M2 : Integration Circuits Systems

Design of generic square root computing architectures on FPGA using VHDL

Advanced Digital Electronics -ADE-

Developed by:
Sara AGIB
Said AGOUZAL

Academic year: 2025/2026

Contents

1 Architecture 1

The VHDL file implementing this architecture is named **Newton_a1.vhd**.

1.1 Sythesis results

Flow Summary	
Flow Status	Successful - Mon Dec 15 17:20:06 2025
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	sqrt
Top-level Entity Name	newton_a1
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	4,513 / 18,752 (24 %)
Total combinational functions	4,483 / 18,752 (24 %)
Dedicated logic registers	68 / 18,752 (< 1 %)
Total registers	68
Total pins	100 / 315 (32 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

(a) Nb of logical elements

Slow Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-414.686	X_reg[60]	X_reg[8]	clk	clk	1.000	0.006	415.730
2	-414.685	X_reg[60]	X_reg[14]	clk	clk	1.000	0.008	415.731
3	-414.685	X_reg[60]	X_reg[13]	clk	clk	1.000	0.008	415.731
4	-414.685	X_reg[60]	X_reg[12]	clk	clk	1.000	0.008	415.731
5	-414.685	X_reg[60]	X_reg[11]	clk	clk	1.000	0.008	415.731
6	-414.685	X_reg[60]	X_reg[10]	clk	clk	1.000	0.008	415.731
7	-414.685	X_reg[60]	X_reg[7]	clk	clk	1.000	0.008	415.731
8	-414.662	X_reg[60]	X_reg[18]	clk	clk	1.000	0.000	415.700
9	-414.662	X_reg[60]	X_reg[6]	clk	clk	1.000	0.000	415.700
10	-414.660	X_reg[60]	X_reg[31]	clk	clk	1.000	0.007	415.705
11	-414.660	X_reg[60]	X_reg[30]	clk	clk	1.000	0.007	415.705
12	-414.660	X_reg[60]	X_reg[29]	clk	clk	1.000	0.007	415.705
13	-414.660	X_reg[60]	X_reg[28]	clk	clk	1.000	0.007	415.705
14	-414.660	X_reg[60]	X_reg[25]	clk	clk	1.000	0.007	415.705
15	-414.660	X_reg[60]	X_reg[24]	clk	clk	1.000	0.007	415.705
16	-414.660	X_reg[60]	X_reg[23]	clk	clk	1.000	0.007	415.705
17	-414.660	X_reg[60]	X_reg[26]	clk	clk	1.000	0.007	415.705
18	-414.653	X_reg[60]	X_reg[20]	clk	clk	1.000	0.005	415.696
19	-414.653	X_reg[60]	X_reg[19]	clk	clk	1.000	0.005	415.696
20	-414.650	X_reg[61]	X_reg[8]	clk	clk	1.000	0.006	415.694

(b) critical paths

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	2.41 MHz	2.41 MHz	clk	

(c) Maximal frequency supported by the architecture

Figure 1: Results of synthesis for Newton architecture for cyclone II target

- Architecture 1 implements the square root computation using the Newton–Raphson iterative method, performing one full Newton iteration per clock cycle. In this architecture, executing a 64-bit division and a 64-bit addition within a single clock cycle results in a very long combinational path and a high utilization of logic resources. Consequently, the design uses 4,513 logic elements and 68 registers (`X_reg` and some signals), and achieves a very low maximum operating frequency of 2.41 MHz. This limitation is mainly due to the critical path at `X_reg`, which corresponds to the register receiving the output of the 64-bit divider, leading to a dominant delay of approximately 413 ns. In addition to the long clock period, the stopping condition of the iteration is not clearly defined. Overall, this architecture exhibits poor operating frequency and relatively high area due to the heavy combinational arithmetic involved.

