

A CORE DESIGN TO OBTAIN SQUARE ROOT BASED ON A NON-RESTORING ALGORITHM

Daniel R. Llamocca-Obregón

llamocca.dr@pucp.edu.pe

Grupo de Procesamiento Digital de Señales e Imágenes - Pontificia Universidad Católica del Perú
Av. Universitaria S/N Cdra. 18 - Lima 32, Perú
Telf.: +511-6262000 Anexo 4681

ABSTRACT

A core that obtains a fixed-point square root is presented. The algorithm used has been proposed by Li and Chu [1]. Three types of architecture are presented: a low cost iterative version, a fully pipelined version, and a fully combinatorial version.

The user can scale the size of the core, choose the precision bits required and select the architecture by simply setting parameters.

In order to prove the efficiency of the algorithm, the algorithm is targeted to FPGAs in order to establish a comparison with a core provided by a leading vendor: ALTERA Corporation. The results are very encouraging: the core consumes fewer gates, is faster and has more options than the ALTERA core.

The core is described in VHDL, and the results were obtained using the Quartus II 4.2 Web Edition.

1. INTRODUCTION

Square root is a basic operation in computer graphics and scientific calculation applications such as math co-processors, DSP algorithms, data processing and control [4]. So, there is a great need to enhance this operation. In 1996, Lu and Chi [1] have proposed a 'new non-restoring sqrt algorithm' for VLSI implementation, which has proved to be better than the existing VLSI algorithms.

The core provides three types of architecture: iterative (which is not provided in the ALTERA core), pipelined and combinatorial. The remainder can be obtained using an extra adder, but since it is rarely used, it is dismissed to reduce the resource usage.

If the user indicates more precision bits, the core considers this case to enhance the hardware; In the case of the ALTERA core, this parameter can not be indicated [2], and a well-known technique must be used to get precision bits from this core, this means that this case is not considered in order to enhance the architecture, and the hardware reduction relies solely on the synthesis tool.

The next sections are divided as follows: Section 2 describes the algorithm. Section 3 presents the three architectures. Section 4 explains the approach taken to get

more precision bits and the changes of the architectures. Next, in the results, a detailed comparison between the core and the ALTERA's core is presented. Finally, conclusions and recommendations are given.

2. A BRIEF DESCRIPTION OF THE NON-RESTORING ALGORITHM

2.1 New non-restoring Algorithm

Radical: 'D' of '2n bits. Square root: 'Q' of 'n' bits:

D:

D_{2n-1}	D_{2n-2}	D_{2n-3}	D_{2n-4}	\dots	D_1	D_0
------------	------------	------------	------------	---------	-------	-------

Q:

Q_{n-1}	Q_{n-2}	\dots	Q_0
-----------	-----------	---------	-------

Define :

$$d_k = D_{2n-1}D_{2n-2} \dots D_k, k \leq 2n-1$$

$$q_k = Q_{n-1}Q_{n-2} \dots Q_k, k \leq n-1$$

* Note that q_k has 'n - k' bits

$$r'_{n-1} \leftarrow D_{2n-1}D_{2n-2} - 01$$

$$Q_{n-1} \begin{cases} 1, & \text{if } r'_{n-1} \geq 0 \\ 0, & \text{if } r'_{n-1} < 0 \end{cases}$$

for $k = n-2$ downto 0 do

$$r'_k \begin{cases} r'_{k+1} D_{2k+1} D_{2k} - q_{k+1} 01, & \text{if } Q_{k+1} = 1 \\ r'_{k+1} D_{2k+1} D_{2k} + q_{k+1} 11, & \text{if } Q_{k+1} = 0 \end{cases}$$

$$Q_k \begin{cases} 1, & \text{if } r'_k \geq 0 \\ 0, & \text{if } r'_k < 0 \end{cases}$$

end

remainder $R = r_0 :$

$$r_0 \begin{cases} r'_0 & \text{if } r'_0 \geq 0 \\ r'_0 + q_1 01, & \text{if } r'_0 < 0 \end{cases}$$

