RISC-Linz
Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria, Europe

Implementing GCD Systolic Arrays on FPGA

Tudor JEBELEAN

(August 19, 1994)

RISC-Linz Report Series No. 94-57

Supported by: National Science Foundation (FWF), project no. P10002

Published in: Proceedings of FPLA 94 (Field Programmable Logic and Applications), Prague, Czech Republic, Sept. 1994.

Implementing GCD Systolic Arrays on FPGA

Tudor Jebelean

RISC-Linz, A-4040 Linz, Austria
jebelean@risc.uni-linz.ac.at

Abstract. We implement on Field Programmable Gate Arrays from Amtel (old CLi) three systolic algorithms for the computation of greatest common divisor of integers. The experiments show that elimination of global broadcasting significantly reduces both area and time consumption. We eliminate broadcasting by using a novel technique which is more suitable to arithmetic algorithms than Leiserson conversion lemma.

This report is part of an on-going research aimed at speeding-up exact-arithmetic systems by adding a systolic dedicated coprocessor. Computation of the greatest common divisor (GCD) of long integers is an important and expensive subalgorithm in exact arithmetic and in some cryptography schemes. Practically, the systolic approach has been so far the most successful in parallelizing this algorithm - see [BrKu85], [YuZh86], [Guyo91].

We compare three GCD systolic implementations using field programmable logic from Amtel (old CLi): a **semisystolic** parallelization of a variant of Brent-Kung plus-minus algorithm, using signed arithmetic and global broadcasting; a purely **systolic** algorithm previously described in [Jebe93], which is obtained from the first by eliminating global broadcasting; an **improved** version of the purely systolic algorithm by halving the number of cells.

We start from a simplified version of the plus-minus algorithm introduced by [BrKu85]. In the description in figure 1, A, B are the input integers and a_0, a_1, b_0, b_1 denote their least-significant bits. A termination and correctness proof of this algorithm are presented in [Jebe94].

1 Systolic Algorithms

In order to parallelize the plus-minus algorithm systolically, the operands are represented using signed-digits, for avoiding carry propagation. Each cell contains 9 registers (variables): the 3-bit signed-digit representation of each operand; one-bit "tags" *ta, tb* which show the sign-bits of each operand; *sa*: the sign of A (finally of the result). Cell 0 (the rightmost) determinates the "instruction" to be executed, which is than broadcasted to all the other cells (there are 5 "instructions"). The tags *ta, tb* are essential for termination detection in the presence of variable-length operands. The tags are set to 1 in those processors containing

* Supported by Austrian FWF, project P10002-PHY.

```
while B ≠ 0 do
   if a0 = 0 and b0 = 0 then [shift both]
      (A, B) ← (A/2, B/2); (As, Bs) ← (A, B)
   if a0 = 0 and b0 = 1 then [interchange and shift]
      (A, B) ← (B, A/2)
   if a0 = 1 and b0 = 0 then [shift B]
      (A, B) ← (A, B/2)
   if a0 = 1 and b0 = 1 then [plus-minus and shift]
      if a1 = b1 then (A, B) ← (B, (A − B)/2)
                 else  (A, B) ← (B, (A + B)/2)
return A [the pseudo-GCD], As, Bs
```

Fig. 1. The plus-minus algorithm.

non-significant bits, and shifted rightward when possible. When tb reaches cell 0, than we know B is null. Also, the value of the lowest tagged bit of A is shifted rightward using sa. At the end this will indicate the sign of the result.

For the elimination of global broadcasting we do not use Leisersons systolic conversion scheme [Leis82]. This method would triplicate the number of registers and the new circuit will inherit the complications of signed-redundant arithmetic. Rather, we us a novel technique, which is more suitable to least-significant bits first arithmetic algorithms: the carries/borrows are rippled along together with the "instruction" signal (which has now 8 values), and the operands are represented in the classical fashion. This reduces the number of registers needed for operands by a factor of 3, compensating the registers added for buffering the instruction signal: only 8 registers are used now instead of 9. The details of this systolic algorithm are presented in [Jebe93].

Similarly to what happens in Leiserson systolic conversion method, our broadcasting-elimination scheme also introduces a slow-down of 2: the array is used with only 50% efficiency. In order to improve this situation, we "pack" two cells in one: a new "double" cell will contain the registers of two old neighboring cells, but only one implementation of the transition function. The function is multiplexed alternatively between the two sets of registers. This brings the theoretical efficiency to 100%, however from the practical point of view only a small improvement in area consumption can be noted.

Nevertheless our systolic algorithms represent a significant improvement over Brent-Kung scheme [BrKu85]. While the old device needed 4n cells, for n-bit operands, each with 24 registers, our devices need n cells with 8 registers (respectively $n/2$ cells with 16 registers) - hence a reduction of 12 times. The running time of the old algorithm is $4n$ for any inputs, the running time of our algorithm is $4.34n$ in the average.

2 Practical Experiments and Conclusions

The three designs were implemented for 8-bits operands using Amtel FPGA development system (successor of Concurrent Logic), which includes CAD from Viewlogic. The layout was done automatically on a 6005 CLi chip (has $56 * 56 = 3{,}136$ primitive cells), and was successful only for the purely systolic designs. For the first (semi-systolic) design, 97 nets out of 694 could not be routed. Figure 2 presents the resulting data.

	semisystolic	systolic	improved
macros	710	476	426
registers	98	89	92
gates	607	377	313
equivalent gates	2,928	2,167	2,210
cells used			
before layout	704	466	405
after layout	2,649	1,889	1,576
time/clock (ns)			
before layout	60.8	56.4	64.0
after layout	650	450	440
increase (times)	11	8	7

Fig. 2. Comparison of the three designs.

The most important conclusion of the experiments is that elimination of global broadcasting using our novel technique is benefic from all points of view: the purely systolic algorithm was **simple** enough for the layout software to be able to process it successfully; the **area consumption** of the semi-systolic device is bigger by 32% (equivalent gates), 73% (logic cells), 68% (layout cells); the **timing** of the semi-systolic device is bigger by 8% (logic), 47% (layout).

References

BrKu85 R. P. Brent, H. T. Kung, *A systolic algorithm for integer GCD computation*, 7th IEEE Symp. on Computer Arithmetic.

Guyo91 A. Guyot, *OCAPI: Architecture of a VLSI coprocessor for the GCD and extended GCD of large numbers*, 10th IEEE Symposium on Computer Arithmetic.

Jebe93 T. Jebelean, *Systolic normalization of rational numbers*, ASAP'93.

Jebe94 T. Jebelean, *Systolic algorithms for long integer GCD computation*, Conpar94.

Leis82 C. E. Leiserson, *Area-efficient VLSI computation*, PhD Thesis, Carnegie Mellon University, MIT Press, 1982.

YuZh86 D. Y. Y. Yun, C. N. Zhang, *A fast carry-free algorithm and hardware design for extended integer GCD computation*, ACM SYMSAC'86.