1.2 Simulation Results

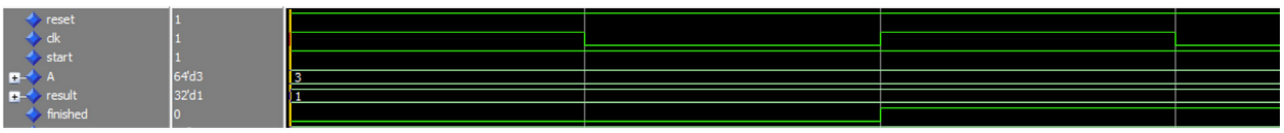

Figure 2: RTL simulation of the computation of the square root of 3

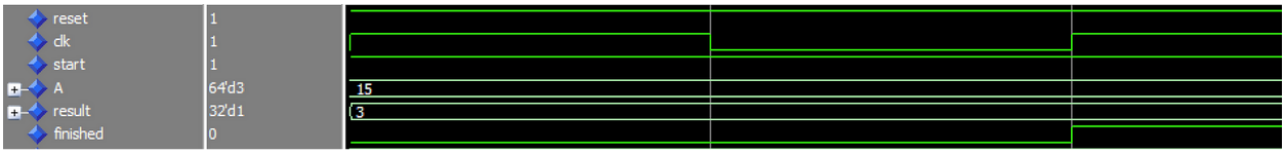

Figure 3: RTL simulation of the computation of the square root of 15

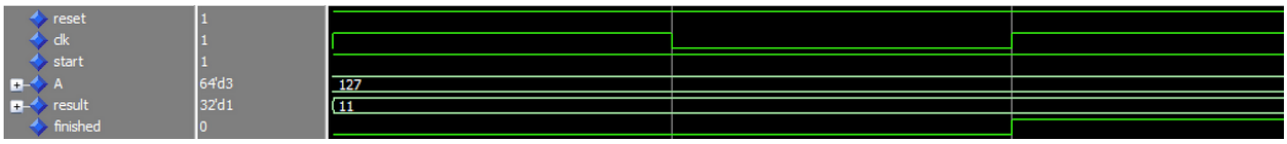

Figure 4: RTL simulation of the computation of the square root of 127

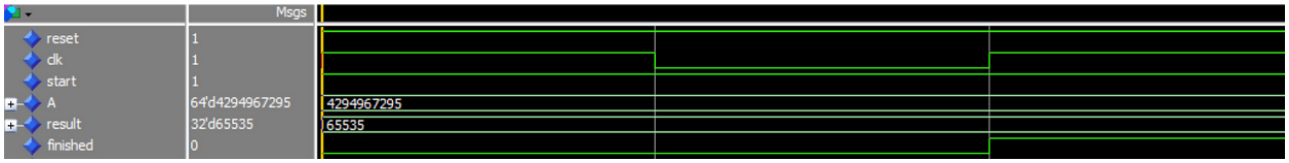

Figure 5: RTL simulation of the computation of the square root of 4.294.967.295

- From the simulation results, it can be observed that the output value is already valid while the `finish` signal is asserted one clock cycle later. This behavior is due to the fact that the proposed architecture does not have a precise stopping criterion and therefore always computes an additional iteration to determine whether the current result has converged. **Note: the square root of zero should not be computed, as it is a trivial case and not supported by this implementation.**

2 Architecture 2

The VHDL file implementing this architecture is named **itera_sqrt_a1.vhd**.

2.1 Sythesis results

Flow Summary	
Flow Status	Successful - Mon Dec 15 17:12:02 2025
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	sqrt
Top-level Entity Name	it_sqrt
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	309 / 18,752 (2 %)
Total combinational functions	308 / 18,752 (2 %)
Dedicated logic registers	138 / 18,752 (< 1 %)
Total registers	138
Total pins	100 / 315 (32 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

(a) Nb of logical elements

Slow Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-7.649	Z[0]	R[32]	clk	clk	1.000	0.000	8.687
2	-7.648	Z[0]	Z[0]	clk	clk	1.000	0.000	8.686
3	-7.611	Z[0]	R[30]	clk	clk	1.000	-0.002	8.647
4	-7.417	Z[5]	R[30]	clk	clk	1.000	0.002	8.457
5	-7.362	R[0]	R[32]	clk	clk	1.000	0.007	8.407
6	-7.361	R[0]	Z[0]	clk	clk	1.000	0.007	8.406
7	-7.345	R[1]	R[30]	clk	clk	1.000	0.002	8.385
8	-7.324	R[0]	R[30]	clk	clk	1.000	0.005	8.367
9	-7.308	Z[1]	R[30]	clk	clk	1.000	0.002	8.348
10	-7.273	R[2]	R[30]	clk	clk	1.000	0.003	8.314
11	-7.232	Z[2]	R[30]	clk	clk	1.000	0.002	8.272
12	-7.220	Z[0]	R[29]	clk	clk	1.000	0.001	8.259
13	-7.200	Z[6]	R[30]	clk	clk	1.000	0.002	8.240
14	-7.168	R[3]	R[30]	clk	clk	1.000	0.003	8.209
15	-7.158	R[2]	R[32]	clk	clk	1.000	0.005	8.201
16	-7.157	R[2]	Z[0]	clk	clk	1.000	0.005	8.200
17	-7.153	Z[3]	R[30]	clk	clk	1.000	0.002	8.193
18	-7.151	Z[0]	R[28]	clk	clk	1.000	0.001	8.190
19	-7.136	Z[5]	R[29]	clk	clk	1.000	0.005	8.179
20	-7.111	Z[4]	R[30]	clk	clk	1.000	0.002	8.151