At each iteration, q_k (the square root of d_{2k}), is computed. Since $d_{2k} = d_{2(k+1)}D_{2k+1}D_{2k}$, that is $D_{2k+1}D_{2k}$ is attached to $d_{2(k+1)}$ to form d_{2k} , it can be inferred that $D_{2k+1}D_{2k}$ must be used to get q_k . That explains the fact the algorithm attaches $D_{2k+1}D_{2k}$ to r'_{k+1} to form r'_k in order to get q_k . The remainder at each iteration, called r_k , has 'n-k+1' bits, one more bit than q_k [1]: $r_k = R_n R_{n-1} R_{n-2} \dots R_k$, $k \leq n$. But the algorithm uses an estimated remainder, called r'_k , that has 'n-k+2' bits, the MSB is the sign bit, which decides the value of Q_k , and it can be demonstrated that only the 'n-k+1' least significant bits of r'_k are used to get the next estimated remainder r'_{k-1} . Also, in order to get the real remainder $R = r_0$, only the 'n+1' LSBs of r'_0 are needed (the MSB determines Q_0). It lessens the gate count, since a register of only 'n-k+1' bits is needed for r'_k .

3. ARCHITECTURES

3.1 Fully-Pipelined Architecture

To develop the architecture, it is necessary to unfold the algorithm described in Section 2.1. Therefore 'n' stages with 'n' adders/subtractors will appear. By observing the first iteration, a reduction is obtained:

$$\begin{aligned} r'_{n-1} &\leftarrow D_{2n-1}D_{2n-2} - 01 \\ Q_{n-1} &\leftarrow 1, \text{ if } r'_{n-1} \geq 0 \\ Q_{n-1} &\leftarrow 0, \text{ if } r'_{n-1} < 0 \end{aligned}$$

Note that r'_{n-1} and Q_{n-1} depend solely on $D_{2n-1}D_{2n-2}$, and a truth table can be described:

D_{2n-1}	D_{2n-2}	$r'_{n-1} = cba$	Q_{n-1}
0	0	111	0
0	1	000	1
1	0	001	1
1	1	010	1

$$\begin{aligned} Q_{n-1} &= \bar{c} = D_{2n-1} + D_{2n-2} \\ b &= (D_{2n-1} \oplus D_{2n-2}) \\ a &= \overline{D_{2n-2}} \end{aligned}$$

As stated in Section 2.1, 'c' is only used to get Q_{n-1} and won't be used further, thus, it is discarded.

There is no need to perform the first subtraction and wait one cycle, if the result from the first iteration can be obtained directly from the first 2 MSBs of D. So the first stage can be embedded into the second stage, and there will be 'n-1' pipeline stages.

This architecture is depicted in Figure 1. The computation of the remainder is not considered, although the core computes it if the user wants. Note that the dotted rectangles indicate the registers that would have appeared if the reduction of the first stage hadn't been performed. Such architecture can obtain a new square root each cycle. The initial latency is 'n' cycles.

