specifications for the demands of a particular system; or (2) for reuse, so that several other macrosystems can be designed considering the already finalized ASIC as a component [3]. A traditional design methodology for ASICs is the standard cell one, which consists of mapping a complex circuit into pre-designed logic cells. The cell library contains the description of the layout of each logic cell.

This article extends the previous work at UFRGS [4], which describes the step-by-step taken to turn a hardware description into a design Steel Core ASIC, reporting the objective and result of each stage of the synthesis. Steel is a microprocessor with a 3-stage pipeline that implements the RV32I, and Zicsr instruction sets the RISC-V specifications, whose description in RTL - Register Transfer Level was developed at UFRGS and is freely available at the OpenCores.org portal [5]. This work brings updated related work and a complete state-of-the-art description.

This paper is organized as follow: In Section II. state-of-the-art and related works are presented. Section III. the Steel microprocessor is described. Section IV. depicts the methodology of this work, including the entire synthesis process of the microprocessor. Section V. presents the estimates obtained from the complete synthesis. Finally, the conclusions are reported in Section VI..

II. Related Work

Many RISC-V ISA (Instruction Set Architecture) implementations are freely available, and some are being marketed. On the RISC-V International website, you can find a list of currently available cores and System-on-Chip (SoC) [6]. Within state-of-the-art, three notable projects related to RISC-V are called BOOM, Rocket, and PULP. The two implementations (Ibex and SCR1) will also be presented.

BOOM is an open-source processor developed by researchers at the University of California, Berkeley, which implements the RISC-V ISA RV64G [7]. Like most high-performance cores, it is superscalar (capable of executing more than one instruction per cycle) and out-of-order (capable of running instructions while their dependencies are resolved, without being restricted to program order). BOOM cores are produced by a parameterizable generator written in Chisel [8] called Rocket Chip Generator and can be used both in FPGAs and in the synthesis of ASICs [9]. BOOM version 2 has a 7-stage pipeline and can execute up to four instructions in parallel. Cores are capable of running Linux operating systems. Various implementation parameters are configurable, such as cache sizes, branch prediction types, etc.

Rocket Chip Generator is an open-source SoC generator developed at the University of California, Berkeley, and it is capable of producing SoCs from high-level descriptions. Several generated SoC parameters can be configured, such as the type and number of cores and the size and associativity of caches. The cores can be Rocket (in-order) or BOOM (out-of-order) types. Rocket cores are single-issue (RV32G/RV64G) processors implemented in a 5-stage pipeline. Rocket cores also have MMUs (Memory Management Unit) that support paged virtual memory, non-blocking data caches, and predictors of deviations [10].

The Rocket Chip Generator generates the RTL description of a complete RISC-V system, composed of cores, caches, memory management units, buses, and interfaces for communication with peripherals. According to the authors, the software can generate large systems from tiny microcontrollers to SoC with multiple cores. Rocket chips have been manufactured at least eleven times, leading to functional implementations capable of running Linux operating systems.

PULP (Parallel Ultra Low Power) Platform is a joint research project between ETH Zürich and Università di Bologna dedicated to developing energy-efficient free hardware. The project provides a portfolio of free cores and SoCs, tested in silicon and widely used by the IoT (Internet of Things) and low-power projects [11].

The platform offers three energy-efficient cores: Ibex, RI5CY, and Ariane. Ibex is an RV32IMC (single-issue) core with a 2-stages pipeline. RI5CY is an RV32IMC (single-issue) core with a 4-stages pipeline and optional F extension support. Ariane is the most sophisticated implementation of the platform, an RV64IMAC (single-issue) core with a 6-stages pipeline and support for M (Machine), S (Supervisor), and U (User) modes of the privileged architecture, capable of running Linux systems [12].

The platform also provides the PULPino and PULPissimo SoCs. PULPino is a single-core microcontroller that can be configured to use one of the platform's 32-bit cores, RI5CY or Ibex. PULPissimo is an energy-efficient PULPino-like microcontroller. It can use both in FPGAs and the synthesis of ASICs.

Ibex [13] (previously called Zero-riscy) is an RV32IMCZicsr implementation of the RISC-V ISA. It features single instruction dispatch and in-order execution and supports the privileged architecture's M (machine) and U (user) modes. The U extension is not present in Steel since it is optional. The operative part is implemented in 2-pipeline stages, with optional support for a third stage, still in the testing phase. You can also reduce the number of registers by turning it into an RV32E implementation. Initially developed by ETH Zürich and Università di Bologna researchers, the Ibex project is maintained by lowRISC, a non-profit company focused on collaborative free hardware development. Ibex can use the core in FPGAs and the synthesis of ASICs aimed at ultra-low-power and ultra-low-area applications.

