Comparative Implementation of PicoSoC
System-on-Chip in X-Fab 180 nm CMOS Technology

Rodrigo N. Wuerdig, Leonardo H. Brendler, Cláudio M. Diniz, Ricardo Reis, and Sergio Bampi

Graduate Program in Microelectronics (PGMICRO) - Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre - Brazil
e-mail: rnwuerdig@inf.ufrgs.br

Abstract—This paper presents the physical implementation of the PicoSoC System-on-Chip (SoC) using commercial EDA tools, to use it as a comparison source for future advances in open-source EDA tools for digital implementation flows. The PICORV32 is a simple and versatile microcontroller core that can be used for different applications (e.g., Internet of Things). The whole process entails logical and physical synthesis, design goals aspects, and reports by tools under different working conditions. The Logical and Physical synthesis of the PicoSoC for the X-Fab 180 nm node technology, presented in this work, relies on using the Cadence EDA tools. The microcontroller core was fully synthesized with and without pads, resulting in a SoC that, including the pads, presents an estimated energy consumption of about 695.3 pJ per operation under nominal conditions.

Index Terms—VLSI, RISC-V, SoC, Physical Design, Microelectronics.

I. INTRODUCTION

Considering the high number of transistors on modern circuits, making a full-custom chip is a challenge for today’s technology. To design circuits with a considerably lower Time to Market (TTM) and with a more favorable yield, the majority part of the industry relies on the cell-based design (also called semi-custom design), as shown in Fig.1. The cell-based design consists of a collection of logic cells, with different logical functions and driving strengths, which are called standard cells. Circuits are commonly described with Hardware Description Languages (HDL) or other high-level programming languages and synthesized/mapped to standard cells with the help of Electronic Design Automation (EDA) tools. Circuits composed of standard cells are promising to work since each piece is fully validated.

This study confines its focus to the synthesis of digital components, without encompassing the intricate analog subsystems crucial for optimal system operation, including Phased-Locked-Loops (PLLs), which remain outside the scope of this investigation. The research conducted herein exclusively leverages X-FAB’s XC018 0.18-micron Modular Logic and Mixed-Signal Technology. This technology Process Design Kit (PDK) has been employed to facilitate the digital synthesis processes elucidated within this work.

The PicoSoC utilizes the PICORV32 [1], a very popular open-source softcore. This facilitates the comparison with several different implementations by the open-source hardware community, including physical implementations done in different technology nodes and EDA tools (see Table IV). Despite its popularity, the PICORV32 presents a good balance between power and performance when compared to other RISC-V cores [2]. The PICORV32 is a slick CPU core that features the RV32IMC Instruction set. However, the project can be configured to use other RISC-V instruction sets such as the RV32E, RV32I, and RV32IM.

In addition to the PICORV32 core, the PicoSoC incorporates additional functional blocks, including an Execute-In-Place (XIP) SPI Flash Controller, UART, and Static Random-Access Memory (SRAM). The 32-bit system bus connecting these blocks is versatile, offering the potential for integrating custom blocks beyond the default offerings in the PicoSoC. This adaptability enhances the versatility of the PicoSoC, making it a flexible platform for diverse applications within the open-source hardware ecosystem.

Unlike previous literature using a similar Process Design Kit (PDK) that provided only superficial treatments of these processes, this paper extensively examines the intricacies, methodologies, and challenges involved in achieving a fabrication-ready design for the RISC-V category open RTL code. This comprehensive investigation addresses the gap in understanding, bringing to light essential nuances often overlooked by earlier digital implementation experiments [3, 4]. It’s important to note that there is existing literature on the digital implementation of the same RISC-V core, and a cross-PDK comparison is presented in Section IV.

The structure of this paper is as follows: Section II delves into the intricacies of logic synthesis, Section III explores the nuances of physical synthesis, Section IV presents the key findings, and Section V wraps up the paper with the main observations.

Fig. 1 Representation of a cell-based design.
II. LOGIC SYNTHESIS

Logic Synthesis (LS) is a key step in developing a hardened IP core. Commencing with an initial Hardware Description Language (HDL) file, the central objective of this step is to refine the IP's description, effecting a transformation from the HDL format into a mapped Register Transfer Level (RTL) representation that consists of logic gates (cells). This translation process is facilitated by EDA tools, among which Cadence Genus™, Synopsys Design Compiler™, and YOSYS + ABC are notable examples - the latter being open-source tools widely available. For the specific undertaking detailed in this project, the commercial Cadence Genus™ served as the LS tool. The entire logic synthesis workflow strictly adheres to the methodology prescribed in [5, 6], with minor adjustments introduced to capitalize on the enhanced capabilities offered by more recent versions of these tools.