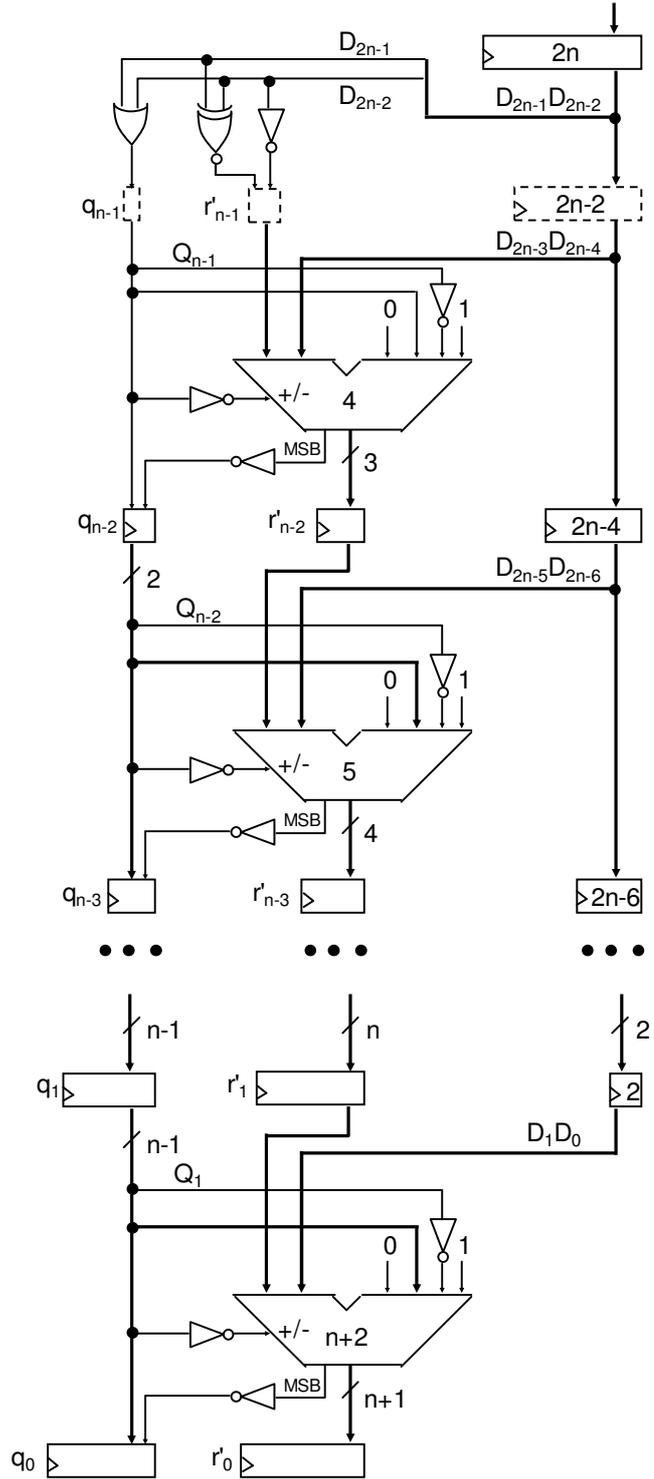


Figure 1. Pipelined sqrt

The longest path delay occurs in the last stage, because the adder/subtractor increases in size as stages advance. A further improvement can be made if the last stages are pipelined, and the initial ones merged.

3.2 Fully-combinatorial Architecture

This architecture is mentioned because some applications may need it, and in order to establish a comparison with the ALTERA core that does have a fully-combinatorial architecture. The architecture is very simple: It is the fully-pipelined architecture without the pipelining registers. It only has one register at the input and one at the output.

3.3 Low-cost iterative Architecture

The size of the elements (registers, adder/subtractor) will be the size of the last stage of the pipelined architecture:

Register R \rightarrow 'n+1' bits Register Q \rightarrow 'n' bits
Adder / subtractor \rightarrow 'n+2' bits.

Since all iterations are embedded in one stage, the reduction of Section 3.1 can not be used.

But a simplification for this case exists:

In the adder/subtractor: the 2 LSBs performs either 'xy-01' or 'xy+11', 'xy' is the pair of D bits used at each step. The operation yields: 'cba'. The truth table is shown:

cba = xy + 11		cba = xy - 01	
xy	cba	xy	cba
00	011	00	111
01	100	01	000
00	101	00	001
01	110	01	010

'c' : carry-in for the next stage of the adder/subtractor
'ba': result of the operation.

'ba' depends only on 'xy', but 'c' depends on the type of operation. Luckily, a conventional adder/subtractor with carry-in (e.g. the ALTERA lpm_add_sub megafunntion) treats the carry-in as positive logic when adding, and as negative logic when subtracting [3] (this is done to reduce gates usage). So, for subtraction, we have to invert 'c' to assure the proper working of the adder/subtractor. The new truth table is:

cba = xy + 11		cba = xy - 01	
xy	cba	xy	cba
00	011	00	011
01	100	01	100
00	101	00	101
01	110	01	110

Now, 'c' and 'ba' depends only on 'xy':

$$c = x + y \quad b = (x \oplus y) \quad a = \bar{y}$$

This reduces the width of the adder/subtractor by 2 bits. The result 'ba' is obtained in parallel and the carry-in comes from just an OR gate. So the new adder/subtractor uses 'n' bits and has carry-in. Also, note that the MSB of the second operator of the adder/subtractor is '0' as in the pipelined case. Figure 2 depicts this architecture.

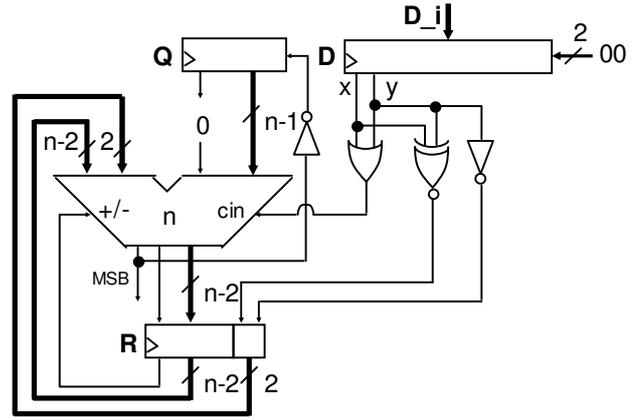


Figure 2. Low-cost iterative sqrt

This FSM controls the iterative architecture. The process start when s = 1. After 'n' clock cycles, the result is obtained in register Q, done = 1, and a new process can be started.

