© Journal of the Technical University - Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications"
Vol. 26, 2020

Parallelism in speed focused FPGA based Canny computations

Dimitre Kromichev

Department of Marketing and International Economic Relations, University of Plovdiv, 24 Tzar Asen Street, Plovdiv 4000, Bulgaria

dkromichev@yahoo.com

Abstract. Explored is the complete functional model of parallelism in FPGA based Canny computations. Distinguished are top-, high-, mid- and low level of parallelism. The specific use of each level in tthe different parts of Canny's contour detection algorithm is thoroughly investigated. Studied is the impact of all levels of parallelism on speed on the basis of particular formulae. The mathematics and speed capabilities of a specific type of FPGA based multiplication is analyzed in the context of parallelism.

1. Introduction

Parallel execution of calculations is a typical capability of programmable logic. This is most advantageous to FPGA based Canny focused on speed. In view of the computational complexity of Canny's algorithm, parallelism requires an in-depth analysis and multilevel exploration.

In the literature, parallelism is used in distributed Canny [1][4][5][6][7]. There are some approaches which use parallelism oriented techniques in Gaussian and Sobel filtering [2][3]. So far there is no in-depth study of all the multifaceted characteristics of parallelism in FPGA based Canny. Nor is there a reliable exploration of the impact of parallelism on performance.

The objective is to present a complete study of parallelism in FPGA based Canny. The task is to analyze all applications of parallelism with respect to speed capabilities and organization of computations in Canny modules. Relevant to the conclusions arrived at are only gray-scale images. The targeted hardware is Intel (Altera) FPGAs. The hardware description language is VHDL.

In VHDL, computational parallelism is based on execution of processes. There are two types of processes: implied and explicit. Implied processes are represented by the concurrent signal assignments which execute at the same time in parallel. When a concurrent signal assignment is realized it automatically implies a process. If there are many signal assignments then all of these signal assignments are operating in parallel and must be treated as so by the synthesis tool and the simulation tool. The explicit process contents consist of sequential statements and simple signal assignments. Inside an explicit process all statements are executed sequentially. All implied processes and explicit processes in general execute in parallel. Thus, the functionality of parallelism in FPGA based Canny relies on the collection of processes used to execute the indispensable Canny mathematics.

2. Top level parallelism

This parallelism is based on Canny's typology. It is characteristic of Sobel filtering. In it, two identical sets of sequentially ordered arithmetic operations are executed simultaneously

$$\begin{aligned}
&A=\{\operatorname{Im}pA(i)aopA(n-(n-m)),\operatorname{Im}pA(i)aopA(n-(n-m)),..,\operatorname{Im}pA(i)aopA(n)\}\\
&D=\{\operatorname{Im}pD(i)aopD(n-(n-m)),\operatorname{Im}pD(i)aopD(n-(n-m)),..,\operatorname{Im}pD(i)aopD(n)\}\\
&A \equiv D,\\
&i=\{1,2\},\\
&n\in N,\\
&m=\{1,2,..,n-1\}\ .
\end{aligned} \tag{1}$$

where

$\operatorname{Im}pA(i)$	is the number of parallel implementations of an arithmetic operation within a clock cycle in set A,
$\operatorname{Im}pD(i)$	is the number of parallel implementations of an arithmetic operation within a clock cycle in set D,
$aopA((n-(n-m)),..,(n))$	is a particular arithmetic operation from set A,
$aopD((n-(n-m)),..,(n))$	is a particular arithmetic operation from set D.