A. Multi Mode Multi Corner Flow

Multi-Mode Multi-Corner (MMMC) analysis refers to performing Static Time Analysis (STA) across multiple chip operating modes, Process, Voltage, and Temperature (PVT) corners, and parasitic RC interconnect corners at the same time. Genus supports this feature. To enable MMMC, a file should be made describing every operating condition, as listed in table I and pointers to each operating condition in a corresponding liberty file (.lib).

<table>
<thead>
<tr>
<th>Corner</th>
<th>Temperature (°C)</th>
<th>Supply $V_{DD}$ (V)</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst Case</td>
<td>125</td>
<td>1.62</td>
<td>Worst</td>
</tr>
<tr>
<td>Nominal Case</td>
<td>25</td>
<td>1.8</td>
<td>Nominal</td>
</tr>
<tr>
<td>Best Case</td>
<td>-40</td>
<td>1.98</td>
<td>Best</td>
</tr>
</tbody>
</table>

The defined views in table I are set according to their impacts on the circuit’s electrical characteristics. In the worst case, the temperature was set at 125°C to decrease electron mobility at FET strong inversion, while in the best case, the temperature was set to -40°C to increase electron mobility. The -10% nominal $V_{DD}$ in the worst case also harms the circuit delays. For the best case, the circuit is set to +10% of the nominal supply voltage. Process variations are also set under the MMMC views, at the worst process the circuit suffers from all sorts of transistor attribute mutation (e.g., length, width, oxide thickness) to deteriorate performance. In the best view, process variations mutate transistor characteristics in order to improve performance.

MMMC synthesis can further give the possibility to generate reports for every view during logic and physical synthesis. These multi-corner reports can roughly display how the circuit will operate in different conditions. This will also open the possibility of extracting different delay files (.sdf) for further HDL simulation. The netlist generated by the logic synthesis describes an association of library cells that perform what is described on the RTL level before logic synthesis.

B. Clock Gating

Despite the lack of clock-specific cells in the cell library, it was included a command to enable this feature. Clock gating is a technique used in many circuits for reducing dynamic power consumption. Clock gating saves power by adding more logic to a circuit to block the clock tree. Gating the clock disables portions of the circuitry so that the flip-flops in them do not have to switch between states.

C. DFT

Among the techniques for Design for Testability (DFT), two scan chains were defined for the design. These scan chains were automatically connected by the tool and they serve as essential components in the testing and verification of the integrated circuit.

The first scan chain, is associated with a specific test condition related to the falling edge of the test clock. It starts at the pin labeled “DFT_sdi_1” and ends at the pin labeled “DFT_sdo_1”.

The second scan chain, is associated with a different test condition related to the rising edge of the test clock. Similar to the first chain, it starts at the pin labeled “DFT_sdi_2” and ends at the pin labeled “DFT_sdo_2”. Both scan chains are enabled by the same “Scan_Enable” pin.

Overall, scan chains are an integral part of DFT, allowing for efficient testing and validation of the integrated circuit’s functionality. They enable the application of test patterns and the observation of responses, helping to identify and diagnose potential defects or faults in the design.

D. Constraints

For further compatibility with pads and increased probability of functionality, some special constraints are applied to the constraints file. An initial try-out displayed the non-functionality for 200 MHz, so, designers opted for designing the SoC for 100 MHz.

In the context of this design, where the term “fanout” refers to the number of gates or loads/capacitance tied to the output of a particular logic gate, a specific constraint command was employed to limit the fanout to 8. This limitation is crucial for maintaining the reliability and performance of the circuit during logic synthesis. Additionally, to enhance the robustness further, special views were configured for different types of analyses. Specifically, the hold analysis utilized the best view to prevent hold violations, ensuring stable operation, while the setup analysis employed the worst view to mitigate setup violations and uphold the integrity of the design.