(b) critical paths

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	115.62 MHz	115.62 MHz	clk	

(c) Maximal frequency supported by the architecture

Figure 6: Results of synthesis for iterative architecture for cyclone II target

- Architecture 2 implements an iterative square-root algorithm that computes one bit of the result per clock cycle. The design relies on three main registers: D, which stores a shifted version of the input value, Z, which holds the partial square root, and R, which contains the partial remainder. All computations are performed in the `S_COMP` state, while a counter increments from 0 to `NBITS`-1. When the counter reaches 31 (for `NBITS` $= 32$), the finite state machine transitions to the `S_FIN` state and asserts the `finished` signal.

  Since each iteration involves only simple operations such as single-bit shifts and additions, the combinational logic per clock cycle is very short, allowing the design to operate at a much higher frequency. Unlike Architecture 1, this approach avoids large divisions and complex 64-bit arithmetic within a single cycle, resulting in a very short critical path. Consequently, Architecture 2 achieves a high maximum clock frequency of approximately 115.62 MHz, at the cost of requiring 32 clock cycles to complete the computation.

2.2 Simulation Results

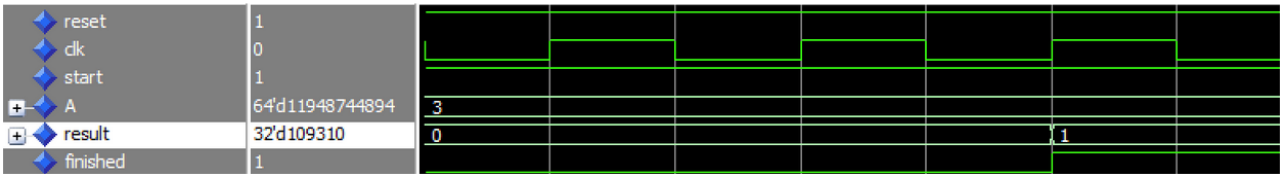

Figure 7: RTL simulation of the computation of the square root of 3

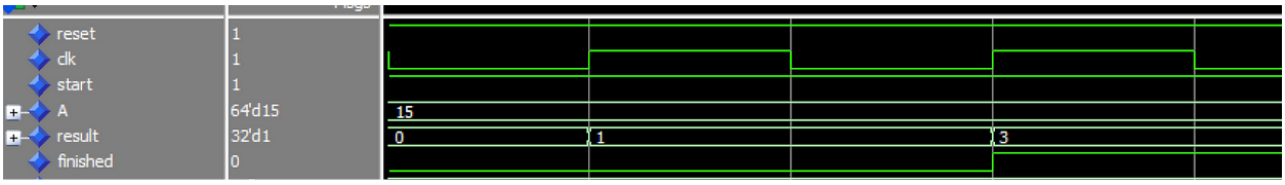

Figure 8: RTL simulation of the computation of the square root of 15

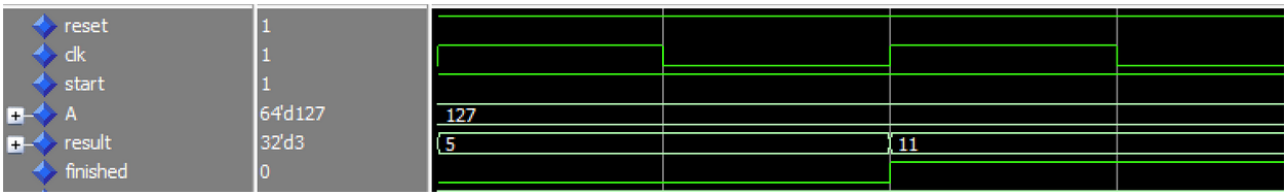

Figure 9: RTL simulation of the computation of the square root of 127

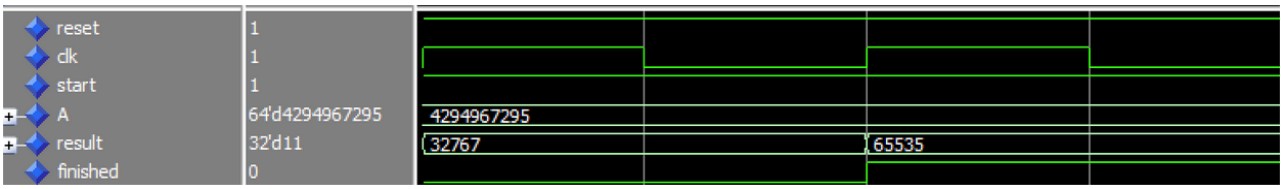

Figure 10: RTL simulation of the computation of the square root of 4.294.967.295

- It can be observed that, unlike the Newton-based architecture, the iterative architecture asserts the `finished` signal immediately when the result becomes valid. Additionally, this architecture can correctly compute the square root of zero without any issues.

3 Architecture 3

The VHDL file implementing this architecture is named **itera_sqrt_a2.vhd**.

3.1 Synthesis Results

Flow Summary	
Flow Status	Successful - Mon Dec 15 17:09:32 2025
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	sqrt
Top-level Entity Name	it_sqrt
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	4,503 / 18,752 (24 %)
Total combinational functions	4,458 / 18,752 (24 %)
Dedicated logic registers	97 / 18,752 (< 1 %)
Total registers	97
Total pins	100 / 315 (32 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

