A Dual Fixed Point implementation of Expanded Hyperbolic Cordic Algorithm

Iyad Mansour, Omar Bataineh
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
ifmansour@oakland.edu, ofbataineh@oakland.edu

***Abstract*—This work presents an architecture for Hyberbolic computation in dual fixed point arithmetic that is based on the expanded hyperbolic CORDIC algorithm, where the user can select the 2-D domain of convergence that suits their application. The fully parameterized hardware implementation allows us to explore trade-offs among design parameters (numerical format, number of iterations), resource usage, accuracy, and execution time. We carry out an exhaustive design space exploration and generate Pareto-optimal realizations in the resource-accuracy space. Our approach allows us to select optimal hardware realizations that meet or exceed accuracy requirements.**

Keywords—computer arithmetic, CORDIC, Hyberbolic, dual fixed point.

I. INTRODUCTION

The hyperbolic CORDIC algorithm as originally proposed by Walther [1] allows the computation of hyperbolic functions in an efficient fashion. However, the domain of the inputs is limited in order to guarantee that outputs converge and yield correct values, and this limitation will not satisfy the applications in which nearly the full range of the hyperbolic functions is needed.

Various strategies have been proposed to address the problem of limited convergence of the hyperbolic CORDIC algorithm. One strategy is to use mathematical identities to preprocess the CORDIC input quantities [1]. While such mathematical identities work, there is no single identity that will remove or reduce the limitations of all the functions in the hyperbolic mode. In addition, the mathematical identities are cumbersome to use in hardware applications because their implementation requires a significant increase in processing time and hardware [2]. Another approach, proposed by Hu et al [2], involves a modification to the basic CORDIC algorithm (inclusion of additional iterations) that can be readily implemented in a VLSI architecture or in a FPGA without excessively increasing the processing time.

One architecture for the dual-fixed-point implementation of the hyperbolic CORDIC algorithm with the expansion scheme proposed by Hu [2] is presented, a low cost iterative version.

One numerical formats is proposed for the inputs. For each hyperbolic function, an analysis of this numerical format is performed. Finally an error analysis is performed for each hyperbolic function. The data obtained with the dual-fixed-point architectures are contrasted with the ideal values obtained with MATLAB®.

The rest of this paper is organized as follows: In Section II, the method used for expansion of the hyperbolic CORDIC algorithm is presented. Section III describes the architecture implemented. Section IV presents an analysis of the input numerical format for each hyperbolic function, Section V presents an error analysis. Finally, conclusions and recommendations are given.

II. METHODOLOGY

A. *Original Hyperbolic CORDIC algorithm*

The original hyperbolic CORDIC algorithm, first described by
Walther [1], states the following iterative equations:

$$\begin{aligned} Xi+1 &= Xi + \delta i Yi 2^{-i} \\ Yi+1 &= Yi + \delta i Xi 2^{-i} \\ Zi+1 &= Zi - \delta i \theta i \end{aligned} \quad (1)$$

$$Where\ \theta i = Tanh^{-1}(2^{-i}) \quad (2)$$

And i is the index of the iteration (i = 1, 2, 3, … 32). The following iterations must be repeated in order to guarantee the convergence: 4, 13, 40,… k, 3k + 1. The value of δi is either +1 or –1 depending on the mode of operation:

$$\begin{aligned} &Rotation:\ \delta i = -1\ if\ zi < 0,\ +1\ otherwise \\ &Vectoring:\ \delta i = -1\ if\ xiyi < 0,\ +1\ otherwise \end{aligned} \quad (3)$$

In the rotation mode, the quantities X, Y and Z tend to the following results, for sufficiently large *N*:

$$\begin{aligned} Xn &\leftarrow An[X_0 CoshZ_0 + Y0SinhY0] \\ Yn &\leftarrow An[Y0CoshZ_0 + X_0 SinhY0] \\ Zn &\leftarrow 0 \end{aligned} \quad (4)$$

And, in the vectoring mode, the quantities X, Y and Z tend to the following results, for sufficiently large N:

$$\begin{aligned} Xn &\leftarrow An\sqrt{X_0{}^2 - Y_0{}^2} \\ Yn &\leftarrow 0 \\ Zn &\leftarrow Z_0 + Tanh^{-1}\left(\frac{Y_0}{X_0}\right) \end{aligned} \quad (5)$$

Where 'An' is: $An \leftarrow \prod_{i=1}^{N} \sqrt{1 - 2^{-2i}}$ (6)

With a proper choice of the initial values X0, Y0, Z0 and the operation mode, the following functions can be directly obtained: *Sinh, Cosh, Tanh-1,* and *exp*. Additional functions (e.g. *ln, sqrt, Tanh*) may be generated by applying mathematical identities, performing extra operations and/or using the circular or linear CORDIC algorithms [3].

