ET4171 Processor Design Project

LEON3 processor optimization

Henrique Dantas

Luca Feltrin

Delft University of Technology

June 19, 2013

Table of contents

Objectives

- Target: Embedded applications
  - Compound metric: P×BS
- Poor Mul/Div execution time
  - Implementation with different algorithms

Multiplier: Wallace Tree

Algorithm:

1. Initial `AND`ing between inputs.
2. Reduce tree using Full Adders and Half Adders.
3. Add the two remaining numbers with a traditional adder.

Pros and Cons

Pros

- Speed improvement.
- Regular structure.

Cons

- Resource usage is high.
- Signifcant Area footprint.
- Increase in power.

Implementation Details

3 Input Signals:`RST`, `CLK` and `MULI`
1 Output Signal:`MULO`
4 generics: `infer`, `multype`, `pipe` and `mac`
Constants used: WallaceTree (3D logic vector), stages, FA, HA, CIN and RE.

Implementation Details II

For negative operation - Alternative Baugh Wooley Negate AND operations (when includes MSD from op1 or op2 but not both) Add a constant vector at the end.

Implementation Details III

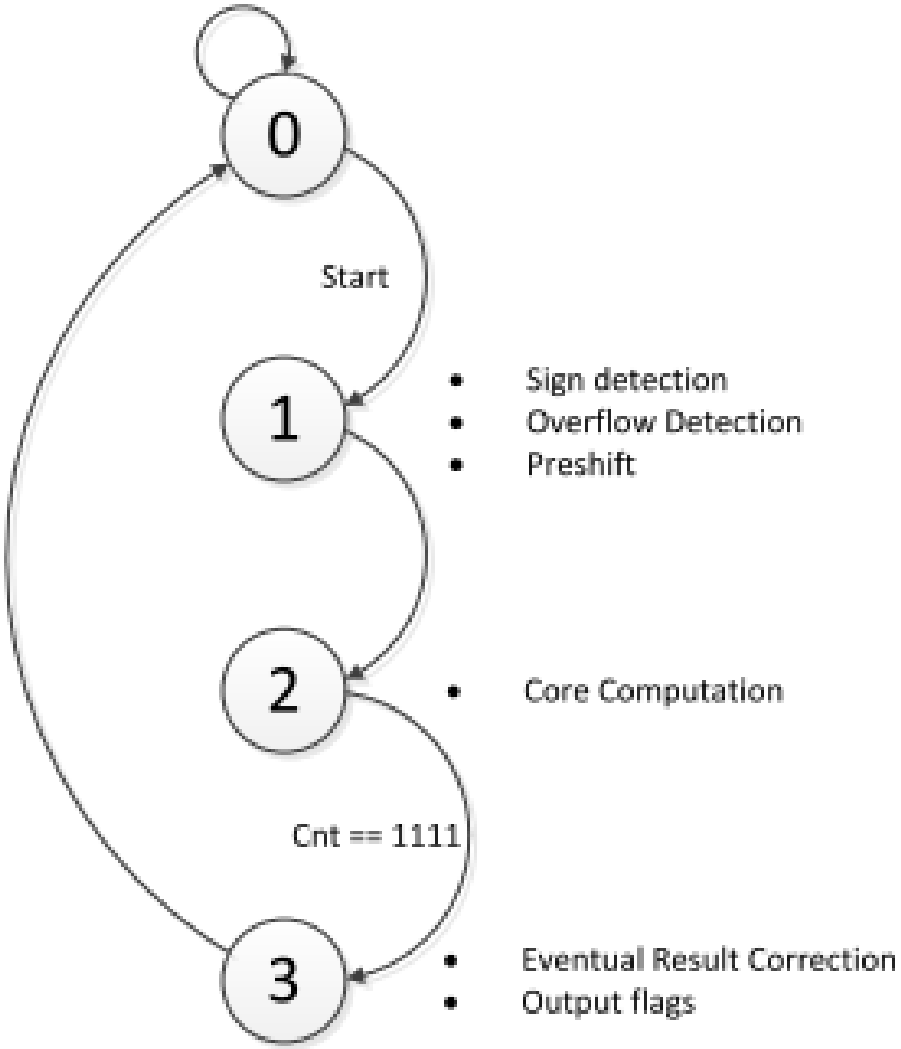
Feed inputs to FOR-GENERATES (with IF-GENERATES) IF holdn is HIGH update Add the two final numbers with Baugh-Wooley vector (for signed operations) (ready and nready are not used). Update MULO.(result/ICC).

Divider: Which algorithm?