(a) Nb of logical elements

Slow Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-228.068	reg_A[62]	result[0]~reg0	clk	clk	1.000	-0.006	229.100
2	-227.694	reg_A[63]	result[0]~reg0	clk	clk	1.000	-0.006	228.726
3	-227.273	reg_A[61]	result[0]~reg0	clk	clk	1.000	-0.006	228.305
4	-227.130	reg_A[60]	result[0]~reg0	clk	clk	1.000	-0.006	228.162
5	-223.499	reg_A[58]	result[0]~reg0	clk	clk	1.000	-0.001	224.536
6	-223.111	reg_A[59]	result[0]~reg0	clk	clk	1.000	-0.001	224.148
7	-220.587	reg_A[62]	result[1]~reg0	clk	clk	1.000	-0.027	221.598
8	-220.213	reg_A[63]	result[1]~reg0	clk	clk	1.000	-0.027	221.224
9	-219.792	reg_A[61]	result[1]~reg0	clk	clk	1.000	-0.027	220.803
10	-219.649	reg_A[60]	result[1]~reg0	clk	clk	1.000	-0.027	220.660
11	-218.245	reg_A[56]	result[0]~reg0	clk	clk	1.000	0.004	219.287
12	-217.845	reg_A[57]	result[0]~reg0	clk	clk	1.000	0.013	218.896
13	-216.018	reg_A[58]	result[1]~reg0	clk	clk	1.000	-0.022	217.034
14	-215.630	reg_A[59]	result[1]~reg0	clk	clk	1.000	-0.022	216.646
15	-212.847	reg_A[62]	result[2]~reg0	clk	clk	1.000	-0.002	213.883
16	-212.473	reg_A[63]	result[2]~reg0	clk	clk	1.000	-0.002	213.509
17	-212.052	reg_A[61]	result[2]~reg0	clk	clk	1.000	-0.002	213.088
18	-211.919	reg_A[54]	result[0]~reg0	clk	clk	1.000	0.006	212.963
19	-211.909	reg_A[60]	result[2]~reg0	clk	clk	1.000	-0.002	212.945
20	-211.780	reg_A[55]	result[0]~reg0	clk	clk	1.000	0.006	212.824

(b) critical paths

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	4.37 MHz	4.37 MHz	clk	

(c) Maximal frequency supported by the architecture

Figure 11: Results of synthesis for combinational iterative architecture for cyclone II target

- Architecture 3 implements the same square-root algorithm as Architecture 2; however, in this case, the entire algorithm is realized using purely combinational logic. Within the `COMPUTATION` process, the arrays D, R, and Z store the intermediate values for all iterations, and a `for` loop computes all 32 steps sequentially in hardware. Since the computation is fully combinational, the output depends directly on the input A and does not require multiple clock cycles.

  This architecture uses 4,503 logic elements and 97 registers. Among these, only one register is functionally required for the state signal, while the remaining 96 registers are introduced for the Quartus synthesis tool. This results in a high area cost and a long critical path of approximately 229 ns, caused by chaining all 32 iterations within a single large combinational block. Consequently, the maximum achievable clock frequency is limited to about 4.37 MHz.