B. Range of Convergance

The basic range of convergence, obtained by a method developed by Hu[2] states the following:

$$\textit{Rotation Mode: } |Z_0| \le \theta_N + \sum_{1=1}^{N} \theta_i \quad (7)$$

$$|Z_0| \le tanh^{-1}(2^{-N}) + \sum_{1=1}^{N} tanh^{-1}(2^{-i}) \quad (8)$$

$$|Z_0|_{max} \approx 1.182, for\ N \to \infty \quad (9)$$

This is the restriction imposed to the domain of the input argument of the hyperbolic functions in the rotation mode. Note that the domain of the functions *Sinh* and *Cosh is* $\langle -\infty, +\infty \rangle$.

$$\textit{Vectoring Mode: } \left|tanh^{-1}\left(\frac{Y_0}{X_0}\right)\right| \le \theta_N + \sum_{1=1}^{N} \theta_i \quad (10)$$

$$\to \left|tanh^{-1}\left(\frac{Y_0}{X_0}\right)\right| \le 1.1182, for\ N \to \infty \quad (11)$$

$$\to \left|\frac{Y_0}{X_0}\right|_{max} \approx 0.80694, for\ N \to \infty \quad (12)$$

This is the limitation imposed to the domain of the quotient of the input arguments of the hyperbolic functions in the vectoring mode. Note that the domain of *Tanh-1* is $\langle -1, +1 \rangle$, and thus this function remains greatly limited in its domain.

C. Expansion of the Range of Convergence

The convergence range described by (9) and (12) is unsuitable to satisfy all applications of the hyperbolic CORDIC algorithm.

One strategy to address the problem of limited convergence is the use of mathematical identities to preprocess the CORDIC input quantities [1]. However, a different preprocessing scheme is necessary for each function, making it very difficult to have a unified hyperbolic CORDIC hardware. Moreover, the preprocessing leads to a significant increase in processing time and hardware.

Hu et al [2] have proposed another scheme to address the problem of the range of convergence. The approach consists in the inclusion of additional iterations to the basic CORDIC algorithm. As it will be shown in Section 3, the hardware and processing time increase is bearable and suitable for VLSI and FPGA implementation.

The method proposed by Hu consists in the inclusion of additional iterations for negative indexes *i*:

$$\theta_i = tanh^{-1}(1 - 2^{i-2}), for\ i \le 0 \quad (13)$$

Therefore, the modified algorithm results:

For $i \le 0$

$$\begin{aligned} X_{i+1} &= Xi + \delta i(1 - 2^{i-2})Yi \\ Y_{i+1} &= Yi + \delta i(1 - 2^{i-2})Xi \\ Z_{i+1} &= Zi - \delta i Tanh^{-1}(1 - 2^{i-2}) \end{aligned} \quad (14)$$

For $i > 0$

$$\begin{aligned} X_{i+1} &= Xi + \delta i Yi 2^{-i} \\ Y_{i+1} &= Yi + \delta i Xi 2^{-i} \\ Z_{i+1} &= Zi - \delta i Tanh^{-1}\, 2^{-i} \end{aligned} \quad (15)$$

The trend of the results for the rotation and vectoring mode is the same as that stated in (4) and (5). The value of δi is the same as indicated in (3). But the quantity A_n, described in (6), must be redefined as follows:

$$A_n \leftarrow \left[\prod_{i=-M}^{0} \sqrt{1 - (1 - 2^{(1-2)})^2}\right]\left[\prod_{i=1}^{N} \sqrt{1 - 2^{-2i}}\right] \quad (16)$$

The range of convergence, stated in (7) and (10) for the basic hyperbolic CORDIC algorithm, now becomes:

$$\textit{Rotation Mode: } |Z_o| \le \theta_{max} \quad (17)$$

$$\textit{Vectoring Mode: } \left|Tanh^{-1}\left({}^{Y_0}/_{X_0}\right)\right| \le \theta_{max} \quad (18)$$

