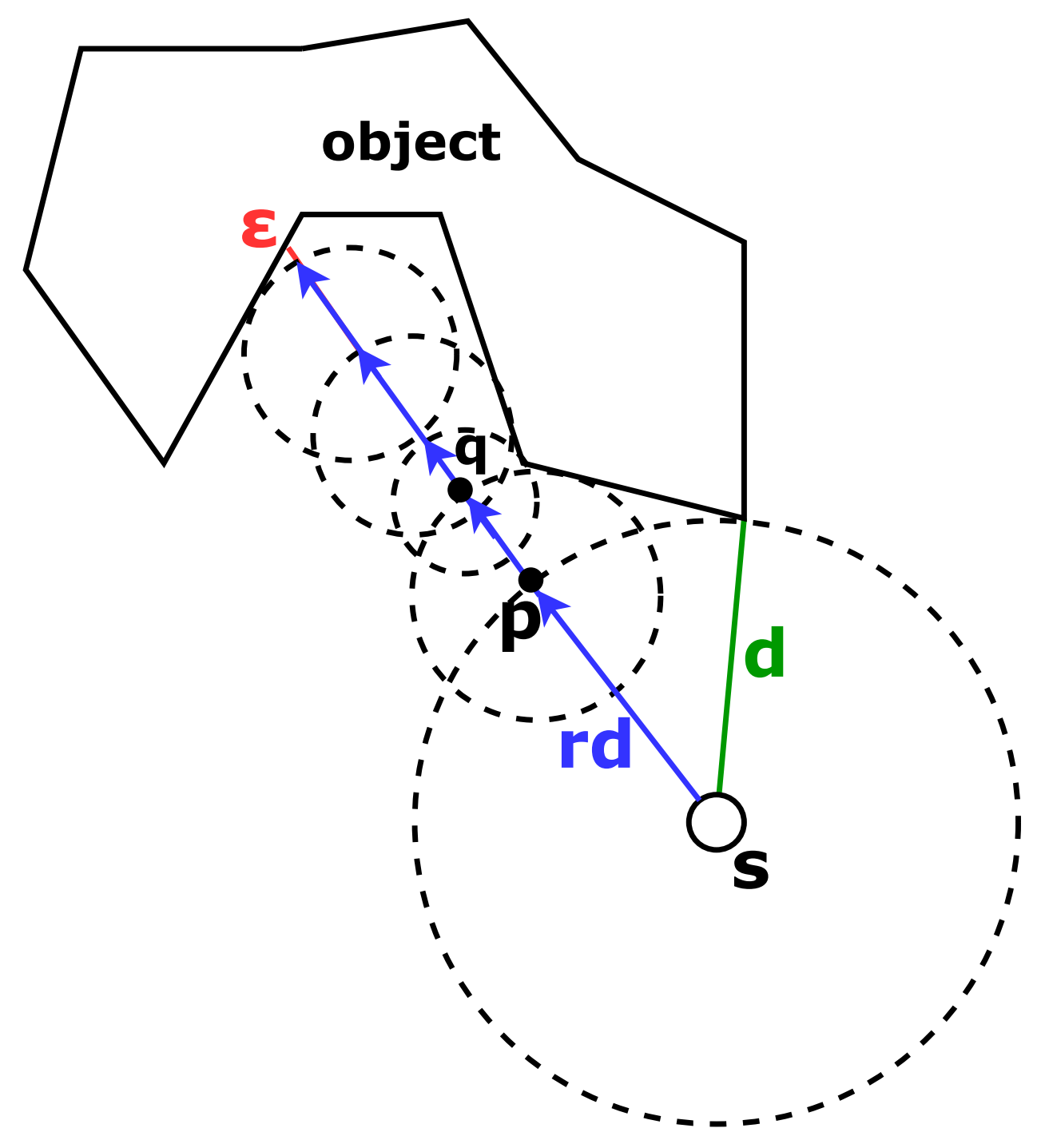
FPGA Ray Marching: 3D CORDIC for Extreme Mandelbulb Precision

Kevin Klein
Kevin.Klein@stud.uni-heidelberg.de, Institute of Computer Engineering, Heidelberg University

Motivation

Why hardware ray marching? Conventional GPUs struggle when extreme precision is required: high-detail fractals like the Mandelbulb[3] demand accurate trigonometric and magnitude operations that either saturate floating-point precision or incur large performance penalties. Ray marching visualises implicit surfaces by stepping along a ray until it approaches the signed distance function (SDF). Hardware implementations can use fixed-point arithmetic and deep pipelining to sustain interactive rates.