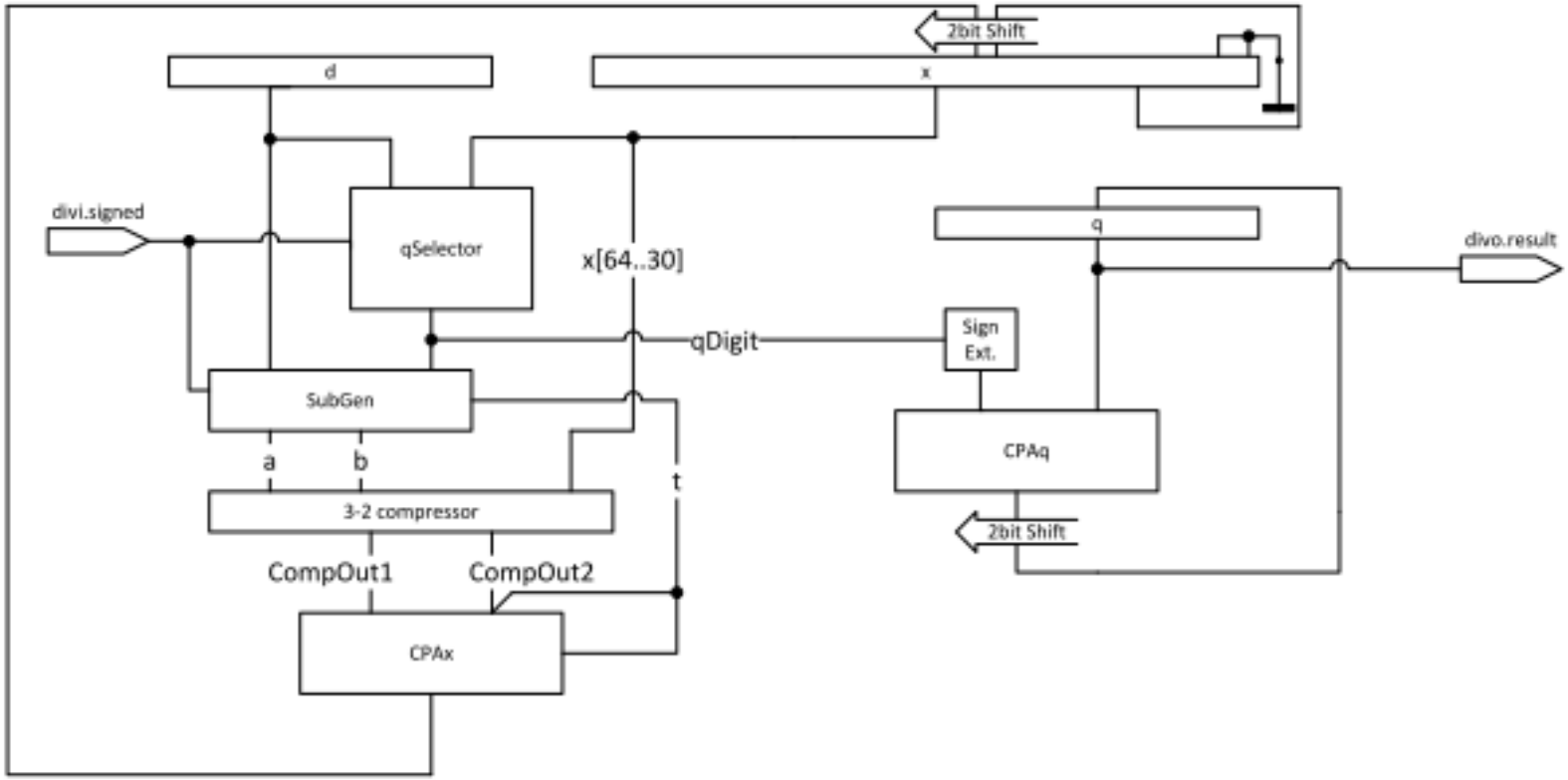
Repeated Multiplication	**Fast**	**Area**
Reciprocation	**Fast**	**Area**
Array Divider	**No control on physical placing**	
Radix >8	**Fast**	**Area**
Radix-4	**Good compromise**	

Execution time $\approx \frac{1}{2}$ of the baseline

Divider II


0
Start
1
Sign detection
Overflow Detection
Preshift
2
Core Computation
Cnt == 1111
3
Eventual Result Correction
Output flags

