Comparing High-Level Synthesis and RTL for Simple FPGA Kernels – Revisited

Vinicius Ramalho Fertrin • Christopher D. Gill • Roger D. Chamberlain
Washington University in St. Louis

How close can current HLS get to hand-tuned RTL?

Straightforward HLS tends to save area for simple kernels, which increases the resulting latency. Explicit loop unrolling can recover latency at significant LUT cost, while a more arithmetic-heavy FIR filter was the most favorable case for HLS.

Why revisit HLS vs RTL comparisons?

- HLS improves productivity, but QoR relative to optimized RTL is still unsettled
- Many prior comparisons use large applications, making the architectural causes hard to isolate
- Simple kernels expose what the tool does when the structure is easy to reason about

Experimental Setup:

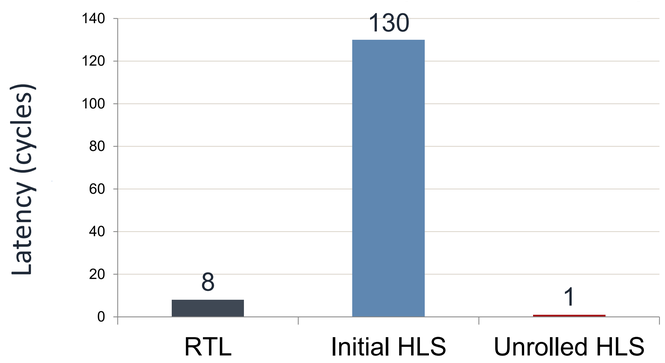
- Target FPGA: Artix-7
- Tool flow: AMD Vivado and Vitis HLS 2025.2
- Metrics: latency, initiation interval (II), Fmax, LUT, FF, DSP
- RTL, initial HLS, partially and fully unrolled HLS

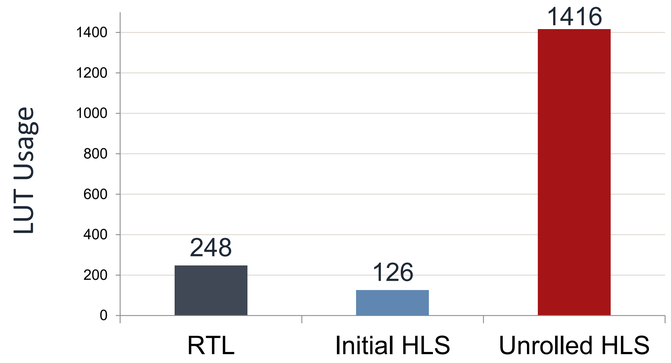
Conclusions:

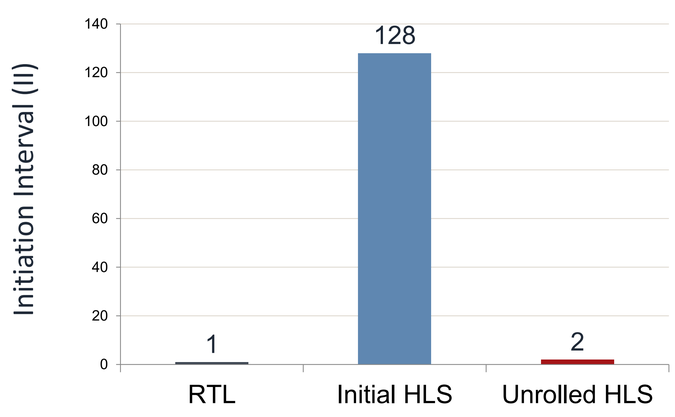
- Straightforward HLS seems to favor compact area over low latency and II
- For popcount and LZC, explicit RTL control still delivers better performance-resource balance
- Loop unrolling closes the latency gap, but usually with LUT overhead and mixed Fmax impact

• Popcount

RTL: Fully pipelined adder tree
HLS: Simple accumulation loop





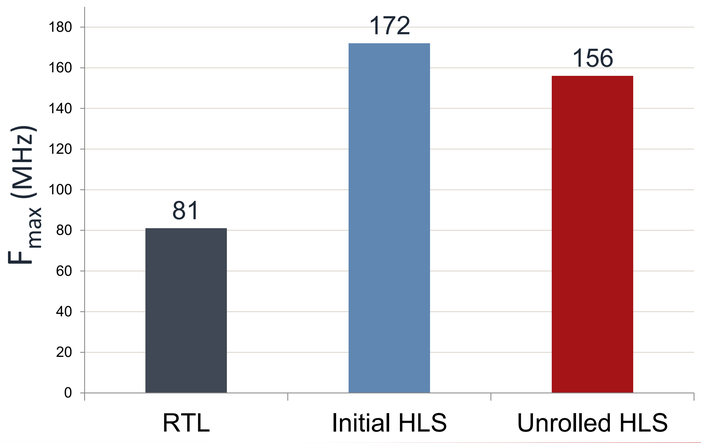


• Partial Unrolling

LZC

Implementation	Latency (cycles)	II (cycles)	LUT	FF	DSP	F_{max} (MHz)
RTL	8	1	457	766	0	594
Initial HLS	133	134	283	160	0	143
Unrolled HLS	3	4	1,090	147	0	143
Factor 2 HLS	69	70	412	179	0	105
Factor 4 HLS	37	38	570	181	0	104
Factor 8 HLS	21	22	886	183	0	103
Factor 16 HLS	13	14	1,566	185	0	100
Factor 32 HLS	10	10	2,858	187	0	91
Factor 64 HLS	8	8	5,390	189	0	91

• FIR Filter



- Initial HLS includes a DSP block
- HLS LUT usage is lower than RTL post-synthesis
- Unrolled version reaches RTL-like latency with lower LUT cost

Frequency estimated from WNS for RTL and recorded from Vitis synthesis results for HLS

Future Work:

- Explore more HLS directives, such as array partitioning
- Memory-intensive kernels, different access patterns, and more complex control logic
- Clarify why certain microarchitectures are chosen under different unroll factors