Parallelism defined by (1) requires that

$$\begin{aligned}
&Tclk > TaopA(n-(n-m)),TaopA(n-(n-m)),..,TaopA(n)\\
&Tclk > TaopD(n-(n-m)),TaopD(n-(n-m)),..,TaopD(n)
\end{aligned} \tag{2}$$

where

$Tclk$	is the period of the system clock,
$TaopA(n-(n-m)),..,TaopA(n)$	is the propagation delay of every arithmetic operation in set A,
$TaopD(n-(n-m)),..,TaopD(n)$	is the propagation delay of every arithmetic operation in set D,

If

$$\begin{aligned}
&TaopA(n-(n-m+1)+TaopA(n-(n-m))>Tclk\ \&\\
&TaopD(n-(n-m+1))+TaopD(n-(n-m))>Tclk
\end{aligned}$$

then

$$\begin{aligned}
&nTclk(A)=n\\
&nTclk(D)=n
\end{aligned} \tag{3}$$

where

$nTclk(A)$	is the number of clock cycles required to execute all arithmetic operations in set A,
$nTclk(D)$	is the number of clock cycles required to execute all arithmetic operations in set D.

If

$$\begin{aligned}
&TaopA(n-(n-m+1))+TaopA(n-(n-m))<Tclk\ \&\\
&TaopD(n-(n-m+1))+TaopD(n-(n-m))<Tclk
\end{aligned}$$

then

$$\begin{aligned}
&nTclk(A)=n-1\\
&nTclk(D)=n-1
\end{aligned} \tag{4}$$

where

$nTclk(A)$	is the number of clock cycles required to execute all arithmetic operations in set A,
$nTclk(D)$	is the number of clock cycles required to execute all arithmetic operations in set D.

Expressions (4) represent the possible minimum number of clock cycles required to execute operations in A and D.

The FPGA based functional model of Sobel filtering implementing top level parallelism is presented in Figure 1 of Appendix. All arithmetic operations used to calculate the *x*-gradient and the *y*-gradient are exactly the same and are executed simultaneously.

3. High level parallelism

This parallelism is based on Canny's structure. It is characteristic of gradient magnitude and direction computations. In it, two different sets of sequentially ordered arithmetic operations are executed simultaneously.

$$\begin{gathered}
A = \{\operatorname{Im} pA(iA)aopA(nA-(nA-mA), \operatorname{Im} pA(iA)aopA(nA-(nA-mA)),.., \operatorname{Im} pA(iA)aopA(nA)\} \\
D = \{\operatorname{Im} pD(iD)aopD(nD-)nD-mD)), \operatorname{Im} pD(iD)aopD(nD-(nD-mD)),.. \\
.., \operatorname{Im} pD(iD)aopD(nD)\} \\
A \neq D, \\
iA > 2^4, \\
2^2 \leq iD \leq 2^3, \\
nA \in N, nD \in N, nA > nD, \\
mA = \{1,2,.., nA-1\}, \\
mD = \{1,2,.., nD-1\}, \qquad (5)
\end{gathered}$$

where

$\operatorname{Im} pA(iA)$	is the number of parallel implementations of an arithmetic operation within a clock cycle in set A,
$\operatorname{Im} pD(iD)$	is the number of parallel implementations of an arithmetic operation within a clock cycle in set D,
$aopA((nA-(nA-mA)),..,(nA))$	is a particular arithmetic operation from set A,
$aopD((nD-(nD-mD)),..,(nD))$	is a particular arithmetic operation from set D.

Parallelism defined by (5) requires that

$$\begin{gathered}
Tclk > TaopA(nA-(nA-mA)), TaopA(nA-(nA-mA),.., TaopA(nA) \\
Tclk > TaopD(nD-(nD-mD)), TaopD(nD-(nD-mD)),.., TaopD(nD) \qquad (6)
\end{gathered}$$

where

$TaopA(nA-(nA-mA)),.., TaopA(nA)$	is the propagation delay of every arithmetic operation in set A,
$TaopD(nD-(nD-mD)),.., TaopD(nD)$	is the propagation delay of every arithmetic operation in set D.

Because in (6) $nA > nD$, it follows that

$$\text{If } TaopA(nA-(nA-mA+1)) + TaopA(nA-(nA-mA)) < Tclk \text{ then } nTclk(A) = nA-1 \qquad (7)$$

where

$nTclk(A)$ is the number of clock cycles required to execute all arithmetic operations in set A.

Expression (7) represents the possible minimum number of clock cycles required to execute operations in both A and D.