III. PHYSICAL SYNTHESIS

The phase of physical synthesis plays a fundamental role in the design process, entailing the transformation of the netlist, typically originating as a Verilog file through logic synthesis, into a spatial arrangement of physically distributed geometries that represent standard cells. The entire physical synthesis procedure adheres to the methodology outlined in [5, 7, 8], with discrete adjustments made to optimize the utilization of contemporary tooling capabilities. These adaptations are undertaken with precision to accommodate the evolving landscape of semiconductor design and ensure compatibility with the latest technological advancements.
A. Pads

Pad insertion is one of the most laborious steps in the RTL to GDSII flow. Most of the difficulty comes with the limitations on the Verilog netlist that should instantiate the logic synthesis output netlist, among pad cells from the I/O (input and output) libraries and connect them mutually. The input netlist for Cadence Innovus™ does not support some constructs (e.g., delays, User-Defined Primitives (UDP), fork and join procedural statements [9]). Furthermore, the process consists basically of creating this Verilog wrapper that inside of it will be instantiated both the instance of the core, macro, and the instance of each pad (since each pad is also a verilog module). This can be quite laborious but it is easily surpassed by using bash commands to support the required manual work.

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

<table>
<thead>
<tr>
<th>Cellname</th>
<th>Function</th>
<th>#Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDDIPADP</td>
<td>$V_{DD}$ Pad</td>
<td>4</td>
</tr>
<tr>
<td>GNDOAPADP</td>
<td>Ground Pad</td>
<td>4</td>
</tr>
<tr>
<td>CORNERP</td>
<td>Corner Pad</td>
<td>4</td>
</tr>
<tr>
<td>ICP</td>
<td>Input Pad</td>
<td>46</td>
</tr>
<tr>
<td>BD8P</td>
<td>Output Pad</td>
<td>82</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>140</strong></td>
</tr>
</tbody>
</table>

In the context of pad selection, it is imperative to conduct a meticulous examination of the available cell types, aligning each choice with the specific requirements of the given task. Table II comprehensively lists the 140 distinct pads employed in this layout. Notably, the design predominantly employs PAD-limited pad types, a deliberate selection driven by the substantial count of inputs and outputs, necessitating a focus on the efficient allocation of physical pad resources. A chip that features pads is commonly classified into two groups: pad-limited or core-limited. A pad-limited chip, see Fig. 2A, is where the total dimensions of the chip are defined by the number of pads. The core-limited, see Fig. 2b, is the opposite case where the total dimension is defined by the core area. For a given chip (that is a square) a pad-limited design will be where: 

$$\#PADS/A * PAD\_PITCH[\mu m] > CORE\_SIDE[\mu m].$$

![Diagram](image)

Fig. 2 Diagram depicting Core- and PAD-Limited designs.

The X-Fab PDK utilized in this work features multiple I/O libraries, the different libraries enable the interface with different external voltage levels (3.3 V and 5.0 V). In this design, the 3.3 V I/O library was employed. Four $V_{DD}$ and four Ground pads were used for better power distribution, even though the total circuit current does not trespass the 25 mA recommended limit for each power pad.

B. Floorplan & Padframe Specification

The initial step in the physical implementation process involves retrieving results obtained from the preceding logic synthesis stage. These files are seamlessly exported within the Cadence toolset to facilitate effective communication between Genus and Innovus. Before formulating the floor plan, the specification of the pad frame assumes significance. The pad frame serves as a blueprint, precisely defining the spatial coordinates for each pad and bonding pad within the chip.

In the context of floorplanning for this design, a crucial requirement is to define a core area and aspect ratio that seamlessly accommodates integration with the memory IP. Subsequently, the SRAM memory macro is subject to a soft placement operation below the core area (see Fig. 3A). This soft placement approach affords the designer the flexibility to make initial approximations regarding its location, subsequently utilizing the snap floorplan option to optimize the alignment of all components within the designated grid. This strategy facilitates efficient spatial organization while considering the specific requirements of the memory IP integration.

The IO cells are like any other standard cells, in this case, we only had access to the Library Exchange Format (LEF) files. Underneath the placed IO Cells coming from the LEF files, there are the padrings. Which are metal rings that go underneath the pads powering the necessary IO circuitry, e.g., Electrostatic discharge (ESD) protections. In order to create a continuous path around the padrings, we need to fill any gaps with IO fillers. In this specific case, the smallest IO filler was allowed to overlap other IO cells.