  The main advantage of this approach is that the square root is obtained in a single clock cycle, leading to a total computation time of approximately 228 ns. This is significantly faster than Architecture 1 (approximately 2.5 μs), but it comes at the expense of much higher hardware utilization and a lower maximum operating frequency.

3.2 Simulation Results

Figure 12: RTL simulation of the computation of the square root of 3, 15, 127 and 4.294.967.295

- From the simulation results, it can be observed that the output becomes valid after only two clock cycles: the first cycle is used to sample the `start` signal, and the second cycle is required to compute the result. At the end of the computation cycle, the `finished` signal is asserted.

4 Architecture 4

The VHDL file implementing this architecture is named **itera_sqrt_a3.vhd**.

4.1 Synthesis Results

Flow Summary	
Flow Status	Successful - Mon Dec 15 16:45:55 2025
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	sqrt
Top-level Entity Name	it_sqrt
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	5,009 / 18,752 (27 %)
Total combinational functions	4,742 / 18,752 (25 %)
Dedicated logic registers	1,891 / 18,752 (10 %)
Total registers	1891
Total pins	100 / 315 (32 %)
Total virtual pins	0
Total memory bits	688 / 239,616 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

(a) Nb of logical elements

Slow Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-8.102	Z[17][0]	R[18][32]	clk	clk	1.000	0.012	9.152
2	-8.056	Z[27][0]	Z[28][0]	clk	clk	1.000	-0.001	9.093
3	-8.003	Z[20][0]	Z[21][0]	clk	clk	1.000	0.015	9.056
4	-7.947	Z[25][0]	Z[26][0]	clk	clk	1.000	-0.002	8.983
5	-7.904	Z[30][0]	Z[31][0]	clk	clk	1.000	-0.001	8.941
6	-7.869	Z[23][0]	Z[24][0]	clk	clk	1.000	0.006	8.913
7	-7.831	Z[12][5]	Z[13][0]	clk	clk	1.000	0.003	8.872
8	-7.825	Z[12][1]	Z[13][0]	clk	clk	1.000	0.003	8.866
9	-7.792	Z[18][0]	Z[19][0]	clk	clk	1.000	0.006	8.836
10	-7.770	Z[17][0]	Z[18][0]	clk	clk	1.000	0.012	8.820
11	-7.742	Z[26][0]	Z[27][0]	clk	clk	1.000	-0.002	8.778
12	-7.729	Z[22][0]	Z[23][0]	clk	clk	1.000	0.005	8.772
13	-7.713	Z[28][0]	Z[29][0]	clk	clk	1.000	-0.006	8.745
14	-7.703	Z[10][0]	Z[11][0]	clk	clk	1.000	-0.024	8.717
15	-7.696	Z[24][0]	Z[25][0]	clk	clk	1.000	0.000	8.734
16	-7.679	R[12][0]	Z[13][0]	clk	clk	1.000	0.003	8.720
17	-7.677	R[12][3]	Z[13][0]	clk	clk	1.000	0.003	8.718
18	-7.661	Z[29][0]	Z[30][0]	clk	clk	1.000	-0.002	8.697
19	-7.656	Z[16][0]	Z[17][0]	clk	clk	1.000	-0.003	8.691
20	-7.636	Z[12][0]	Z[13][0]	clk	clk	1.000	0.018	8.692

(b) critical paths

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	109.87 MHz	109.87 MHz	clk	

(c) Maximal frequency supported by the architecture

Figure 13: Results of synthesis for pipelined iterative architecture for cyclone II target

- Architecture 4 is a pipelined implementation of the square-root algorithm. Thanks to the pipeline registers, the critical path within a single stage is short (approximately 9.1 ns), allowing the design to operate at a high clock frequency of about 109.6 MHz. However, this comes at the cost of a large number of registers: the design uses 5,009 logic elements and 1,891 registers, making it the most resource-intensive architecture in terms of both logic elements and registers.

  This architecture achieves very high throughput, producing one result per clock cycle once the pipeline is filled, with a latency of 32 clock cycles. Nevertheless, it requires significantly more area than the iterative Architecture 2.

4.2 Simulation Results

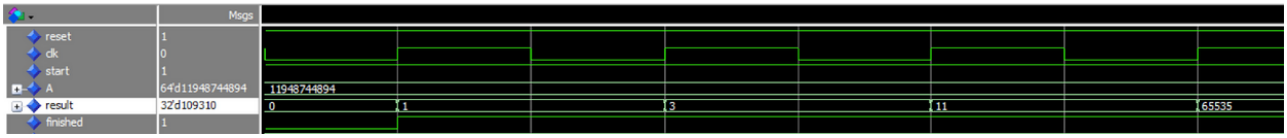

Figure 14: RTL simulation of the pipelined computation of the square root of 3, 15, 127 and 4.294.967.295

- The simulation shows that the user should provide a new input number at each clock cycle. Once the `finished` signal is asserted, the user receives all the corresponding square-root results, one result per cycle.