The FPGA based functional model of gradient magnitude is presemted in Figure 3(a) of Appendix. The FPGA based functional model of gradient direction is presented in Figure 3(b) of Appendix. In FPGA based Canny execution, gradient magnitude submodule and gradient direction submodule use the same inputs (the *x*- and *y*-gradients) and work simultaneously, thus implementing high level parallelism.

4. Mid-level parallelism

This parallelism is based on the organization of computations in a particular Canny module. It is characteristic of Gaussian filtering. In it, two or more sets of pipelined arithmetic operations are executed simultaneously.

$$\begin{aligned}
&A1 = \{\operatorname{Im} pA1(i)aopA1(n-(n-m)), \operatorname{Im} pA1(i)aopA1(n-(n-m)),.., \operatorname{Im} pA1(i)aopA1(n)\}\\
&A2 = \{\operatorname{Im} pA2(i)aopA2(n-(n-m)), \operatorname{Im} pA2(i)aopA2(n-(n-m)),.., \operatorname{Im} pA2(i)aopA2(n)\}\\
&AF = \{\operatorname{Im} pAF(i)aopAN(n-(n-m)), \operatorname{Im} pAF(i)aopAN(n-(n-m)),.., \operatorname{Im} pAF(i)aopAF(n)\}\\
&A1 \equiv A2 \equiv ,.., \equiv AN\\
&i = 1\\
&n \in N\\
&m = \{1,2,..,n-1\}
\end{aligned} \tag{8}$$

where

$\operatorname{Im} p(A1, A2,.., AF)(i)$	is the number of implementations of an arithmetic operation within a single clock cycle in each of the sets *A1, A2, .., AF*,
$aop(A1, A2,.., AF)((n-(n-m)),..,(n))$	is a particular arithmetic operation from set *A1, A2, .., AF.*

This level of parallelism requires that:

- All arithmetic operations in set *A1, A2, .., AF* are binary
- One of the operands of arithmetic operation # (n-(n-m)) is the same for all sets $A1, A2,.., AF$.

Parallelism defined by (8) requires that

$$\begin{aligned}
Tclk > {} & Taop(A1, A2,.., AF)(n-(n-m)), Taop(A1, A2,.., AF)(n-(n-m)),..\\
& .., Taop(A1, A2,.., AF(n)
\end{aligned} \tag{9}$$

where

$Taop(A1, A2,.., AF)(n-(n-m))$	is the propagation delay of every arithmetic operation in sets $A1, A2,.., AF$.

This parallelism has two variants:

- Variant #1. If

$$Taop(A1, A2,.., AF)(n-(n-m+1)) + Taop(A1, A2,.., AF)(n-(n-m) > Tclk \tag{10}$$

then arithmetic operations in sets $A1, A2,.., AF$ are pipelined in 3 stages

- Variant #2. If

$$Taop(A1, A2,.., AF)(n-(n-m+1)) + Taop(A1, A2,.., AF)(n-(n-m)) < Tclk \tag{11}$$

then arithmetic operations in sets $A1, A2,.., AF$ are pipelined in 2 stages.

Therefore, this variant is preferred in speed focused computations..

The minimum number of clock cycles required to execute in parallel the pipelined operations in $A1, A2,.., AF$ depends on one of the input variables to Canny: the size of Gaussian filter.

The FPGA based functional model of Gaussian filtering implementing mid-level parallelism is presented in Figure 2 of Appendix. The size of Gaussian filter used in the model is 5x5. Mid-level parallelism is the basis for the fatest technology of Gaussian filtering in terms of minimum number of clock cycles required to process an input image pixel. Mid-level parallelism guarantees that for a Gaussian filter of size zxz the number of input image pixels which are processed simultaneously is equal to z .

5. Low level parallelism

This parallelism is based on the functional mechanism of a particular computational algorithm. It is characteristic of square root and multiplication. It consists of a set of two consecutive arithmetic operations

$$\begin{aligned} &A = \{\mathrm{Im}\, pA(i) aop1, \mathrm{Im}\, pA(i) aop2\} \\ &i > 2^4 \end{aligned} \tag{12}$$

where

$\mathrm{Im}\, pA(i)$	is the number of parallel implementations of an arithmetic operation within a single clock cycle,
$aop1$	is arithmetic operation #1,
$aop2$	is arithmetic operation #2.