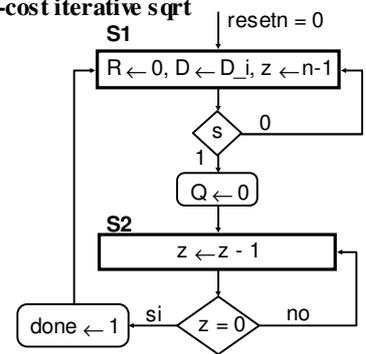


Figure 3. iterative sqrt's FSM

4. COMPUTING MORE PRECISION BITS

If 'x' more precision bits are needed, a well-known technique is to attach '2x' zeros to D. So, D has '2n+2x' bits, and Q 'n+x' bits. This is the only way to do so in the ALTERA core. The core presented has 'x' as a parameter. Let define: $nq = n+x$. 'x': number of precision bits

4.1. Iterative case

The same hardware of Figure 2 is used, except that:
- Q uses 'nq' bits - R uses 'nq+1' bits
- The adder/subtractor uses 'nq+2' bits.

D remains with '2n' bits since '00' is shifted left and emulates the attachment of '2x' zeros to D. In the FSM, 'z' starts with 'nq-1', the result is obtained after 'nq' cycles.

4.2 Pipelined case (also applied to the combinatorial)

There are 'nq-1' stages. The first 'n-1' stages are equal to those of Section 3.1. The last 'x' stages are different, since the new pair of D bits is '00', which is attached to the adder/subtractor; a register for '00' is not needed. So, the number of D registers remains 'n-1' and its maximum size remains '2n'. At the final stage, the register Q has 'nq' bits, the register R 'nq+1' bits and the adder/subtractor 'nq+2' bits. There are 'nq-1' adder/subtractors, Q and R registers. The hardware for the 1st and last stages is shown in Figure 4. Initial latency: 'nq' cycles.

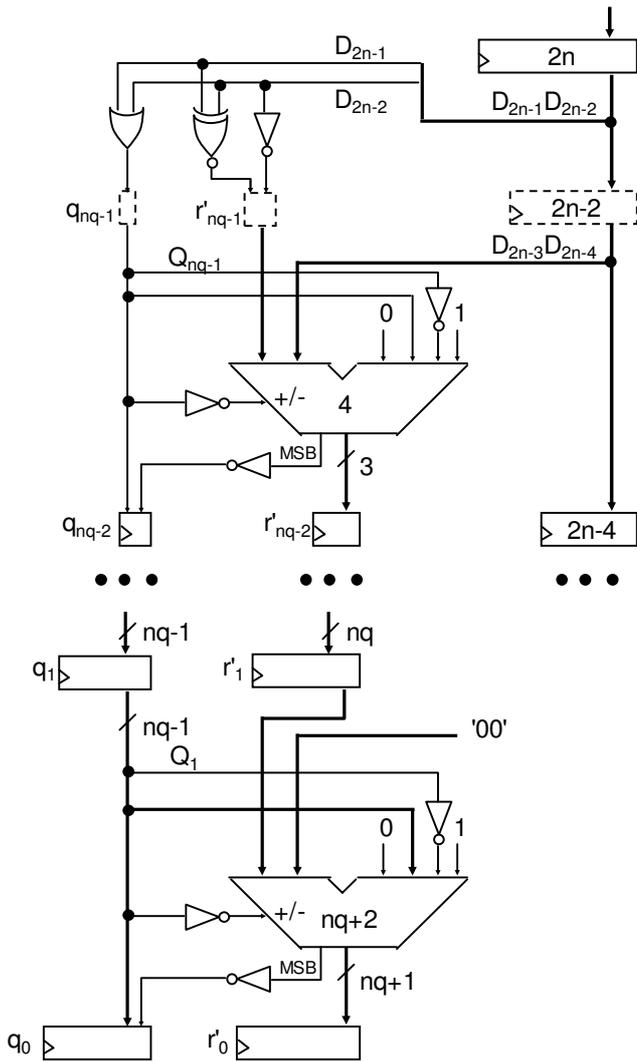


Figure 4. Pipelined sqrt with more precision bits

5. RESULTS

The core presented does not compute the remainder, since it is rarely used. Figure 5 depicts the core with all its options. Table 1 establishes a comparison between this core and the ALTERA core.

