PSM Digital – AA 2024/25
Project Report – *SAD Calculation Between Two Images*

Alessio Fontana

(Data: 14 luglio 2025)

Abstract. This report presents the design and implementation of a digital architecture to compute the Sum of Absolute Differences (SAD) between pairs of grayscale images. The architecture is implemented in VHDL and targets FPGA devices. The design has been verified through a comprehensive set of testbenches and synthesized and implemented on a ZyBo Development Board. Results confirm correct functionality, low resource and power usage, and a maximum operating frequency suitable for real-time processing.

Indice

1. INTRODUCTION

The **Sum of Absolute Differences** (SAD) is a simple yet effective metric for comparing blocks of image data. It is widely used in areas such as *motion estimation in video compression* (e.g., MPEG, H.264), *stereo vision for depth estimation*, and *template matching in object detection*. The SAD between two image blocks is computed as the sum of the absolute value of pixel-wise differences.

This project aims to develop a parameterizable, resource-efficient SAD architecture using `VHDL`. The core design processes one pixel per clock cycle in a pipelined manner and accumulates the differences until the entire image is processed. The design supports images of configurable size (N×N) and pixel width, and includes a wrapper that automates the sequential processing of multiple images.

The work includes architectural design, functional verification via simulation, and hardware implementation through synthesis and place-and-route targeting a ZyBo Development Board based on the Xilinx Zynq-7000 All Programmable SoC, using the Vivado Design Suite. Particular attention is paid to reset handling, corner cases, timing closure, and synthesis optimization. The outcome is a robust and flexible SAD computation block ready to be integrated into larger real-time image processing systems.

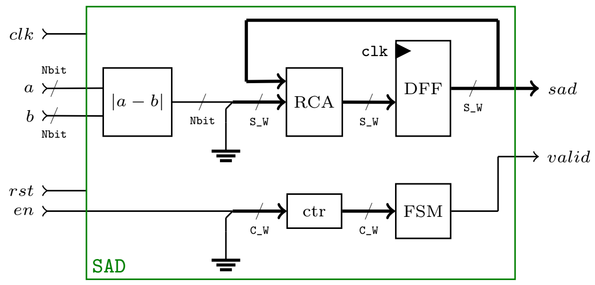

Figura 1 Functional block diagram of the SAD architecture with i/o.

2. ARCHITECTURE DESCRIPTION

Traditional SAD architectures rely on either parallel pixel block processing (high throughput, high area) or serial streaming architectures (low area, moderate throughput). Parallel architectures are suitable for high-speed applications but require a large number of resources. On the contrary, streaming architectures are suitable for real-time image processing with low hardware complexity. Our design follows the latter philosophy.

A high-level diagram of the selected architecture is presented in Fig.1. The design is composed of the following key functional blocks:

1. **I/O registers** for data synchronization. This ensures stable data propagation and avoids timing hazards in the processing path.
2. **Absolute Difference Unit**: computes the absolute value of the difference between each pixel pair. It consists of a subtractor and a comparison logic, and ensures that the result is always non-negative, with no overflow.
3. **Accumulator**: ripple-carry adder (RCA) and a register. At each clock cycle, the current difference is added to the previous accumulation value, stored in the register.
4. **Pixel Counter**: keeps track of the number of pixel pairs received.
5. **Finite state machine** to generate valid signal once the pixel counter reaches the target count.

6. **Top-Level Wrapper** (SAD-Wrapper): instantiates the SAD core to support multiple sequential image comparisons[1].

3. VHDL IMPLEMENTATION

The SAD module was implemented using a behavioral architecture to provide flexibility and clarity in expressing sequential and combinational processes. The architecture is modular and structured around clearly defined functional blocks such as registers, subtractor, adder, and control logic.

Two main processes are used:

- a *combinational process* (`p_diff`) to compute the absolute difference between the input pixels,
- a *sequential process* (`p_FSM`) to handle the generation of the valid signal and reset logic.

The accumulator and pixel counter are instantiated using external configurable components. Signals such as `diff`, `sad_acc`, `next_sad`, `pixel_count` and internal control lines like `done` are used to structure the datapath and synchronization. Constants and generics (e.g., `Nbit`, `N`, `SAD_WIDTH`, `COUNT_WIDTH`) ensure parameterizability for different image sizes and pixel resolutions.

The SAD Wrapper is instead implemented with an RTL architecture, chosen to maintain clear structural mapping between control logic and datapath. A dedicated counter tracks the number of images processed. Generics ensure coherence between the wrapper and the SAD core.