In each pair $aop1, aop2$ the output of $aop1$ is the input of a comparison operation in $aop2$. The output of an operation #2 cannot be the input of any other operation #1. Therefore, none of the i implementations of pairs $aop1, aop2$ can be executed sequentially.

This parallelism has two variants:

- If $Taop1 + Taop2 > Tclk$ then $nTclk = 2$ (13)

where

$Taop1$	is the propagation delay of operation #1,
$Taop2$	is the propagation delay of operation #2,
$nTclk$	is the number of clock cycles required to execute the computational algorithm.

- If $Taop1 + Taop2 < Tclk$ then $nTclk = 1$. (14)

Integer square root algorithm is part of gradient magnitude computations. Its FPGA based functional model implementing low level parallelism is presented in Figure 4 of Appendix.

Integer arithmetic addition has a specific relation to low level parallelism when the number of addends is more than two. Executing addition by using a multistaged two-input adder structure impacts $Tclk$ according to the expression:

$$\begin{aligned} L = P \quad &if \;\; N = 2^P \\ L = P + 1 \quad &if \;\; 2^P < N < 2^{P+1} \end{aligned} \tag{15}$$

where

L	is the number of consecutive levels of adders ,
N	is the number of addends,
$P \in N$.	

From (15) it follows that if for $N = 2$ the period of the system clock is *Tclk*, then for $N > 2$ the period of the system clock becomes *L*Tclk..* Therefore, in that case the use of registers between the levels of adders is a must.

Expressions (15) are used to assess the applicability of a multiplication technique based on the fact that for any $Mr \in N$ one of the following expressions holds true

$$Mr = 2^k + 2^m + 2^n + .. + 2^s \tag{16}$$

where

$k,m,n,s,.. \in N$ and $k > m > n > ..s \geq 1$;

$$Mr = 2^k + 2^m + 2^n + 2^s + .. + 2^0 \tag{17}$$

where

$k,m,n,s,.. \in N$ and $k > m > n > s > .. \geq 1$.

Because maximum image pixel value is $2^8 - 1$, the product P of multiplicand $Md = 2^8 - 1$ and multiplier Mr is one of the following:

- according to (16)

$$P = (2^8 - 1 << k) + (2^8 - 1 << m) + (2^8 - 1 << n) + (2^8 - 1 << s) + .. + (2^8 - 1 << 1) \tag{18}$$

- according to (17)

$$P = (2^8 - 1 << k) + (2^8 - 1 << m) + (2^8 - 1 << n) + (2^8 - 1 << s) + .. + (2^8 - 1). \tag{19}$$

Because the two addends must have equal number of bits and maximum image pixel value is $2^8 - 1$, according to (15) the largest output data width of an adder at level #1 is $2^3 + m$. Hence,

$$\begin{gathered}\text{for expression (18) } L = nShl \\ \text{for expression (19) } L = nShl + 1\end{gathered} \tag{20}$$

where

$nShl$ is the number of shift left operations.

Therefore, maximum clock frequency of this multiplication technique is defined on the basis of the inequality

$$Tclk > T(<< m) + Tadd(2^3 + m) \tag{21}$$

where

$T(<< m)$ is the propagation delay of shift left operation executed by *m* number of bits,

$Tadd(2^3 + m)$ is the propagation delay of the adder with input data widths equal to $2^3 + m$ bits.

The FPGA based functional model of a multiplier implementing low level parallelism is presented in Figure 5 of Appendix.

6. Conclusion

Presented is an in-depth study of parallelism in FPGA based Canny. Defined are top-, high-, mid- and low level of parallelism. The specific use of each level in the different parts of Canny's algorithm is thoroughly investigated. Analyzed is the impact of all levels of parallelism on speed on the basis of particular formulae. The mathematics and speed capabilities of a specific type of FPGA based multiplication is explored in the context of parallelism.