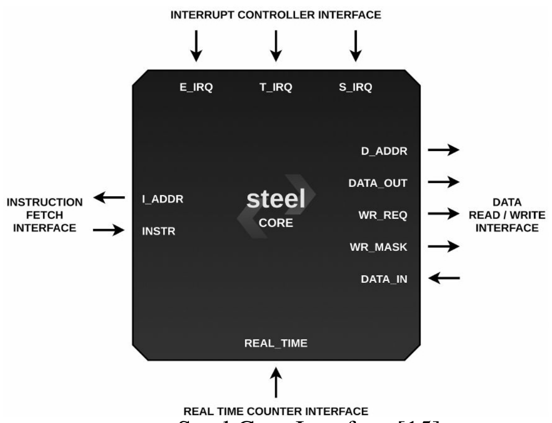
The SCR1 is a core RV32IZicsr (with optional support for M and C extensions) produced by Syntacore and a company specializing in cores and tools for the RISC-V architecture. It is the company's most uncomplicated and the only free and open-source design. Like Ibex, it can be configured to reduce the bank of registers, turning into an RV32E implementation. The operative part of the core can be configured to have between 2 and 4 pipeline stages. Like Ibex, SCR1 is geared towards applications in FPGAs and ASICs, having been successfully used to manufacture microcontrollers and SoCs.

III. Steel Core overview.

There are two main architectural approaches to design computers: CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer), which differ regarding the instructions' quantity and complexity. Generally speaking, RISC architectures have a significantly smaller and simpler instruction set when compared to CISC architectures [14].

Figure 1 presents the Steel top module architecture. There are two interfaces for memory access: connecting it to an interrupt controller and reading from a real-time counter. Its two memory buses allow the construction of a computer system based on the Harvard architecture (connecting it to two memories, one for data and the other for instructions) and the von Neumann architecture (connecting it to a single dual-port memory).

Steel is a 32-bit RISC-V microprocessor core designed to be easy to use and primarily targeted for use as a softcore in embedded system designs. The operating frequency of CPUs rarely exceeds the range of a few hundred MHz, and most of


INTERRUPT CONTROLLER INTERFACE
E_IRQ
T_IRQ
S_IRQ
D_ADDR
DATA_OUT
WR_REQ
WR_MASK
DATA_IN
INSTRUCTION FETCH INTERFACE
I_ADDR
INSTR
steel
CORE
DATA READ / WRITE INTERFACE
REAL_TIME
REAL TIME COUNTER INTERFACE
Fig. 1: Steel Core Interface [15]

these systems do not have strict security requirements. Steel implements the basic RISC-V instruction set RV32I and the Zicsr [16] extension only so that it can be used as a processing unit in small and medium-size embedded systems. Figure 2 presents a simple example system to demonstrate how Steel can be used in a small embedded system. It consists of a RAM unit, a bus arbiter, a UART transmitter, and Steel. The real-time counter signal was linked to a fixed value and the interrupt request signals because, in this presented system, these resources are not used.Other systems with different interfaces and peripherals can be built using a similar architecture. Although they can also use them in large systems, they usually require features still absent in Steel, such as specialized instructions for integer multiplication and division (M extension) and floating-point arithmetic (F extension).

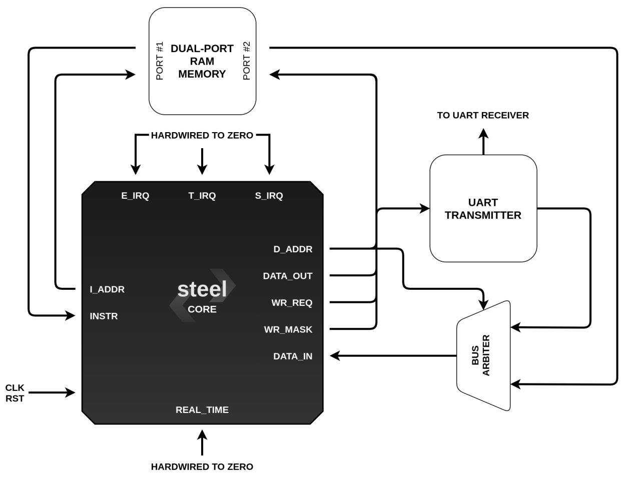


Fig. 2: Example system built with Steel

Nevertheless, Steel can run embedded software and even real-time operating systems (for example, FreeRTOS), taking advantage that the RV32I base instruction set can emulate nearly all extensions. Its repository on the internet is downloaded five times a week on average and has been forked seven users so far. The version freely available at the https://opencores.org/projects/steelcore portal [5] is validated and passed all RISC-V Compliance Suite tests for the RV32I and Zicsr instruction sets. The full FPGA Steel documentation is available at https://rafaelcalcada.github.io/steel-core/.