Where:

$$\theta_{max} = \sum_{i=-M}^{0} Tanh^{-1}(1 - 2^{i-2}) + \left[Tanh^{-1}(2^{-N}) + \sum_{i=i}^{N} Tanh^{-1}(2^{-i})\right] \quad (19)$$

Although (17) and (18) look nearly the same, they are interpreted differently: (17) states the maximum input angle the user can enter to obtain a valid result, whereas (18) states the maximum value attainable for the $Tanh^{-1}$ function to which $Z\text{-}Z_0$ tends (according to (5)). If Z_0=0, (18) states the maximum value attainable at Z, and therefore imposes a limitation to the inputs X_0 and Y0.

The values for θ_{max} have been tabulated for M between 0 and 10 and are shown in Table 1.

M	θ_{max} from(19)
0	2.09113
1	3.44515
2	5.16215
3	7.23371
4	9.65581
5	12.42644

Table 1: θ_{max} versus M for the Modified Hyperbolic CORDIC algorithm (after Hu[2])

For example, if M = 5 is chosen (six additional iterations), then θ_{max}=*12.42644*, and the domain of the functions Cosh and Sinh is greatly expanded to [-12.42644,+12.42644] compared with the domain in (9). Similarly, the range of the function Tanh-1 is increased to [-12.42644,+12.42644], which means that the domain of the quotient *Y0/X0* becomes nearly $\langle -1, +1 \rangle$, which is the entire domain of Tanh-1. From the last

example, it is clear that the expansion scheme does work. The more domain of the functions is needed, the more the iterations (M+1) that must be executed.

III. ARCHITECTURE

The architecture presented here implement the expanded hyperbolic CORDIC algorithm described in (14) and (15). The architecture is such that the inputs and outputs have an identical bit width. The intermediate registers and operators can be of higher bit width due to particular details of the algorithm and precision considerations which will be explored later in this paper. In Section IV, we explore the particularities of the architecture for $Tanh^{-1}$, Sinh, Cosh, and exp. It is worth to note, however, that a unified hyperbolic ORDIC hardware, capable of obtaining all the functions within the same architecture, is desirable for certain applications, as has been shown in [5]. The same principle which will be applied to the analysis of $Tanh^{-1}$, Sinh, Cosh and exp can be applied to this case and thus the optimum architecture can be attained. In addition, there exists a precision consideration which extends the bit width: it is a 'rule of thumb' found in [5]: "If n bits is the desired output precision, the internal registers should have log2(n) additional guard bits at the LSB position". This consideration, although arbitrary, have proved to work very well. With these considerations in mind, one dual-fixed-point architectures is presented, which is a low cost iterative version.

Fig. 1 depicts the architecture that implements the equations (14) and (15) in an iterative fashion. The two LUTs (look-up tables) are needed to store the two sets of elementary angles defined in equations (2) and (13). The process begins when a start signal is asserted. After '*M+1+N+v*' clock cycles ('*v*' is the number of repeated iterations stated in Section II.A), the result is obtained in the registers *X*, *Y* and *Z*, and a new process can be started.