C. Power Plan

The power plan is an essential part of VLSI physical design flow that focuses on power distribution and management within a chip. The goal of the power plan is to ensure that all the circuit components receive the necessary power, while minimizing power consumption and reducing power supply noise. The power plan design also considers various power domains, where different parts of the chip require different power voltages and currents. In this work, only two power domains were utilized for the core. One called "gnd" was responsible for the ground net and another called "vdd" was responsible for the +1.8V net.

Two separate power rings were created. One for the SRAM memory and another that contours both the memory ring and the core site. Each of them is a pair of rings for the VDD and GND power domains. The power ring for the memory had a width of 20 $\mu m$ and a spacing of 1 $\mu m$ and was created using metals 1 (horizontal) and 2 (vertical). The outer ring had a metal width of 30 $\mu m$ and a spacing of 10 $\mu m$ and was created using metals 3 (horizontal) and 4 (vertical). The route special step connects the instances to the nets that were aliased for each power domain; furthermore, connecting the rails, and pad ring to core ring connections. To decrease the power degradation to the middle parts of the core site, vertical power stripes were added at every
120 μm using thick metal. Those steps can be visualized in Fig. 3B. Even though the usage of well taps inside each logic cell is optional in this technology node, well tap cells were added at every 20 μm for better bulk and well voltage strapping.

D. Placement

The placement step is a critical stage in VLSI physical design that determines the physical location of each logic gate on the chip. The placement process seeks to minimize the wire length and interconnect delay between the gates while also adhering to various design constraints such as area, power, and timing. The placement algorithm typically takes the netlist, which describes the logical connectivity of the gates and generates a layout that maximizes the chip’s performance while meeting the design constraints.

After the coarse placement of standard cells, the spare modules (a set of spare cells) were created and placed. These cells are intended to provide flexibility for the designer to make changes and corrections to the chip. Spare cells can be used to replace a defective cell or to add additional functionality to the chip without having to redesign the entire circuit. The selection of cells to be added as spare cells were based on the number of times each instance was placed. The 4 most used types of cells (DFRQX0, AN22X1, AO22X0, and NA2X1) were selected to compose the spare modules. Ten of these spare modules were placed.

After the coarse placement and the placement of the spare modules, an incremental step was done. During the incremental placement, the tool iterates trying to improve the placement. Fig. 3C displays the layout after the placement. Tie-cells are specialized circuit components used to connect or “tie” the inputs or outputs of a circuit to a known logic state. In the design described herein, two tie-cell types were added. One of the tie-cells is called LOGIC0 and is connected to the ground power domain, i.e., ”gnd”. Another used tie-cell is called LOGIC 1 and it is connected to the +1.8V power domain, i.e., ”vdd”.

E. Clock Tree Synthesis (CTS)

Clock tree synthesis (CTS) focuses on distributing the clock signal to all synchronous elements of a chip. The central goal of CTS is to minimize clock skew, which refers to the variation in arrival times of the clock signal at different elements, while concurrently ensuring that each element receives a stable and adequate clock signal.

CTS achieves this objective through the intricacies of designing a clock tree network, aiming to deliver the clock signal with minimal skew and jitter while adhering to various design constraints. Leveraging advanced techniques such as buffer insertion, wire sizing, and optimization algorithms, CTS endeavors to optimize the clock network, producing high-quality results. It is essential to underscore that the quality of the clock network directly impacts the performance and reliability of the chip, underscoring the critical nature of CTS within the broader landscape of VLSI physical design.

The layout transformation during the clock tree synthesis is visually depicted in Fig. 4, showcasing the state before and after the CTS process. In preparation for CTS, pre-CTS optimizations are employed to rectify any driving strength discrepancies. Subsequently, the clock tree is synthesized in accordance with specified constraints, encompassing parameters such as delay, skew, and frequency. Post-CTS optimizations follow suit, aiming to enhance the CTS process and improve setup time values. This iterative approach ensures that the clock distribution is not only efficient but also aligns seamlessly with the intricate specifications and requirements of the design.

F. Routing

During the routing step, a connection between logic gates through interconnects or wires is established. These interconnects must meet the timing constraints of the design, ensuring a maximum allowable propagation delay while simultaneously minimizing resistance, capacitance, and inductance.
In the context of this project, the tool provides the flexibility of routing in five metal layers, along with the option of incorporating a thick metal layer denoted as "METTP" in the Process Design Kit (PDK). Despite the lower resistance offered by the thick metal layer, it is not utilized in this case due to its higher capacitance per µm, which could potentially result in increased antenna errors.