5 Architecture 5

The VHDL file implementing this architecture is named **itera_sqrt_a4.vhd**.

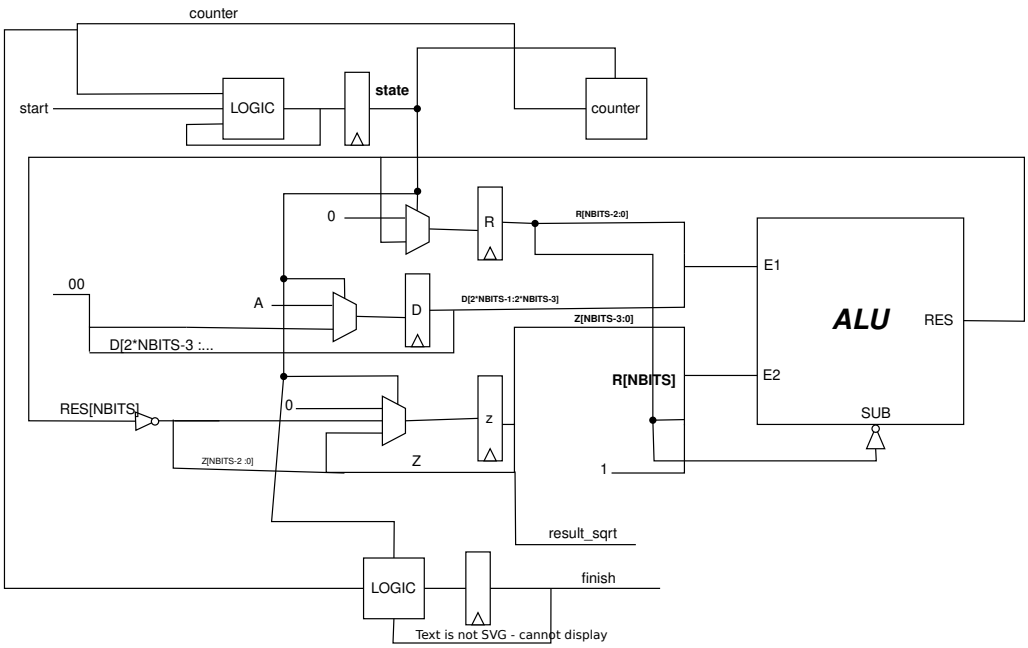

Figure 15: Structural overview of the architecture

5.1 Synthesis Results

Flow Summary	
Flow Status	Successful - Mon Dec 15 16:32:16 2025
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	sqrt
Top-level Entity Name	itera_sqrt_a4
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	210 / 18,752 (1 %)
Total combinational functions	210 / 18,752 (1 %)
Dedicated logic registers	138 / 18,752 (< 1 %)
Total registers	138
Total pins	100 / 315 (32 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

(a) Nb of logical elements

Slow Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-5.229	Z[0]	R[32]	clk	clk	1.000	0.000	6.267
2	-5.226	Z[0]	Z[0]	clk	clk	1.000	0.000	6.264
3	-5.178	Z[1]	R[32]	clk	clk	1.000	0.000	6.216
4	-5.175	Z[1]	Z[0]	clk	clk	1.000	0.000	6.213
5	-5.149	R[32]	R[32]	clk	clk	1.000	0.000	6.187
6	-5.146	R[32]	Z[0]	clk	clk	1.000	0.000	6.184
7	-5.139	Z[2]	R[32]	clk	clk	1.000	0.000	6.177
8	-5.136	Z[2]	Z[0]	clk	clk	1.000	0.000	6.174
9	-4.971	Z[3]	R[32]	clk	clk	1.000	0.000	6.009
10	-4.968	Z[3]	Z[0]	clk	clk	1.000	0.000	6.006
11	-4.916	Z[11]	R[32]	clk	clk	1.000	0.000	5.954
12	-4.913	Z[11]	Z[0]	clk	clk	1.000	0.000	5.951
13	-4.892	Z[4]	R[32]	clk	clk	1.000	0.000	5.930
14	-4.889	Z[4]	Z[0]	clk	clk	1.000	0.000	5.927
15	-4.847	R[1]	R[32]	clk	clk	1.000	0.000	5.885
16	-4.844	R[1]	Z[0]	clk	clk	1.000	0.000	5.882
17	-4.831	Z[6]	R[32]	clk	clk	1.000	0.000	5.869
18	-4.828	Z[7]	R[32]	clk	clk	1.000	0.002	5.868
19	-4.828	Z[6]	Z[0]	clk	clk	1.000	0.000	5.866
20	-4.825	Z[7]	Z[0]	clk	clk	1.000	0.002	5.865