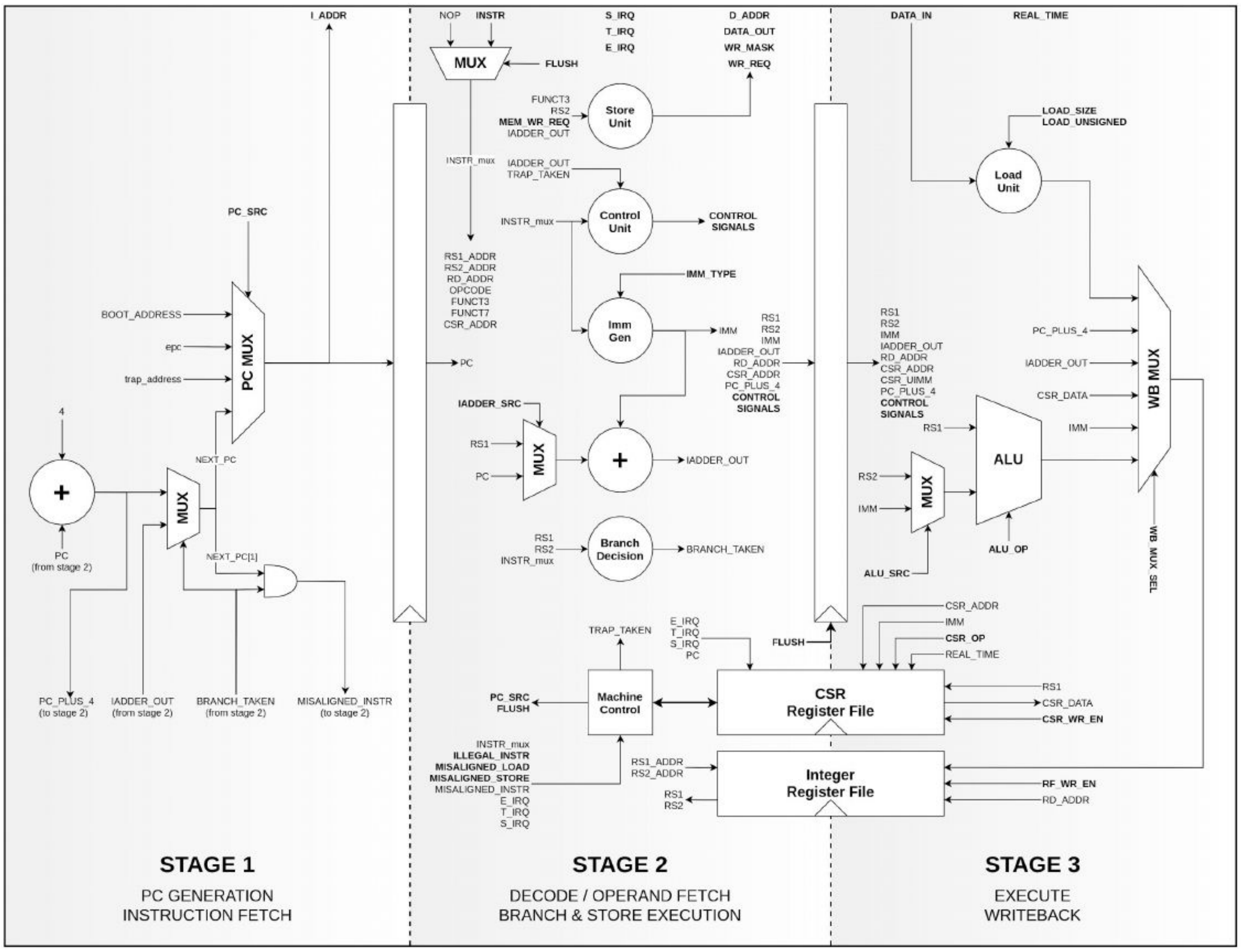


Fig. 3: Steel Microarchitecture [15]

A. *Specification and Architecture*

Figure 3 presents a detailed logic diagram of Steel's microarchitecture, and it was in Verilog following the diagram step by step. Steel has three pipeline stages, a single execution thread, and issues one instruction per clock cycle. Therefore, all instructions are executed in program order. Its pipeline is simple, divided into fetch, decode, and execution stages. The first pipeline stage is responsible for generating the program counter value, which contains the address of the instruction to be fetched. The second stage decodes the instruction, fetches the operands in the Integer Registers Bank, and generates the immediate value. Branch and store instructions are executed ahead of time at this stage. The last stage executes all other instructions and writes the result of operations to the register bank. The small number of pipeline stages eliminates the need for branch predictors and other advanced microarchitectural units, like data hazard detectors and forwarding units, making the design of Steel easy to understand.

IV. Design Methodology

This section presents the synthesis design flow of the microprocessor, using Cadence's tools, which consists of: simulation of the RTL description, the logical synthesis of the RTL description (mapping of the logical units of the circuit into cells from the standard cell library), and finally the physical synthesis of the circuit. Figure 4 presents the design flow overview of the processor synthesis.