As illustrated in Fig. 3D, the layout undergoes transformations during the routing stage, encompassing global routing and detailed routing processes. Subsequently, an Engineering Change Order (ECO) routing stage is executed. ECO routing becomes imperative when alterations or corrections are introduced to the design post-initial routing [10]. This iterative process ensures that any modifications made are seamlessly integrated into the existing layout, maintaining the overall integrity of the design while accommodating changes and rectifying errors effectively.

### 4. Fill Insertion & Verification

Filler Cells are a crucial component used to fill up the empty spaces between standard cells on the chip. These fillers ensure that the overall chip layout is more uniform. While fillers may seem like a small and insignificant part of the overall VLSI design process, their importance cannot be understated, as they play a critical role in ensuring that the chip performs as expected and is free from any design-related issues. In fact, the impact on performance and yield is so substantial that several techniques were presented in the literature to counteract them [11, 12].

The implemented design incorporates various types of fillers, each serving distinct purposes. Notably, the inclusion of decoupling capacitors is employed to mitigate fluctuations in the Supply-V<sub>DD</sub> lines, effectively filtering out undesired variations. Additionally, cell fillers are strategically deployed to ensure the continuity of wells and rails within the design. Furthermore, supplementary well taps are introduced to enhance the overall functionality of the design.

To ensure the functionality and correctness of the physical implementation, a series of verification steps were done. These verification steps encompass Design Rule Checking (DRC), Layout vs. Schematic (LVS) comparison, and Logical Equivalence Checking (LEC).

The combination of these verification steps, DRC, LVS, and LEC, helps guarantee the correctness and reliability of the physical design, minimizing the risk of manufacturing defects and ensuring that the chip meets its functional specifications.

### IV. Results and Discussions

This section aims to present and analyze the main results obtained from the physical implementation of the PicoSoC System-on-Chip, which features the popular RISC-V core, PICORV32. In this section, we will discuss the main results of the PicoSoC implementation, such as power consumption, area utilization, and clock frequency. Moreover, we will compare the performance and efficiency of our PicoSoC with other existing SoC implementations of the PICORV32 core, highlighting the strengths and weaknesses of each approach.

The main synthesis results are shown in Table III. Those values were given by logical and physical synthesis tools. Since the circuit was synthesized aiming at the worst corner, the other views are just for analyzing the post-synthesis behavior; moreover, that explains why the area and cell values are the same across the different corner values. The 4× area increase, from the logical to the physical synthesis, is explained by the pad-limited design, where the total die area is limited by the pads and not by the core site itself. One way of reducing such area is to do some kind of multiplexing or by doing a parallel to serial conversion to reduce the number of used pads. Going from the logical to the physical synthesis, displayed a 9 to 10 % increase in power dissipation. Around 40 % of the total power dissipation is being dissipated at the memory macro. Power results are considering the <Tsoc.vcd> file, which contains information about the switching activity coming from the functional verification.

This <Tsoc.vcd> was generated by the HDL simulation of the default test workload that comes with the PICORV32. This workload tests several different operations and instructions (e.g., interruptions, hardware multiplication, and hardware division).

In our study, we observed a notable phenomenon in the behavior of LP cells with respect to VT (threshold voltage). The theory underlying this observation posits that LP cells tend to have a higher VT, which is advantageous in terms of minimizing leakage power. However, when operating at a clock frequency of 100 MHz, a different trend becomes
apparent. At this frequency, the power consumption is predominantly influenced by switching power. As a result, to meet the specified timing constraints, it becomes necessary to incorporate more LP cells into the design while compared to the same design using the SVT cells. To illustrate, the LP implementation includes 41 buffers, whereas the SVT implementation utilizes only 14 buffer cells.

This assertion is supported by the data presented in Table III. Specifically, when examining the "Worst" corner case, we can see that LP cells, whether in logical or physical libraries, indeed exhibit a lower leakage power compared to SVT cells. However, despite this lower leakage power, LP cells consume more total power (Total P.) when operating at 100 MHz.

This trend is consistent across different corners, including the "Nominal" and "Best" cases, indicating that it is a general characteristic of LP cells in this specific context. The need for additional LP cells to compensate for their higher VT at a 100 MHz clock frequency has implications for the overall power consumption and design complexity in integrated circuits.