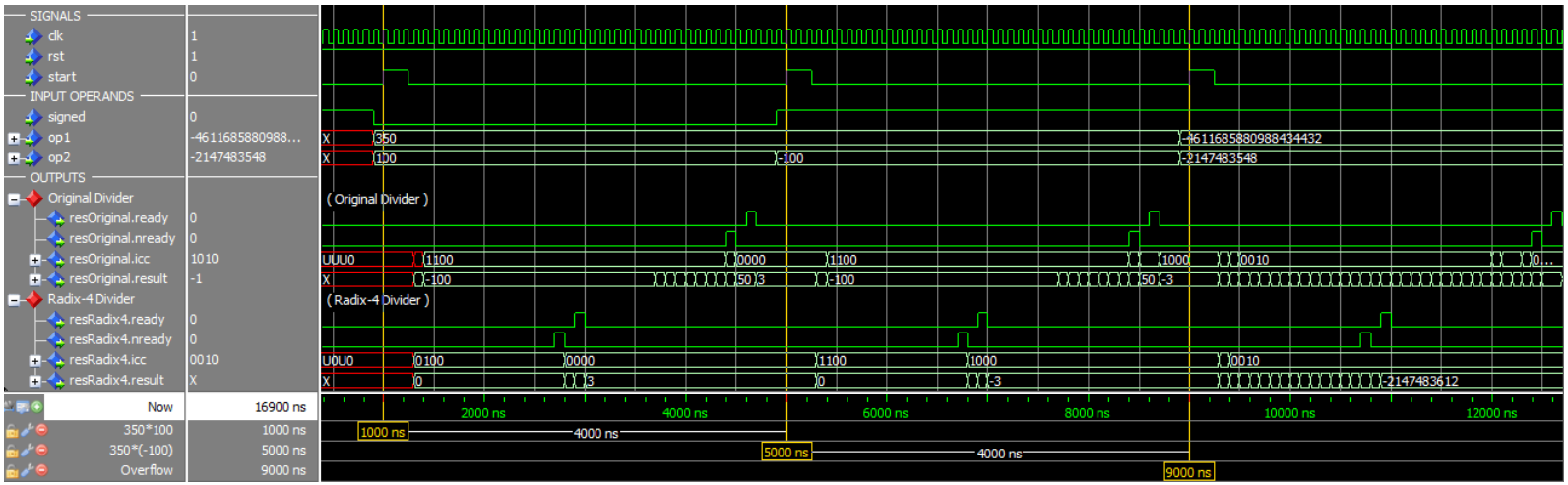
Divider III


2bit Shift
d
x
divi.signed
qSelector
x[64..30]
q
divo.result
qDigit
Sign
Ext.
SubGen
CPAq
a
b
t
3-2 compressor
2bit Shift
CompOut1
CompOut2
CPAx

Dvider IV

- Signed division (signed p-d plot) vs. Unsigned division (half p-d plot) + 1 cycle for sign
  - No area differences but 1 more cycle delay: signed Division

Divider V



Baseline vs. Radix-4: 19 cycles vs 36

Synthesis Results

	Baseline	Div	Div&Mul
Clock freq. [MHz])	80.522	80.535	40.197
Slices	9904	10479	11886
LUTs	16889	17865	20469
Quiescent power [W]	2.467	2.468	2.511
Dynamic power [W]	0.721	0.743	0.832
Total power [W]	3.188	3.211	3.343
P/f [W/MHz]	0.03959	0.03987	0.08317

Benchmarks Scores

	Baseline	Modified
Stanford [s]	2.30	2.21
Whetstone [s]	116.2	112.08
Gmpbench Multiply [Op/ s]	781	914
Gmpbench Divide [Op/ s]	15876	19205
Gmpbench RSA [Op/ s]	5123	5353
Division [s]	8.06	7.31
Mibench JPEG (average) [s]	23.215	21.76
SSD [s]	10.59	8.60
Total [s]	**219.28**	**206.92**

Conclusion - Comparison with Matrics (Div)

	Baseline	Modified	Improvements
A	2.68	2.83	-5.8%
D	1.24	1.24	-
P	3.19	3.21	-0.7%
BS	2.19	2.13	2.7%
A×D	3.33	3.52	-5.8%
A×BS	5.88	6.05	-2.9%
P×D	3.96	3.99	-0.7%
P×BS	6.69	6.85	2.08%

Conclusion II - Comparison with Matrics (Div & Mul)

	Baseline	Modified	Improvements
A	2.68	3.24	-21%
D	1.24	2.49	-100%
P	3.19	3.34	-5.0%
BS	2.19	2.07	6%
A×D	3.33	8.05	-142%
A×BS	5.88	6.69	-14%
P×D	3.96	8.32	-100%
P×BS	6.69	6.92	1%

Further Improvements

- Cache size
  - More power consumption, need to determine actual miss rate
- Branch prediction
  - Now: Static prediction, only slight advantage, power consumption
- Out of Order Execution
  - Radical change of the integer unit, improved execution time