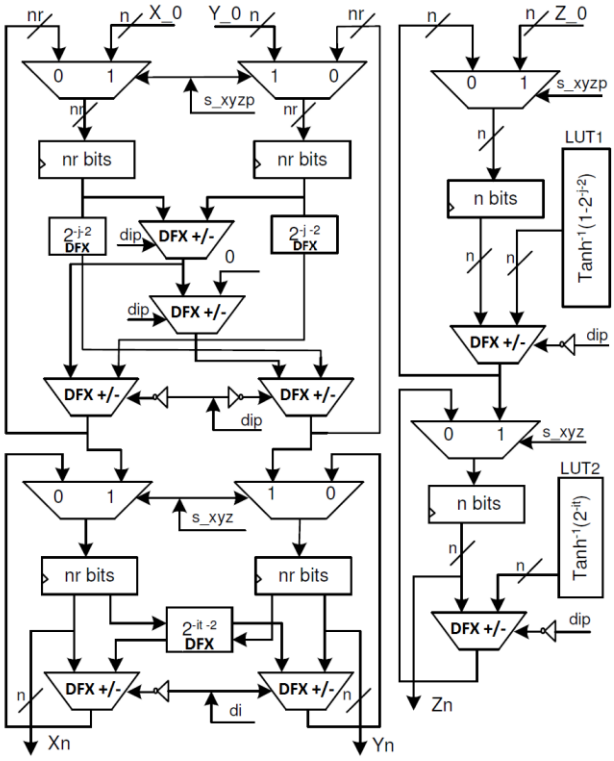

Figure 1

Inputs: X_0, Y_0, and Z_0
Outputs: X_N, Y_N, and Z_N
j = -5→ 0 it = 1→32

There are two stages: One that implements the iterations for $i \leq 0$ and is depicted in the upper part, it needs two multiplexers, two registers, four adders and two barrel shifters. This is the most critical part of the design, and introduces considerable delay, thus reducing the frequency of operation. The lower part of Figure (1) implements the iterations for $i > 0$, this is a classical hardware found in many textbooks and papers.

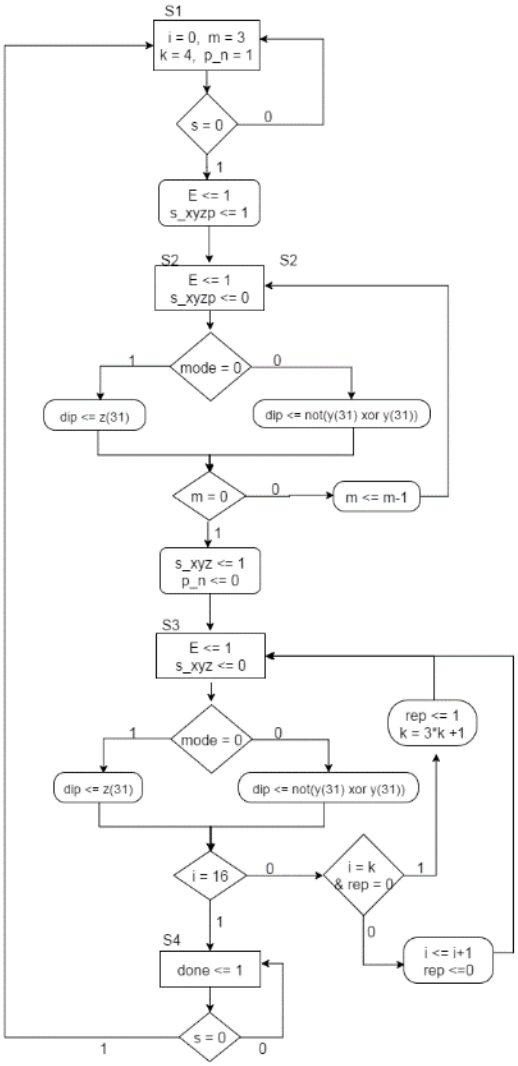

Figure 2

Figure (2) shows a state machine that controls the load of the registers, the data that passes onto the multiplexers, the add/subtract decision of the adder/substracters, and the count given to the barrel shifters.

IV. EXPERIMENTAL SETUP

For our dual fixed-point (DFX) hardware, we used the Vivado IDE from Xilinx to simulate our hardware, in addition to that an actual hardware implementation have been implemented on the Zypo board (Zynq architecture + ARM microprocessor).

Our design implements the Tanh-1, Cosh, Sinh and Exp. each of the mentioned functions has a different setup to get the desired accuracy as disciped in the following sections.

A. Inverse Hyperbolic Tangent ($Tanh^{-1}$)

To obtain the $Tanh^{-1}$ function in the output Z, we have to set $Z_0 = 0$, and $X_0 = 1$, and the operational mode to Vectoring mode, Then after the appropriate iterations the $Z_N \leftarrow Tanh^{-1}(Y_0)$.

Since the domain of the $Tanh^{-1}$ is (-1, 1) the input Y_0 is restricted to 1 integer bit in the 2's complement fractional dual fixed point representation, ($|Y_0| < 1$), But, as the input $X_0 = 1$ requires 2 integer bits for correct representation, and the format for X and Y must be the same, then X and Y must have 2 integer bits. And since the DFX representation have been used, one more bit is needed as an exponent (Num0, Num1 selection) so the final format used to represent X and Y is selected to be [N P_0 P_1] = [32 29 25]

The critical case occurs when Y_0 is at its maximum value, from which the maximum value of Z_N is obtained. And since we are using M=5 negative iteration in our design the max to be represented in Z equal to 12.42644 which needs at least 5 integer bits in the 2's complement fractional DFX representation, and to get the two format as an output (the Num0, and Num1) the following format is selected to represent the Z [N P_0 P_1] = [32 27 25] which will cover the range [-8, 8) as Num0 and [-32, 32) as Num1.