(b) Maximal frequency supported by the architecture

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	160.54 MHz	160.54 MHz	clk	

(c) critical paths

Figure 16: Results of synthesis for structural iterative architecture for cyclone II target

- In this architecture, the area is significantly reduced because the design does not rely on multiple parallel adders or subtractors. Instead, a single arithmetic unit is reused across the different computation steps under the control of the FSM. By sharing the same adder/subtractor hardware rather than duplicating it for each operation or iteration,

the amount of combinational logic required is greatly reduced. This resource-sharing approach minimizes logic utilization while still preserving correct functionality, making the architecture much more area-efficient compared to designs that instantiate several adders in parallel. At the same time, the architecture achieves a very high maximum operating frequency and the critical path is short. This architecture achieves the perfect balance between low area and high frequency, it represents the best overall design among the studied implementations.

5.2 Simulation Results

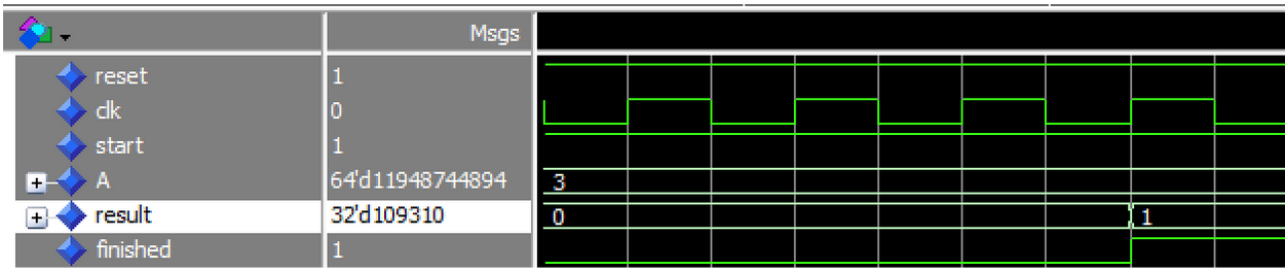

Figure 17: RTL simulation of the computation of the square root of 3

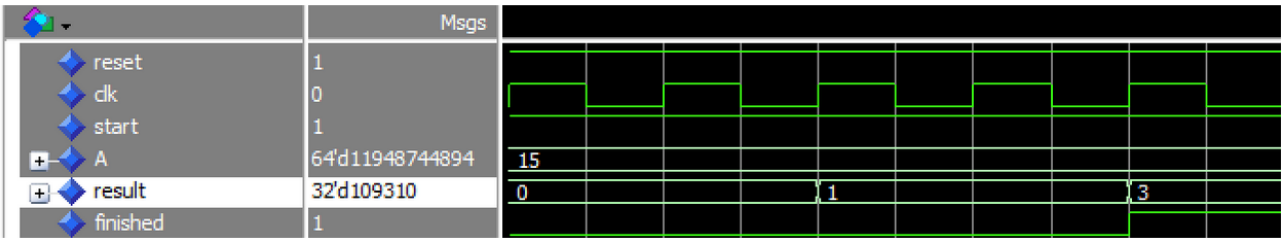

Figure 18: RTL simulation of the computation of the square root of 15

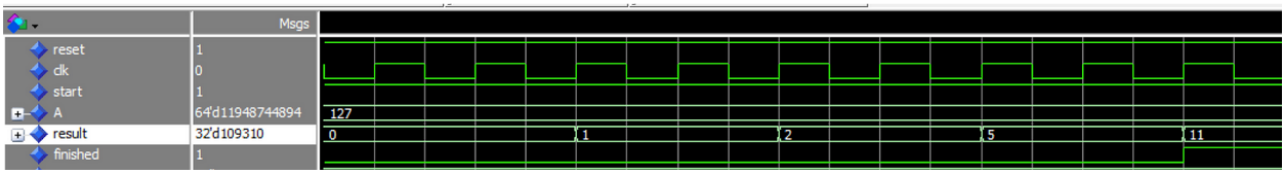

Figure 19: RTL simulation of the computation of the square root of 127

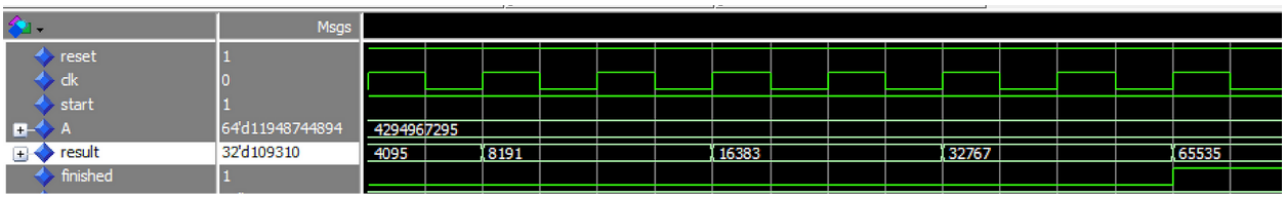

Figure 20: RTL simulation of the computation of the square root of 4.294.967.295

- We notice that the simulation results of this architecture are the same as those of Architecture 2, the only differences are the number of combinational blocks and the maximum operating frequency.

6 Conclusions

6.1 Architectures comparaison

Table 1: Comparison of architectures performance and resource usage

Architecture	Fmax (MHz)	Logical Elements	Registers	Computation of 1 number [μs]	Computation of 500 numbers [μs]
Architecture 1	2.41	4513	68	X	Y
Architecture 2	115.62	309	138	0.27	138.38
Architecture 3	4.37	4503	1	0.22	114.41
Architecture 4	109.87	5009	1891	0.29	4.83
Architecture 5	160.54	210	138	0.199	99.66

6.2 The best choice

In summary, the different architectures illustrate clear trade-offs between area, latency, throughput, and maximum operating frequency. The Newton-based architecture performs the worst, as it suffers from a very low operating frequency and a large number of logic elements due to the use of a divider. The iterative non-restoring architecture provides a good compromise between area and speed, while the fully combinational version achieves minimal latency at the cost of a very large area and a low maximum frequency. The pipelined architecture maximizes throughput but requires a large number of registers and logic elements. Finally, the FSM-based architecture with a shared arithmetic unit achieves both very low area and very high operating frequency, making it the most balanced and efficient solution among the studied implementations.