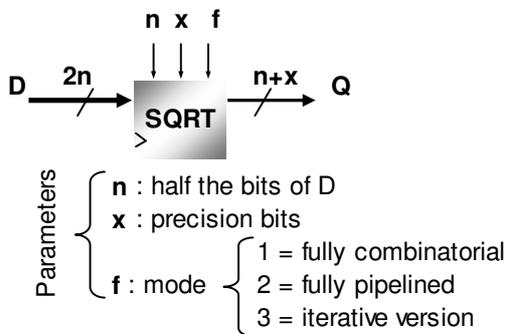


Figure 5. CORE presentation

The remainder is also dismissed in the ALTERA core in order to establish a fair comparison.

Results are shown only for a specific device because the amount of results is huge with just one device and the results are enough to demonstrate the benefits of the core presented.

Type	2n	x	OUR core		ALTERA core	
			Les	f _{max}	LEs	f _{max}
FULLY COMBINATORIAL (Unfolded) Latency cycles: 1	12	0	48	89.56	67	86.59
		4	94	47.05	130	44.88
		8	156	32.90	222	28.95
	16	0	73	62.46	107	57.80
		4	127	39.87	186	35.37
		8	197	27.57	294	22.64
	32	0	213	28.76	347	23.18
		4	299	20.54	490	17.36
		8	401	16.15	662	13.50
FULLY PIPELINED (Unfolded) pipelined Latency cycles: 1	12	0	82	317.86	105	273.45
		4	158	273.97	199	252.02
		8	266	279.17	337	246.49
	16	0	142	301.30	182	243.66
		4	234	274.80	300	247.59
		8	358	266.05	462	245.64
	32	0	542	267.02	690	227.01
		4	698	265.82	904	231.80
		8	886	258.00	1162	224.77
ITERATIVE (Folded recursive) Latency cycles: n + x	12	0	41	344.47	-	-
		4	54	320.92	-	-
		8	66	292.74	-	-
	16	0	50	316.46	-	-
		4	64	281.29	-	-
		8	75	273.52	-	-
	32	0	91	274.50	-	-
		4	105	268.96	-	-
		8	117	257.27	-	-

Table 1. Final Results

Device selected: Stratix EP1S10F48C5
 '2n' = number of bits of the radical D
 'nq' = n + x = number of bits of Q
 'x' = number of additional precision bits
 f_{max} is in MHz.

For every architecture implemented and for every number of bits and precision bits, the core presented is always better in terms of both speed and resource usage than the ALTERA core.

Although the improvement is not huge, it is significant, and as will be stated in the recommendations, the pipelined architecture presented can be further improved.

6. CONCLUSIONS

- The core presented achieves high speed at minimum cost since it only uses an adder/subtractor unit to perform the operations.
- The core is very flexible, so that the user can choose the best architecture that his application needs.
- The efficiency of this core can be observed directly from the hardware shown, and not solely from the fact that the core performs better than the ALTERA's.,

7. RECOMMENDATIONS

The results are better in terms of speed and resource effort than the ALTERA core, and an iterative version is presented that the ALTERA core lacks. However some improvements remain to be done:

- For the pipelined architecture, the pipelined applied resembles each step of the algorithm. However this can be changed: some stages can be further pipelined and others can be merged in order to get the maximum frequency of operation, since each stage has a different size. It is left to the reader to establish the best pipelining approach.
- The simplification for the iterative architecture can be applied to each stage of the pipelined architecture. It is left to the reader to test this improvement.

7. REFERENCES

- [1] Y. Li and W. Chu, "A New Non-Restoring Square Root Algorithm and Its VLSI Implementations", Proc. Of 1996 IEEE International Conference on Computer Designs: VLSI in Computers and Processors, Austin, Texas, USA, October 1996, pp538-544.
- [2] ALTERA Corporation, "altsqrt Megafunction User Guide", v. 1.0, September 2004.
- [3] Brown & Vranesic. Fundamentals of Digital Logic with VHDL Design, McGraw Hill, 2000..
- [4] U. Meyer – Baese, Digital Signal Processing with Field Programmable Gate Arrays: Springer-Verlag Berlin Heidelberg, May 2001,