B. Hyperbolic Sine and Hyperbolic Cosine.

To obtain the value of the Sinh and Cosh functions in the X and Y output, we have to set $Y_0 = 0$, and $X_0 = 1$, and the operational mode to Rotational mode, Then after the appropriate iterations the $X_N \leftarrow$ An Cosh(Z_0) and $Y_N \leftarrow$ An Sinh(Z_0).

Since the domain of the Sinh and Cosh is ($-\infty$, ∞) there is no input restriction. So we select the format [32 27 25] for Z in the DFX which will give the following two range:

[-8, 8) if Num0
[-32, 32) if Num1

To cover test cases from the Num0 and Num1 regions we select the input Z to be (-11, 11).

The critical case occurs when |Z0| is at its maximum value = 11 (in our selection), that will give a values for X and Y to be less than 16, so the format [32 29 25] for X and Y is selected to cover the range selected for the input Z0.

The range for X and Y will be

[-2, 2) if Num0
[-32, 32) if Num1

C. Exponential (e^x)

To obtain the value of the exponential functions in the X and Y output, we have to set $Y_0 = X_0 = 1$, and the operational mode to Rotational mode, Then after the appropriate iterations the $X_N \leftarrow$ An e^{Z0} and $Y_N \leftarrow$ A e^{Z0}

Since the domain of the e^x is ($-\infty$, ∞) there is no input restriction. So we select the format [32 27 25] for Z in the DFX which will give the following two range:

[-8, 8) if Num0
[-32, 32) if Num1

To cover test cases from the Num0 and Num1 regions we select the input Z to be (-10, 10).

The critical case occurs when |Z0| is at its maximum value = 10 (in our selection), that will give a values for X and Y to be less than 12, so the format [32 29 25] for X and Y is selected to cover the range selected for the input Z0.
The range for X and Y will be

[-2, 2) if Num0
[-32, 32) if Num1

V. RESULTS

To measure the accuracy of the implemented DFX hyperbolic CORDIC an error analysis is performed on a different cases that will cover the two formats in the DFX representation (num0, num1), the results are compared with the ideal values obtained in MATLAB.

The error measures will be:

$$Relative\ Error = \frac{|\,ideal\ value - DFX\ CORDIC\ value|}{|ideal\ value|}$$

The following subsections describes the detailed error analysis for each implemented function.

A. Inverse Hyperbolic Tangent ($Tanh^{-1}$)

Figure (3) show the relative error for the $Tanh^{-1}$ on the entire domain of the $Tanh^{-1}$ (-1, 1), although the domain of $Tanh^{-1}$ is (-1, 1) we have just plotted for [0, 1) since $Tanh^{-1}$ is an odd function,
We have taken 1000 value equally spaced along the domain (step = 0.001)

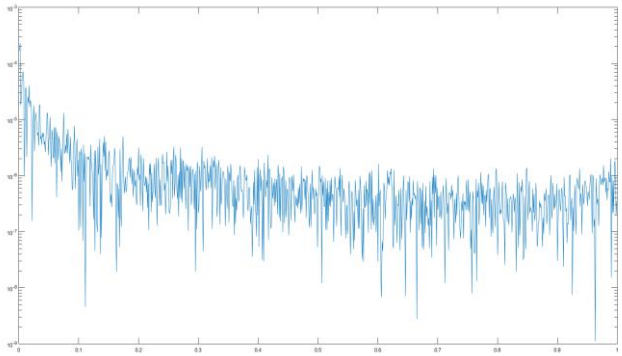

Figure 3

Figure (4) show the relative error for the Tanh-1 on the range [0.999999, 0.999999998], this range have been selected such that the results will cover the two DFX formats Num0 and Num1.
We have taken 1000 value equally spaced along the domain (step = 0.001)

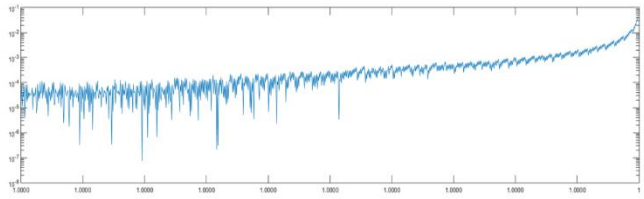

Figure 4

B. Hyperbolic Sine and Hyperbolic Cosine.

Figure (5) show the relative error for the Sinh on the range [0, 11] this range have been selected such that the results will cover the two DFX formats Num0 and Num1.