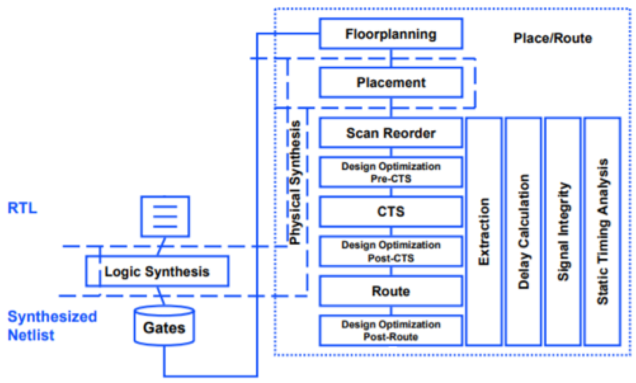


Fig. 4: Design flow overview of the processor synthesis

A. *Register Transfer Level description*

An RTL description is a high-level design abstraction that models the processor logic through control signals and data flow between registers. A testbench was written for the described hardware units to verify that the observed behavior in response to stimuli corresponds to the projected one. We tested the RTL target description using the NClaunch and SimVision analysis environment of Cadence Design Sytems. The detection of errors in this step also led to correcting the hardware description. After the corrections and expected simulation results (confirmed after running the testbench), the circuit validation is carried out.

B. *Logical Synthesis*

The logical synthesis step maps the description in RTL into a description based on standard cells [3]. The used cell library was the XC018 MOSST Digital Core Library [17], from X-FAB Semiconductors Foundries. This cell library uses a 180nm technology. It uses a single polysilicon layer with up to six metal layers, 0.18-micron drawn gate length, N-well process, and dual gate oxide (1.8 V with 3.3 V or 5.0 V) transistors. The logical synthesis was divided into three basic steps: Constraints, Generic Technological Mapping, and Optimized Technological Mapping.

The constraints file defines the clock period, the rise and fall times, the ramp at the inputs and outputs of the circuit for rising and fall transitions, and the minimum capacitance. The main setting is that of the clock period, as this parameter affects the slack(worst negative setup slack (WNS), total negative setup slack (TNS), worst negative hold slack (WHS), and total negative hold slack (THS)). With this in mind, one must carefully analyze the slack to see if there is a need to change the clock constraints to avoid time violations. A positive slack indicates that constraints have been met.The clock period parameter has been changed to obtain a positive slack value, which is 5 percent more than the clock period value when slack is equal to zero. We adopted this strategy to avoid complications in the physical synthesis stage. After several syntheses, with different clock values, the chosen one was a clock with a period of 51 ns. This synthesis output describes the processor at the gate level and is used in the physical synthesis. Table I shows the clock period values used in several logic syntheses performed to obtain at least 5 percent positive slack.

Table I.: Clock values tested in logic synthesis

clock period (ns)	**slack (ps)**	**rise slew (ps)**	**fall slew (ps)**
2	-1555,7	192.5	111.7
5	0	111.4	84.5
10	0	108.6	82.2
20	0	108.6	82.2
30	101,7	108.6	82.2
40	530,1	108.6	82.2
51	2928	108.6	82.2
72	14232,8	108.6	82.2

C. *Steel Physical Synthesis*

After the logic synthesis, the physical synthesis use as input the netlist generated in the logic synthesis step, which describes the design with the blocks, gates, and logical connections between them. We can divide the physical synthesis into two stages: Floorplanning and Back-end Flow. In the floorplanning step, we obtain the floorplan of the processor, where we aim to minimize the area and delay. The entire physical synthesis flow is done according to [18], with minor tweaks to avail newer tool amenities better.

C..1 Pads: The I/O (input and output) placement consists of placing the pins of the circuit, which is a laborious step when doing the RTL to GDSII flow. Thus, the insertion of pads was performed during the initial stage, because the positions of the pads can influence the posterior routing. The 240 used pads for this layout can be seen in Table II. This design relies on PAD-limited types of pads, due to the high number of inputs and outputs.

Table II.: Used pad cells from the 3.3V I/O Library

Cellname	**Used pads**	**Function**
VDDIPADP	1	V_{DD} Pad
GNDOPADP	1	Ground Pad
CORNERP	4	Corner Pad
ICP	134	Input Pad
BD8P	100	Output Pad
Total	240	

C..2 Floorplaning: With the addition of pads, the floorplanning step consists of the placement of the macros (memory, IPs, etc.) and PADs in the desired places to provide the best circuit performance, whether in timing, power, or area. A square floorplan was generated at this stage, with a density of 70 percent and margins of 15 µm, as shown in Figure 5. The addition of pad fillers during physical synthesis ensures a connection to the entire pad ring. The choice of these numbers was based on tests carried out in the later stages of routing. The distribution of the V_{DD} and GND tracks, both vertically and horizontally, should be as homogeneous as possible to avoid heating points on the chip.