- High-precision arithmetic is necessary to explore deeply zoomed fractals.
- GPUs offer limited precision and poor energy efficiency for these workloads.
- FPGAs provide scalable precision and deterministic latency using shift-add algorithms.

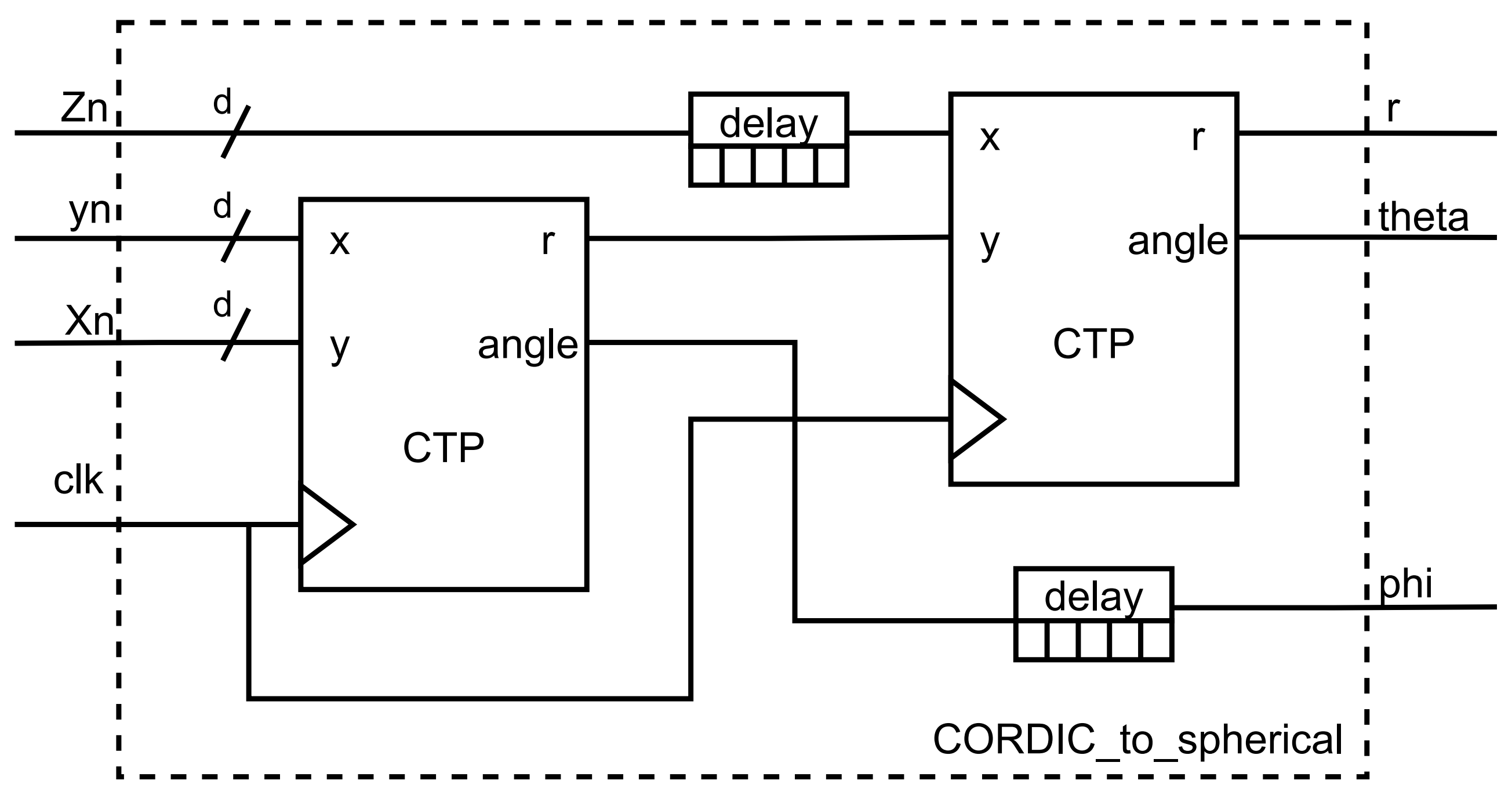


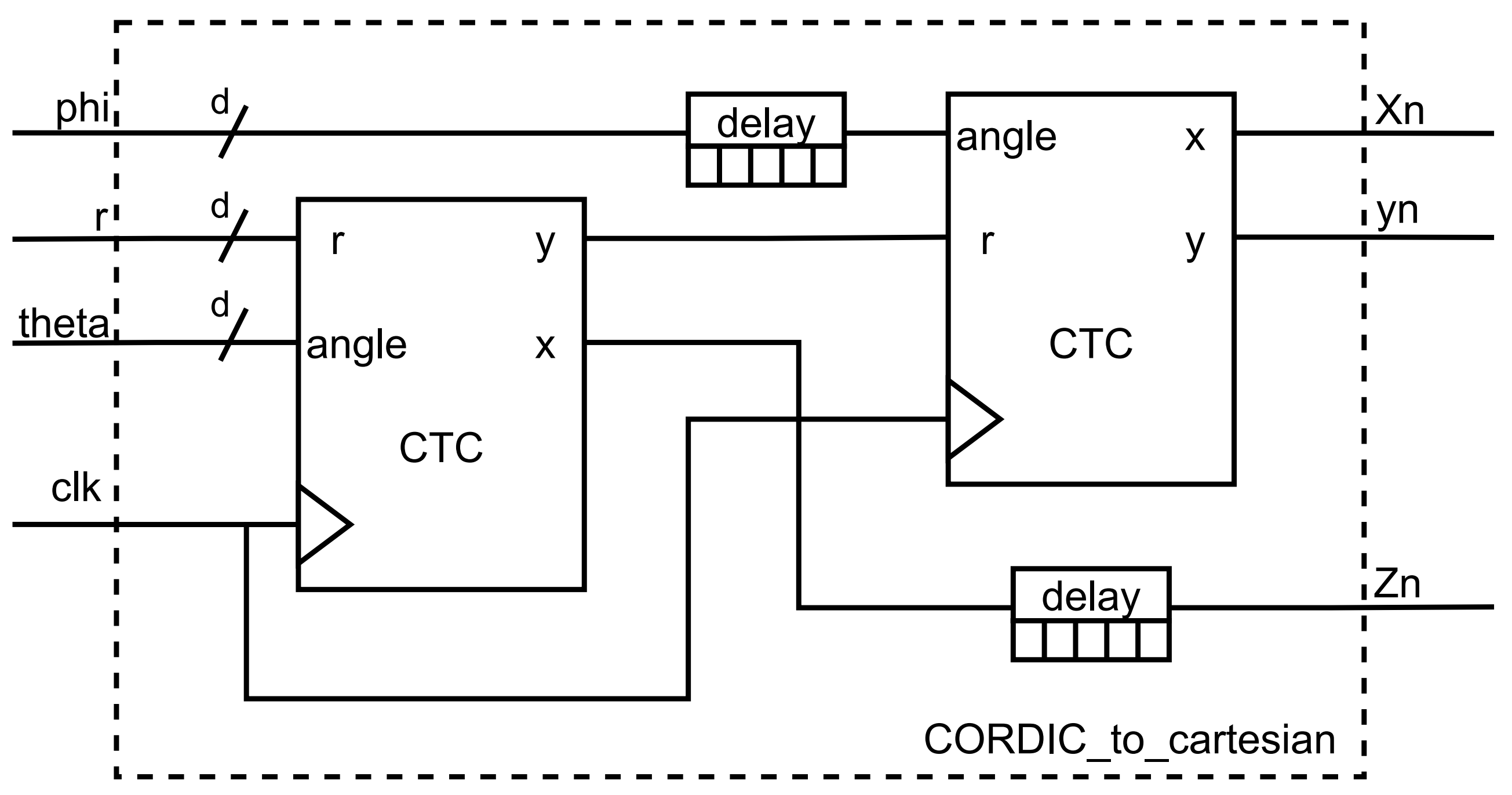
Contributions

- **3D CORDIC core:** A fully pipelined extension of the CORDIC algorithm capable of converting Cartesian coordinates to spherical coordinates and back using only shift–add operations. Precision scales linearly with the number of iterations.
- **Hardware ray marcher:** A generic architecture that reuses an SDF block within a data-flow loop, injecting new rays as soon as previous rays finish and supporting dynamic step sizes based on the current distance.
- **Mandelbulb SDF pipeline:** An unrolled and deeply pipelined (more than 1000 pipeline stages) implementation of the Mandelbulb SDF using the 3D CORDIC core together with the BKM[2] algorithm for logarithms and CORDIC division for reciprocals.
- **Evaluation:** Implementations on Artix-7 and VU9P FPGAs are compared against an RTX 3080 GPU, showing superior precision and energy efficiency with comparable frame rates.

3D CORDIC Core

The CORDIC algorithm computes rotations and vector magnitudes using only shifts and additions[1]. To operate in three dimensions, two 2D CORDIC stages are cascaded: first $(x, y) \rightarrow (\rho, \varphi)$, then $(z, \rho) \rightarrow (r, \theta)$. An inverse stack transforms spherical back to Cartesian coordinates. The pipeline depth equals the number of iterations, allowing precision to scale as needed.