Appendix

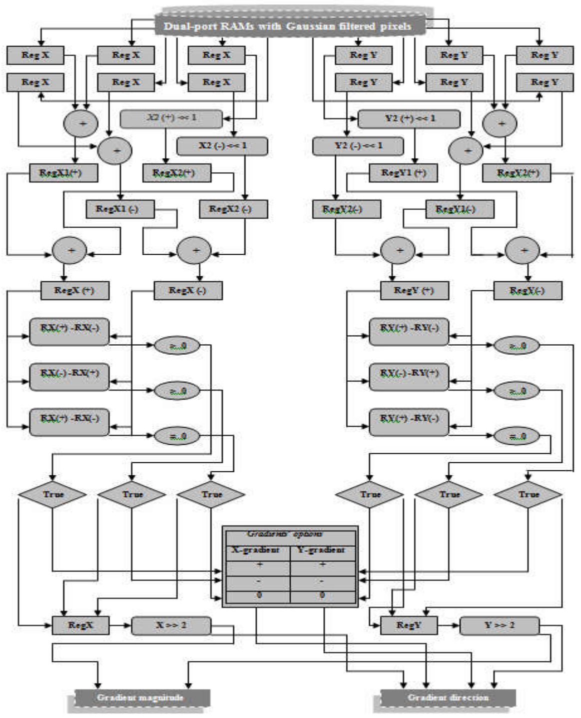


Figure 1. Functional model of Sobel filtering implementing top level parallelism