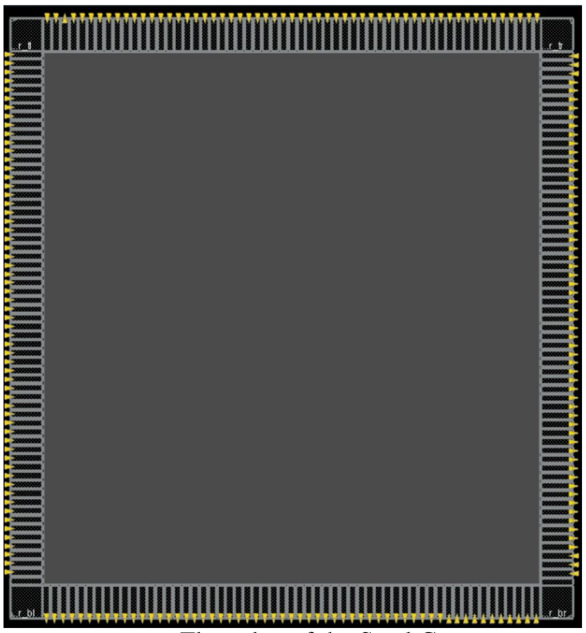


Fig. 5: Floorplan of the Steel Core

C..3 Power Planning: In addition, we defined power planning, which adds VDD and Ground lines and creates a power ring around the core. One of the concerns on the distribution of the metal layers, that generate the circuit supply, is the unwanted generation of parasitic capacitances caused by the proximity of two charged electric conductors that imitate the plates of a capacitor. The implementation with pads shown in Figure 6, displays the supply connections between the pad-ring and the core-ring. The 180 nm technology still uses the intra-cell connections to avoid latch-up. However, this work does not cover the usage of well-taps (special cells for tying the substrate and n-well to V_{DD} and V_{SS}, commonly used in tapless technologies under 65 nm).

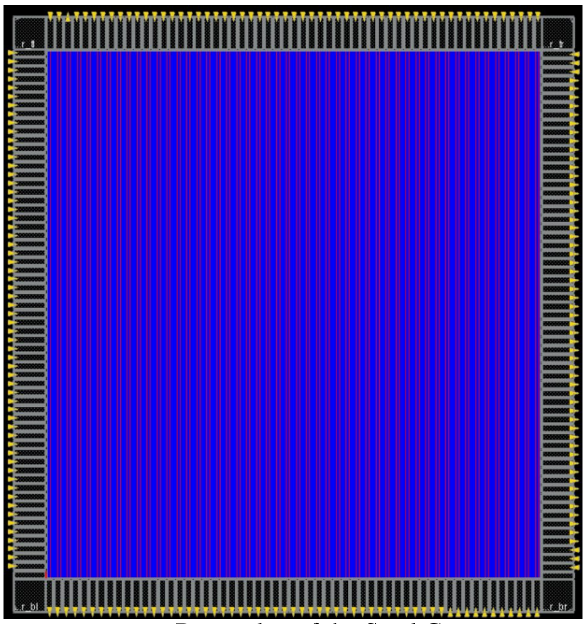
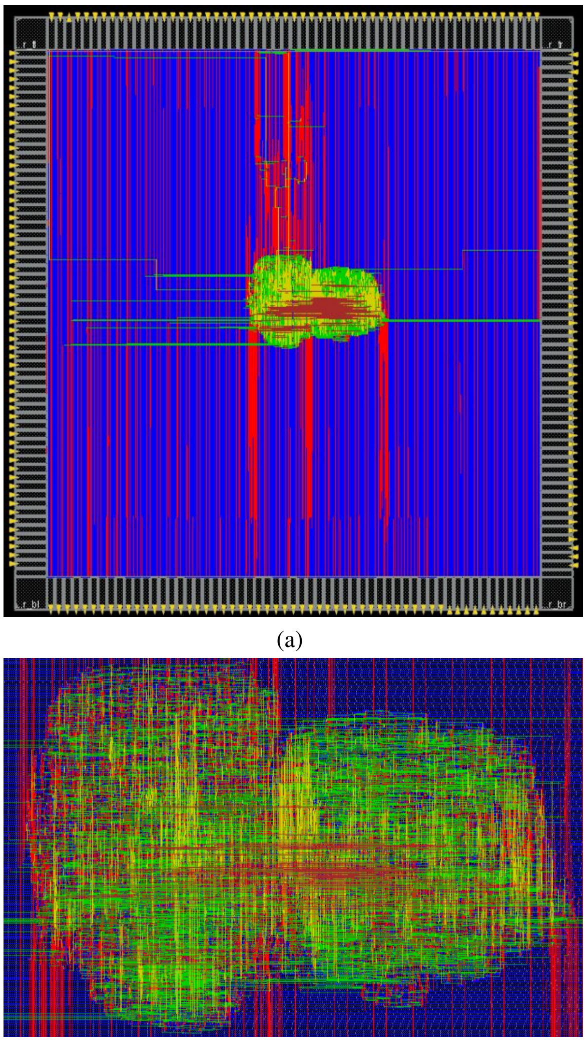