Furthermore, the decision-making process regarding library selection is a multifaceted one, often contingent on the specific application at hand. A comprehensive understanding of the target application, coupled with a detailed product description, can serve as invaluable guiding factors in making an informed choice.

One consideration in this context is the power profile of the application. Applications vary widely in their power consumption characteristics, with some being predominantly influenced by leakage current, particularly when power gating mechanisms are not employed, and the processing demands remain relatively low. In such scenarios, where static power dissipation is a significant factor, the use of libraries with higher threshold voltages (VT), such as the LP (Low Power) library, may be advantageous. The higher VT in LP cells can effectively reduce leakage power, aligning with the power-efficient requirements of these applications.

Conversely, in applications where dynamic power, stemming from switching activity, takes precedence over leakage power, libraries like SVT (Standard VT) may offer a more suitable choice. SVT libraries, characterized by lower VT values, facilitate faster switching speeds, making them well-suited for applications with high processing demands and frequent state transitions.

Ultimately, the optimal library selection hinges on a delicate balance between factors like power consumption, processing requirements, and the specific constraints of the application. A thorough evaluation of these factors, as well as a detailed product description, plays a important role in making an informed decision that aligns with the desired performance and power efficiency objectives of the chip’s target application.

One of the main goals of this work was to assess the current status of open-source EDA tools and PDKs when compared to the established commercial frameworks; However, as seen in Tab IV the literature still misses works with a detailed description of the results. It is hard to find a work that comprises information such as power dissipation, technology node, area, and target frequency. This lack of information severely affected the comparison. Nonetheless, it is a good advance considering that we can find an SoC that was synthesized using open source synthesis toolchains, as the Raven SoC [13] which used the Qflow [17]. Unfortunately, Raven's SRAM is not compiled using open-source tools, it is an X-Fab IP, as done in the work presented by Sowash [16] which utilizes the openRAM memory compiler [18]. In terms of the µW/MHz our work shows a good performance since the only implementations that surpass its results have a much smaller technology node (such as the predictive FreePDK45nm) or do not comprise the whole System-on-Chip by including the Pads and using a smaller memory.

V. CONCLUSION

This paper has presented an extensive exploration into the physical implementation and comparative assessment of PicoSoC, featuring the PICORV32 RISC-V core. Our methodological approach and the outcomes derived therein substantiate the platform’s commendable energy efficiency and versatile applicability.

Moreover, our investigation has examined the impact of diverse libraries, with particular emphasis on SVT and LP libraries, on the physical design of PicoSoC. Notably, LP cells, characterized by higher VT values, exhibit distinct advantages, primarily associated with the mitigation of leakage power. However, when the clock frequency escalates to 100 MHz, the prevalence of switching power becomes conspicuous. This circumstance necessitates the strategic incorporation of a greater number of LP cells to meet stringent timing requirements.

This empirical observation informs the nuanced trade-offs entailed in library selection, and their ramifications on power consumption and design complexity.

Furthermore, our performance analysis, juxtaposed with other PICORV32-based SoC implementations, serves to illuminate the respective strengths and weaknesses inherent in each approach. The evaluation establishes that PicoSoC attains competitive outcomes across metrics encompassing power consumption, area utilization, and clock frequency. This study posits that our findings can offer valuable guidance for future research endeavors and the evolution of RISC-V-based SoC designs, thereby contributing substantively to the advancement of embedded systems and Internet of Things (IoT) applications.

ACKNOWLEDGMENT

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Table IV. Results comparison.

<table>
<thead>
<tr>
<th>Technology Node [nm]</th>
<th>180</th>
<th>180</th>
<th>?</th>
<th>130</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done w/ Open Tools</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Pad Insertion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Target Freq. [MHz]</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>40</td>
<td>?</td>
</tr>
<tr>
<td>Power [µW/MHz]</td>
<td>691</td>
<td>?</td>
<td>1250</td>
<td>313</td>
<td>?</td>
</tr>
<tr>
<td>Area [mm²]</td>
<td>2.9</td>
<td>3.3</td>
<td>6.5</td>
<td>2.48</td>
<td>0.174</td>
</tr>
<tr>
<td>Memory Type</td>
<td>HP-SRAM</td>
<td>SRAM</td>
<td>SRAM</td>
<td>SRAM</td>
<td>SRAM</td>
</tr>
<tr>
<td>Memory Size [kb]</td>
<td>32</td>
<td>32</td>
<td>512</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
REFERENCES