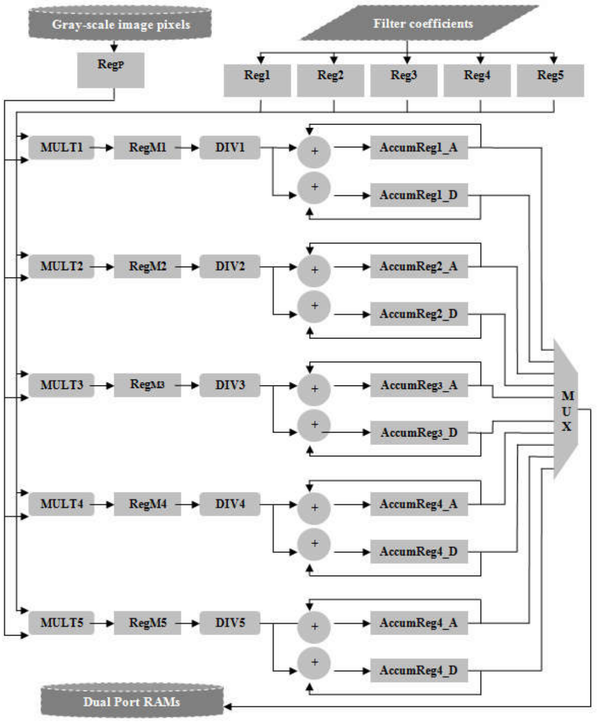
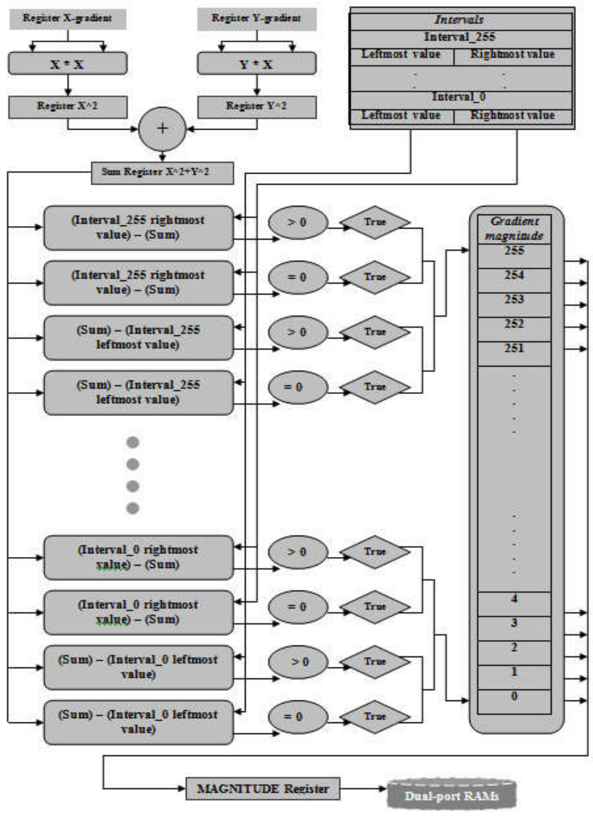
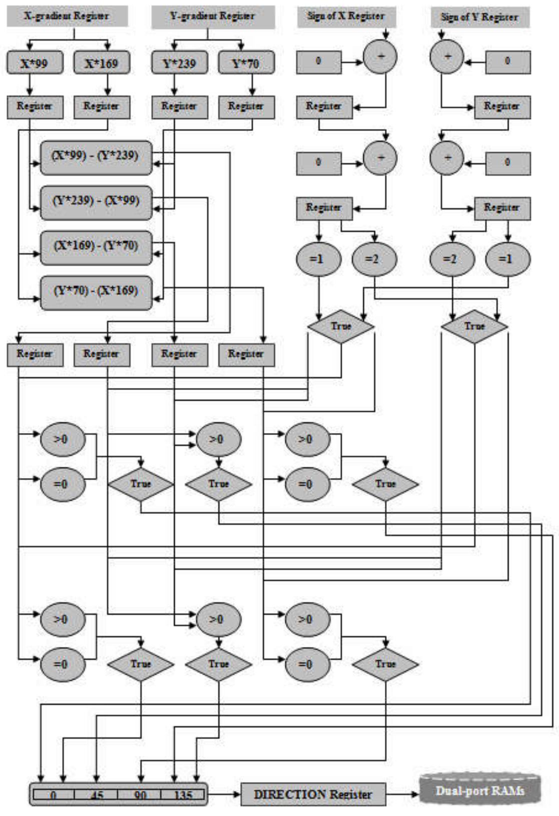