Fig. 6: Powerplan of the Steel Core

C..4 Placement and Routing: In physical synthesis, the process of Placement and Routing of the logic cells is the most time-consuming. The standard cells placement step consists of distributing all of them in the layout core. The routing step refers to the physical interconnection of the standard cells pins through wires using several metal layers. The main concerns in routing is congestion and the average length of connections. In this step, we carry out the placement of the cells together with their pre-routing, or as called by Cadence's tool, a weak routing where the tool is concerned at first with the placement of the cells. Afterward, a cell routing was performed using Nanoroute. Figure 7 shows the circuit rails. The Rail Analysis step consists of verifying the power distribution in the layout to ensure that the supply voltage does IR Drop below a certain level, causing an increase in delays operating standard cells.

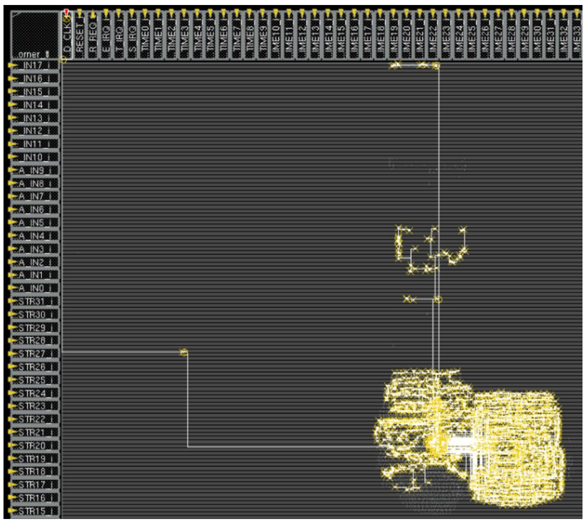
C..5 Clock Tree Synthesis - CTS: The step of synthesizing the clock tree consists of adding buffers in the clock signal paths [19]. Clock distribution is a meticulous task in the synthesis design flow. We work with six layers of metals, where the clock signal starts from the upper pad near the left corner and is distributed to all cells that need it. Thus, the clock skew becomes one of the main concerns: the clock signal must simultaneously reach different components to guarantee the operation's synchronism and correctness. The addition of buffers mitigates the effects caused by different wire lengths. The clock buffers can be used to redistribute path slacks among adjacent timing paths and possibly fix timing violations. This synthesis step is performed with a file of information passed to the tool defining maximum and


(a)

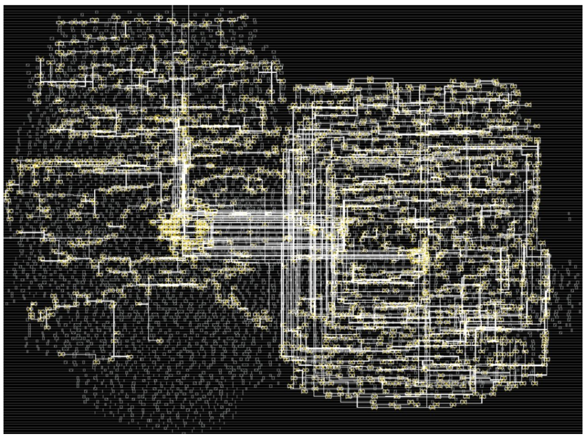
(b)

Fig. 7: Steel Processor Core and Rails. 7(a) with pads and 7(b) zoom of the area with the highest density

minimum delays.



(a)



(b)

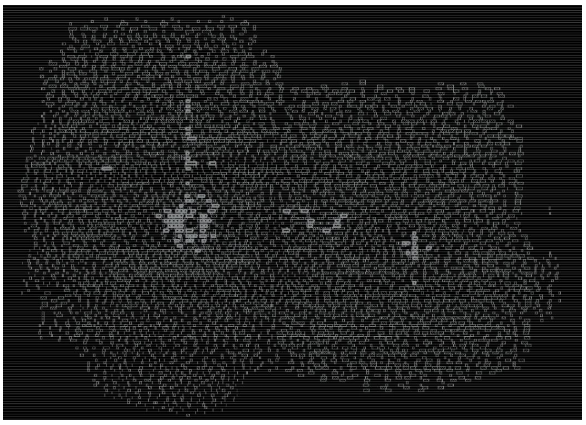
Fig. 8: Post-CTS view of the Steel Core. 8(a) with pads and 8(b) zoom of the central area

Figure 8a gives an overview of the circuit clock distribution as a whole. The clock starts from the upper PAD near the left corner and spreads out over all the cells that need it. Figure 8b shows that not all cells need a clock, as some are designed for sequential logic execution. An optimized clock tree (CT) can help avoiding serious issues (like excessive power consumption, routing congestion, and elongated timing closure phase) further down the flow [20].