But what is the best choice :

- **Architecture 4:** This architecture is the best choice when computing the square root of a large number of elements, for example in digital signal processing or AI applications involving large datasets. In such cases, square roots must be computed for large arrays of data, because this architecture significantly speeds up data processing due to its high throughput.
- **Architecture 5:** This architecture is the most suitable for normal use cases where square root computations are performed frequently but on only one or a few values at a time. It is energy-efficient and uses minimal resources, resulting in a small hardware area.

7 Makefile and TestBench guide

7.1 Makefile

This Makefile works perfectly under Linux, but running it under WSL can be slightly tricky due to differences in file system paths and library of Linux environment.

- `SRC`: Lists all VHDL source files located in the `src` directory and includes the testbench file `sqrt_TB.vhd`.
- `TB`: Defines the name of the top-level testbench entity used during simulation.
- `all`: Default target that launches the complete simulation flow.
- `work`: Creates the `work` library required by ModelSim to store compiled VHDL units.

- `compile`: Compiles all VHDL source files into the `work` library using the `vcom` compiler.
- `sim`: Runs the simulation in command-line mode using `vsim`, executes the testbench until completion, and then exits automatically. This step is performed from WSL using the terminal (older Debian libraries may be required to do that in WSL).
- `deplace`: Copies the VHDL source files from the WSL file system to the Windows ModelSim source directory, enabling interoperability between WSL and Windows. Because impossible to run gui simulation on WSL.
- `clean`: Removes temporary files generated during simulation, such as logs and waveform files.
- `mrproper`: Performs a full cleanup by deleting the `work` library and all generated simulation artifacts.

7.2 TestBench

- **1.** To add a new test, include it in the array and remember to update the variable `N_test`.
- **2.** To change the architecture being tested, simply modify the name of the component to the desired architecture. For the iterative architecture, use `entity work.it_sqrt(ax)` to select exactly which architecture x you want to simulate.
- **3.** When simulating the pipelined architecture, set the `PIPE_LINE` variable to 1 to enable simultaneous input bursts and take full advantage of the pipeline.
- **4.** The simulation returns the status of each test, indicating whether it completed successfully or failed.