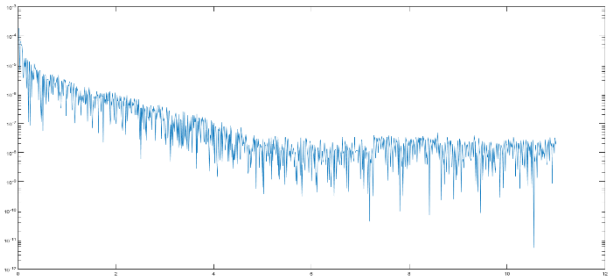

Figure 5

Figure (6) show the relative error for the Cosh on the range [0, 11] this range have been selected such that the results will cover the two DFX formats Num0 and Num1.

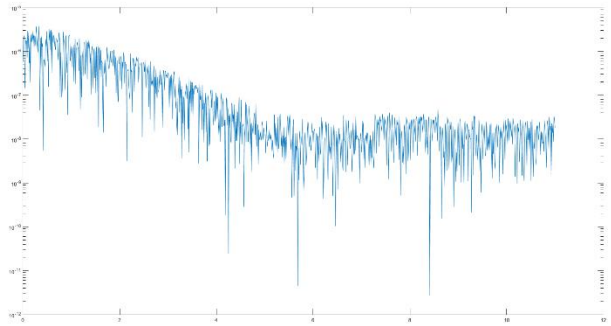

Figure 6

C. Exponential (e^x)

Figure (7) show the relative error for the e^x on the range [0, 10] this range have been selected such that the results will cover the two DFX formats Num0 and Num1.

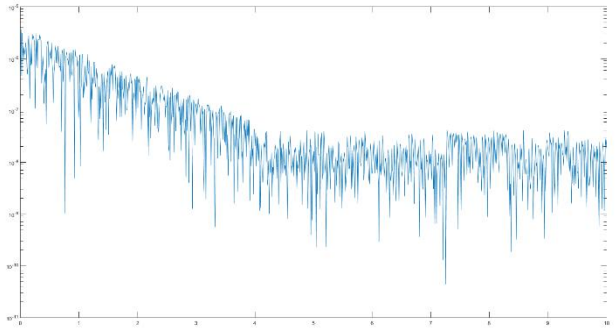

Figure 7

CONCLUSIONS

• The expansion scheme proposed by Hu[2], despite the additional hardware needed, has proved to be amenable for our FPGA implementation, as the clock rate and resource effort indicates. The function *Tanh-1* gets expanded in all its domain, and the functions *Cosh* and *Sinh* have a greater domain as the bit width increases.

• The error analysis shows certain irregularities in the relative error performance. This irregularities are due to the truncation of the fractional bits, conversion between decimal and binary and the ever-limited number of basic and additional iterations. We have tested the CORDIC algorithm in MATLAB® and have found that the error performance is uniform.

• Parametrized VHDL design can be implemented to test more than format and compare each format on each Hyberbolic function.

•Designing a dual-fixed-point adder/subtraector which recovers truncated fractional bits can enhance the output which yields to minimal error.

REFERENCES

[1] D. Llamocca, C. Agurto, "A Fixed-point implementation of the expanded hyperbolic CORDIC algorithm," *Latin American Applied Research*, vol. 37, no. 1, pp. 83-91, Jan. 2007

[2] X. Hu, R.G. Harber, S.C. Bass, "Expanding the range of convergence of the CORDIC algorithm," IEEE Transactions on Computers, vol. 40, no. 1, pp. 13-21, Jan. 1991.

[3] J. Becker, M. Platzner, S. Vernalde (Eds.): FPL 2004, LNCS 3203, pp. 200–208, 2004. c Springer-Verlag Berlin Heidelberg 200