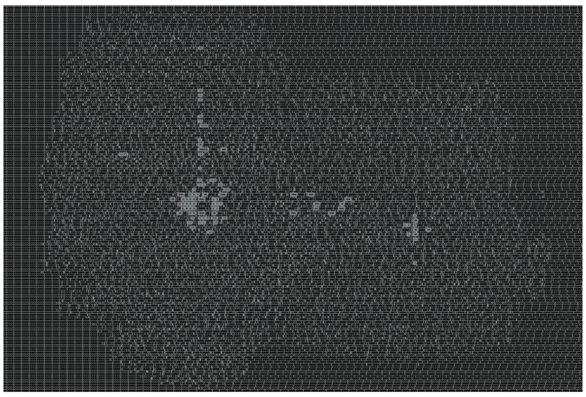
C..6 Filler cells and Metal Fill: Once the design is validated by performing the Design Rules Check (DRC), and thus all reported errors or violations have been resolved, it was possible to insert the padding cells. These cells fill the empty spaces between the already placement standard cells, avoiding planarity problems. Furthermore, they do not alter the functional characteristics of the circuit in any way.

Figure 9a shows only the cell instances so that we can understand the need for filler cells. Note that the project contains "holes" between instances of cells. After inserting the filler cells, we can visually notice the absence of holes between the cells in Figure 9b. Therefore, the design will undoubtedly meet the manufacturing production requirements imposed by the foundry.

The insertion of metal fill has a similar function to that of



(a)



(b)

Fig. 9: View of the Steel Core with and without filler cells. 9(a) pre fillers cells and 9(b) post fillers cells

filler cells: to help to maintain the density of metal layers, during the chemical mechanical planarization (CMP) step of the manufacturing process, with a certain percentage of metal density in each of the used metal layers, according to the foundry specifications [21]. Another concern on what comes to filler insertion is to guarantee that metal fillers are connected to V_{SS}. The metal fill also avoids the likelihood of an antenna effect. Yet, the metal fill is performed to uniform the density of metal on the chip. Thus, it is the last step related to the physical synthesis of the Steel core.

We can see in Figure 10 that the insertion of metal fill is carried out not only in the area between the pads but also in the area referring to the core.

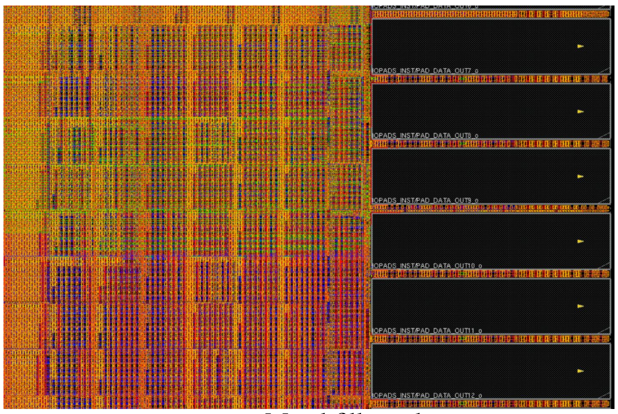


Fig. 10: Metal fill result

V. Results and Discussion

Table III presents the results reported by Innovus after completing the physical design. From it, we get estimates for critical design issues such as timing, power, and area.

Table III.: Result after the physical synthesis

General Design Information	
Design Status	Routed
Design Name	SteelCore top
Instances	231471
Hard Macros	0
Std Cells	231231
Pads	240
Net	7288
Special Net	2
I/O Pins	234
Pins	25234
PG Pins	463661
Average Pins Per Net (Signal)	3.462
General Cell Library Information	
Routing Layers	6
Masterslice Layers	3
Pin Layers	2
Layers	15
Netlist Information	
No of Nets (Int)	7179
No of Connections (Ext)	18242
Floorplan/Placement Information	
Total area of Standard cells	168,81 mm^2
Total area of Core	15.793,46 mm^2
Total area of Chip	20.166,83 mm^2
Core Density (w/Std Cells MACROs)	99.996%
Power Report	
Total Internal Power	7.51769609 mW
Total Switching Power	2.57942532 mW
Total Leakage Power	0.00045145 mW
Total Power	10.09757281 mW

The presented results were obtained right before GDSII file generation, in the late stage of the design flow, and after performing parasitic capacitance extraction and verification such as DRC, process antenna, connectivity, geometry, and metal density. Innovus did not report any errors or alerts after these checkings. The Steel ASIC can reach 19.61 MHz and 10.09 mW as power consumption.For embedded applications, a 19MHz clock speed is in a good range. For example, ATMega328 (Arduino Uno microcontroller) works up to 20MHz. The power consumption (static and dynamic) was estimated using Cadence tools, which provide these informations after completing the physical design, taking into account several factors such as: used PDK characteristics, supply voltage, circuit geometry, and frequency of clock signals. Both in simulation and execution, they were not made using real inputs. However, Cadence tools provide very accurate information that takes into account the variability of the input signals. Dynamic power consumption is also estimated. Power simulation tools nowadays can carry out very accurate calculations. The estimate given by the tool is based on the average power consumption of a cell. It was not possible to make a comparison with [15] implementation because we synthesized our work for ASIC, and the original version of Steel was designed for FPGA.