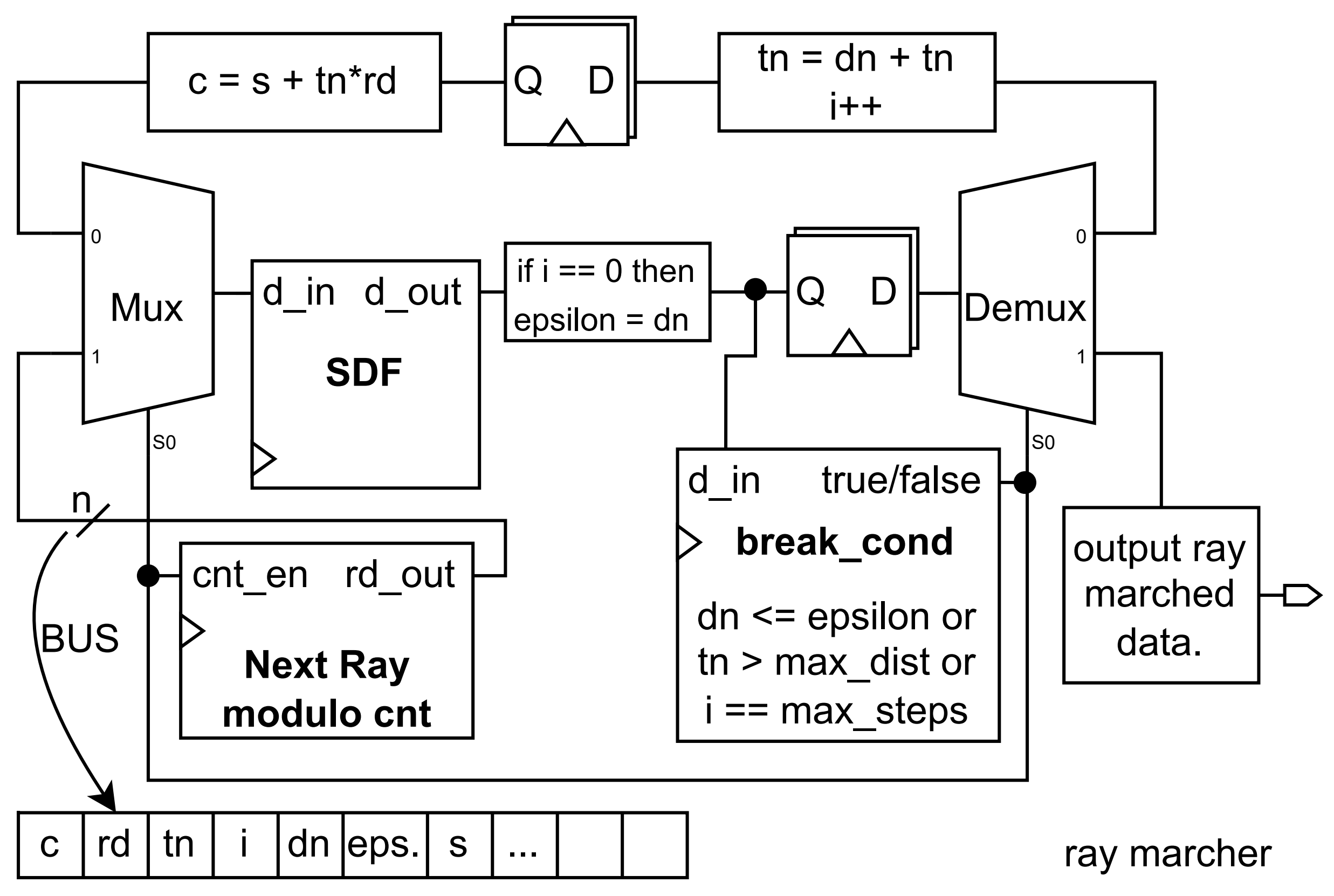




Hardware Ray Marcher

The hardware ray marcher approximates the intersection of a ray with a surface by iteratively accumulating distances from the SDF. Given a starting point s and unit direction rd, the accumulated length t_n is updated by $d_n = \mathrm{SDF}(s + t_n\, rd)$ until $d_n < \varepsilon$ or a maximum travel distance is reached. Our architecture implements this loop as a streaming pipeline without idle cycles.

- A generic, pipelined SDF core accepts a 3D coordinate and returns the distance to the nearest surface.
- A multiplexer/demultiplexer pair recirculates data through the SDF until the break condition is met, then injects the next ray.
- Dynamic step size is achieved by setting $\varepsilon = d_n \gg b$, where b controls aliasing at deep zoom.



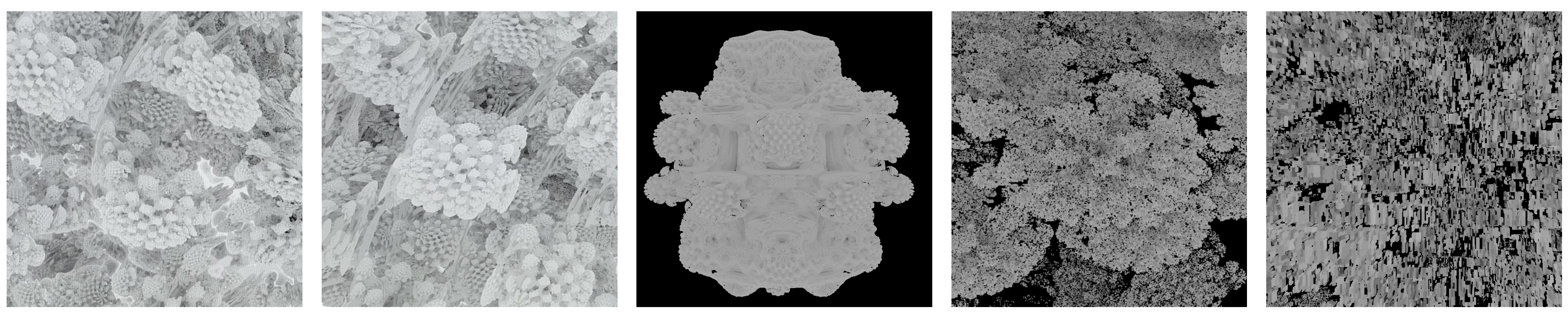
References

[1] Ray Andraka. A survey of cordic algorithms for fpga based computers. In *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on FPGAs*, page 191–200, 1998.

[2] J.-C. Bajard, S. Kla, and J.-M. Muller. Bkm: a new hardware algorithm for complex elementary functions. *IEEE Transactions on Computers*, 43(8):955–963, 1994.

[3] Inigo Quilez. Mandelbulb, 2009. `https://iquilezles.org/articles/mandelbulb`.

Results and Comparison



Implementations on the Artix-7 `XC7A200T` were compared against a GLSL implementation running on an NVIDIA RTX 3080. The FPGAs achieve higher precision while matching or exceeding the frame rates of the GPU (VU9P) at a fraction of the power. The first two images show FPGA results at increasing zoom levels, the last two depict GPU renderings where precision begins to degrade. The middle image is the whole Mandelbulb rendeerd on the Artix-7 `XC7A200T` FPGA with 5 fps.