4. VERIFICATION AND TEST PLAN

The verification strategy follows a modular and progressive approach, targeting both functional correctness and robustness under borderline conditions. Three testbenches were developed to validate the system at increasing levels of complexity:

1. **SAD Unit Testbench**: This testbench verifies the core SAD module in isolation. It feeds synthetic pairs of pixels with known differences and checks that the computed SAD matches the expected result. The test covers: pixel-by-pixel accumulation over the full image, enable and reset behavior (beginning and mid-stream), accurate valid signal assertion, expected vs. actual SAD comparison.

[1] Trigger reset between images, current image index counting, final SAD value holding until the next one is ready, i/o registers.

Metric	Value
Target Clock	100 MHz
Worst Negative Slack (WNS)	+1.141 ns
Max Clock	$\sim$ 112.88MHz
Slice LUTs Used	108(0.61%)
Slice FFs Used	131(0.37%)
Total On-Chip Power	0.091W
Junction Temp.	$26.1C$

Tabella I Implementation Results.

2. **Wrapper Testbench**: The second testbench (`SAD_Wrapper_tb`) validates the behavior of the `SAD_Wrapper`, which manages the sequential processing of multiple image pairs. The test includes: streaming of multiple image pairs (same size, predefined difference) separated by three clock cycles (reset time), streaming of pseudorandom images (variables used here) with expected SAD calculation.
3. **Large Image Testbench**: The third testbench (`SAD_Wrapper_tb_large`) focuses on corner cases related to high-resolution input and overflow. It tests the system with: larger images (e.g., 128×128 pixels), verification of correct accumulator sizing to avoid overflow, analysis of timing stability during long input streams, automatic comparison between the computed SAD and the expected result[2].

4.1. Overflow Risk Mitigation

The accumulator width (`SAD_WIDTH`) is calculated based on the theoretical maximum SAD value ($(2^{Nbit} - 1) \cdot N^2$). The test confirms that no overflow occurs under worst-case conditions.

The simulation waveforms have also been inspected manually to verify timing behavior and signal synchronization. This comprehensive test plan ensures both functional correctness and resilience under realistic and worst-case operating conditions.

5. SYNTHESIS AND IMPLEMENTATION RESULTS

The synthesis and implementation were carried out using the Vivado Design Suite in Out-Of-Context (OOC) mode. The implementation results are summarized in Tab.I, which reports the maximum clock frequency, resource utilization and estimated power consumption.

[2] Any mismatch is detected and reported during simulation, ensuring reliable validation even for high-resolution input cases.

The critical path analysis shows a Worst Negative Slack (WNS) of +1.141ns, relative to a target clock period of 10 ns (100 MHz). This indicates that the design meets timing comfortably and can operate at a higher frequency.

The timing analysis reveals that the critical path starts from the register storing input vector A (`regA`) and ends at the final accumulation register `sad_reg`. This path spans the entire computation pipeline of the SAD unit, including the absolute difference computation, accumulation, and final result storage. The path delay is 8.893ns. Based on the WNS, the estimated maximum achievable clock frequency is approximately 112.88 MHz.

5.1. Warning Messages

During synthesis and implementation, some warning messages were issued by Vivado. These include:

1. **Design too small warnings**: These warnings indicate that the design does not utilize a significant portion of the FPGA fabric. This is expected, as the SAD module implements a simple and narrowly scoped function. Such warnings are informational and do not point to any error or inefficiency in the design.

2. **OOC mode and PS7 warnings**: Since the module was synthesized in Out-Of-Context mode, Vivado issued a warning regarding the absence of the Processing System 7 (PS7) block and several related warnings. This is normal in OOC flows, where the PS7 is not instantiated. The warning can be safely ignored during module-level synthesis. In the complete system, however, the PS7 is required to provide clock and reset signals, and must be properly configured at the top level.

No critical warnings were reported. A minor warning about a removed sequential register was due to optimization during logic trimming and was later fixed in the VHDL source.

6. CONCLUSION

The implemented SAD (Sum of Absolute Differences) module demonstrates how a simple yet fundamental image processing algorithm can gain efficiency when carefully implemented in hardware. Despite its compact footprint and minimal resource usage, the design achieves a high maximum clock frequency (over 110 MHz), making it suitable for real-time applications such as object tracking, stereo vision, or motion estimation on embedded systems.

The low power consumption and ease of integration into a Zynq-based SoC platform further highlight its practicality for edge computing scenarios. Additionally, the modular and scalable architecture allows for straightforward extension to larger image sizes or parallel processing for higher throughput.