Figure 2. Functional model of Gaussian filtering implementing mid-level parallelism





(a) (b)

Figure 3. Functional model of gradient magnitude – (a), and functional model of gradient direction - (b). Working simultaneously, they implement high level parallelism

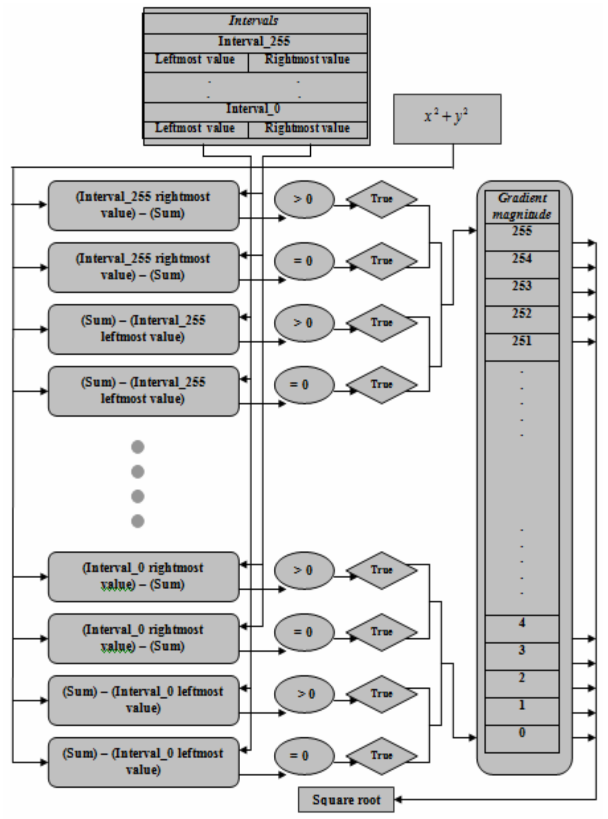

Intervals
Interval_255
Leftmost value
Rightmost value
Interval_0
Leftmost value
Rightmost value
x^2+y^2
(Interval_255 rightmost value) – (Sum)
(Interval_255 rightmost value) – (Sum)
(Sum) – (Interval_255 leftmost value)
(Sum) – (Interval_255 leftmost value)
(Interval_0 rightmost value) – (Sum)
(Interval_0 rightmost value) – (Sum)
(Sum) – (Interval_0 leftmost value)
(Sum) – (Interval_0 leftmost value)
> 0
= 0
True
Gradient magnitude
255
254
253
252
251
4
3
2
1
0
Square root

Figure 4. Functional model of integer square root algorithm implementing low level parallelism

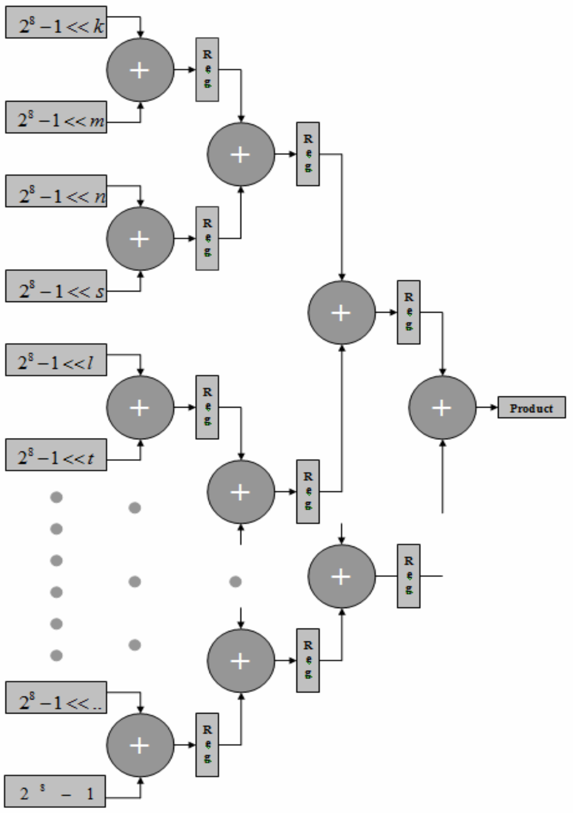


Figure 5. Functional model of multiplier implementing low level parallelism

References

[1] Aravindh G., Manikandababu C. S. 2015 Algprithm and Implementation of Distributed Canny Edge Detector on FPGA *ARPN Journal of Engineering and Applied Sciences* Vol. **10** (7), pp.3208-3216

[2] Chandrashekar N.S., Nataraj K.R. 2013 Image Smoothening Gradient Magnitude and Hystersis Calculation for Canny Edge Detector Using FPGA for Area Optimization *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)* Vol. **2** (2), pp. 5-9

[3] Supraya K. 2017 Hardware Implementation Of Canny Edge Detection Algorithm With FPGA *Journal of Engineering Science and Technology* Vol. **12**, No. 9, pp. 2536-2550

[4] Pallavi Ramgundewar, Hingway S .P., Mankar K. 2015 Design of modified Canny Edge Detector based on FPGA for Portable Device *Journal of The International Association of Advanced Technology and Science* Vol. **16** (2), pp. 10-16

[5] Poonam S. Deokar and Anagha P. Khedkar 2015 Implementation of Modified Distributed Canny Edge Detector Algorithm Using FPGA *International Journal of Information Research and Review* Vol. **2** (8), pp. 999-1003

[6] Veeranagoudapatil, Chitra Prabhu 2015 Distributed Canny Edge Detector:Algorithm & FPGA Implementation, *International Journal for Research in Applied Science & Engineering Technology* Vol. **3** (5), pp. 586-588

[7] Yu Chen, Caixia Deng, Xiaxia Chen 2015 An Improved Canny Edge Detection Algorithm, *International Journal of Hybrid Information Technology*, Vol.**8** (10), pp.359-370