Nevertheless, it isn't easy to compare the results obtained for Steel regarding parameters such as area, frequency, and power to similar ASIC implementations of RISC-V processor cores. These parameters are heavily dependent on the manufacturing process technology and the used standard cell library. A fair comparison assumes that all variables influencing these parameters are kept constant. We cannot compare our work with PicoRV32, Ibex Core, or Syntacore SCR1 because they use other PDKs.

VI. Conclusions

This paper reports the ASIC synthesis of the Steel processor, describing how the ASIC was designed and obtained results. The estimates from the physical synthesis indicate that the implemented processor has an area around 20.166,83 mm^2, switching and leakage power around 2.58 mW and 0.00045145 mW, respectively, with a clock period of 51ns, which is equivalent to a frequency of 19.61 MHz.

A possible improvement in Steel would be to add new RISC-V extensions. Steel implements RV32I and Zicsr instruction sets, but there are many others: M, A, F, D, Ztso, and more.

Acknowledgements

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) Finance Code 001, the National Council for Scientific and Technological Development – CNPq and Fapergs (Research Support Foundation of the State of Rio Grande do Sul).

References

[1] M. Haselman and S. Hauck, "The future of integrated circuits: A survey of nanoelectronics," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 11–38, 2009.

[2] T. Alsop. (2021, jun) Semiconductor integrated circuits global revenue 2009-2022. [Online]. Available: https://www.statista.com/statistics/519456/forecast-of-worldwide-semiconductor-sales-of-integrated-circuits/

[3] G. D. Hachtel and F. Somenzi, *Logic synthesis and verification algorithms*. Springer Science & Business Media, 2007.

[4] V. d. Santos, F. Petkowicz, R. d. Silva, R. Calçada, and R. Reis, "Design of steel asic, a risc-v processor," *SForum*, 2021.

[5] R. Calçada. (2020, Oct 14) Steel core. [Online]. Available: https://opencores.org/projects/steelcore

[6] RISC-V Foundation, "Risc-v cores and soc overview — RISC-V Foundation Website," https://riscv.org/risc-v-cores/, [Online; accessed 8-May-2020].

[7] C. Celio, D. A. Patterson, and K. Asanovic, "The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167*, 2015.

[8] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, “Chisel: constructing hardware in a scala embedded language,” in *DAC Design Automation Conference 2012*. IEEE, 2012, pp. 1212–1221.

[9] C. Celio, P.-F. Chiu, B. Nikolic, D. A. Patterson, and K. Asanović, “Boomv2: an open-source out-of-order risc-v core,” in *First Workshop on Computer Architecture Research with RISC-V*, 2017.

[10] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, “The Rocket Chip Generator,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 4 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[11] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini, “Pulp: A parallel ultra low power platform for next generation iot applications,” in *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE, 2015, pp. 1–39.

[12] F. Zaruba and L. Benini, “The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, 11 2019.

[13] P. Davide Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini, “Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications,” in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[14] D. Patterson, “Reduced instruction set computers then and now,” *Computer*, vol. 50, no. 12, pp. 10–12, 2017.

[15] R. d. O. Calçada, *Design of Steel: a RISC-V Core*. UFRGS, 2020.

[16] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, “The risc-v instruction set manual. volume 1: User-level isa, version 2.0,” UC Berkeley Dept of EE and CS, Tech. Rep., 2014.

[17] X.-F. S. Foundries. Xc018 cmos data sheet, x-fab semiconductor foundries. [Online]. Available: https://www.xfab.com/technology/cmos/018-um-xc018/

[18] L. Lavagno, L. Scheffer, and G. Martin, *EDA for IC implementation, circuit design, and process technology*. CRC press, 2018.

[19] J.-L. Tsai, L. Zhang, and C. C.-P. Chen, “Statistical timing analysis driven post-silicon-tunable clock-tree synthesis,” in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* IEEE, 2005, pp. 575–581.

[20] D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low power methodology manual: for system-on-chip design*. Springer Science & Business Media, 2007.

[21] Subramanian. (2005) Performance impact from metal fill insertion. [Online]. Available: http://www.axiomweb.co.uk/cadence/MetalFill_Paper.pdf