Fakultät Informatik

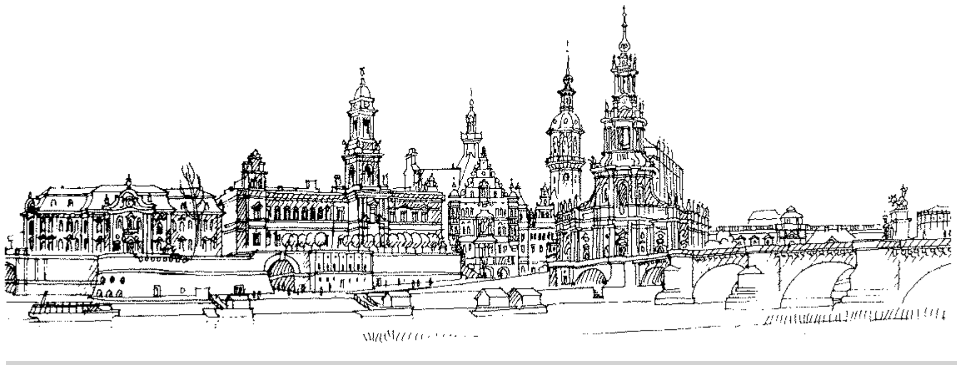
Technische Berichte
Technical Reports
ISSN 1430-211X

TUD-FI05-07 Juni 2005

Thomas B. Preußer
Steffen Köhler

Institut für Technische Informatik

Discrete Fractional Clock Generation for Systems-on-FPGA



Technische Universität Dresden
Fakultät Informatik
D−01062 Dresden
Germany
URL: http://www.inf.tu−dresden.de/

Discrete Fractional Clock Generation for Systems-on-FPGA

Thomas B. Preußer and Steffen Köhler

Dresden University of Technology, Dresden, Germany
{preusser,stk}@ite.inf.tu-dresden.de

Abstract. *This article describes an inexpensive way of clock generation for FPGA-based circuit cores, which reduces the number of external clock sources and eases synchronization problems. We introduce a modified version of the* BRESENHAM *line drawing algorithm and use it outside its original application domain for the rational division of clocks. An optimized hardware design for* BRESENHAM*-based clock division is presented and the quality of its output is evaluated. The optimal initialization conditions in terms of phase shift and jitter are identified and formally proven. Finally, the complexity characteristics of a generic synthesizable VHDL design based on this algorithm are examined and verified by synthesis examples. Special attention is paid to implementation results in conjunction with different FPGA families.*

Keywords
BRESENHAM Algorithm, Clock Division

1 Introduction

In a system-on-FPGA [Alte] consisting of several circuit cores, it is not uncommon for each core to have its own clocking requirements. This is especially true for peripheral interface cores as their clock frequencies are often tailored to application-specific criteria, such as data rates or sampling intervals. In the past, external clock generator circuits have been applied to fulfill the requirement for a flexible circuit clocking. One major drawback of this approach that introduces new clock domains is the requirement of expensive cross-clock-domain FIFO cores for synchronization reasons. To overcome this, the circuit clocks can be derived from the system clock. Maximum clock accuracy in terms of phase jitter as well as low FPGA resource consumption can be assured through the implementation of a fractional clock divider using the BRESENHAM algorithm. Additionally, external clock generator circuits are no longer required.

The BRESENHAM algorithm is an ubiquitous algorithm in computer graphics as it provides a fast incremental interpolation scheme originally used for line plotting. Its major advantages are the elimination of expensive multiplications and divisions as well as the numerical scaling to integer-only arithmetic. The first implementation of this algorithm is due to Jack E. BRESENHAM and was written for an IBM 1401 controlling a plotter [Bre65].

```
float const  m = (y1-y0)/(x1-x0); // slope of line

int     y = y0;        // initializing y accumulation
float   e =  0;        // current (still fractional) y-error
for(int  x = x0; x <= x1; x++) {
  putPixel(x, y);      // put current pixel
  e += m;              // y-increment for a unit x-step
  if(e > 0.5) {        // would next y-pixel be closer?
    e -= 1.0;
    y++;
  }
}
```

Listing 1: Basic BRESENHAM Algorithm

Establishing a universal interpolation scheme, the algorithmic idea behind the BRESENHAM algorithm is easily generalized to other domains beyond computer graphics. In particular, this algorithm can be applied for synthetic clock division. It enables the generation of any clock representable as a rational fraction of a reference clock with a minimum phase jitter according to the resolution of the reference.

This article revises the BRESENHAM algorithm in its original context for line plotting. It will then generalize this approach to the implementation of synthetic clock division and identify optimization potential arising in this domain of application. The initialization condition for an optimal approximation of the desired clock division will be presented and proven. The paper will close with the presentation of synthesis results obtained for a choice of FPGA architectures from a generic synthesizable VHDL model for the BRESENHAM clock divider.

2 Algorithm Fundamentals

The original application domain of the BRESENHAM algorithm is the approximate plotting of straight lines on rastered devices. FPGA-accelerated rendering for desktop publishing utilizing the BRESENHAM algorithm has already been evaluated, e.g. in [MS98].

Given the coordinates of the start and end points of the line to be drawn, (x_0, y_0) and (x_1, y_1), respectively, the coordinate growing faster for this line must be determined first as to ensure a connected plot. Without loss of generality, assume x to be the faster growing coordinate. Further, let, for the purpose of illustration, $x_0 \leq x_1$ and $y_0 \leq y_1$ such that all increments while iterating from the start to the end point of the line become non-negative.

The pixels to color as part of the line to be plotted are now chosen as follows: A variable x is iterated over the pixel coordinates x_0 through x_1. Doing so a variable y initialized to y_0 is only incremented when the deviation from the ideal, usually fractional, y position is thereby reduced. By further calculating the error of the current y position incrementally, all multiplication, division and rounding operations as implied

```
int const  dx2 = 2*(x1 - x0);
int const  dy2 = 2*(y1 - y0);

int  y = y0;          // initializing y accumulation
int  e = x1 - x0;
for(int  x = x0; x <= x1; x++) {
  putPixel(x, y);   // put current pixel
  e += dy2;
  if(e > dx2) {      // would next y-pixel be closer?
    e -= dx2;
    y++;
  }
}
```

Listing 2: Integer-only BRESENHAM for Line Drawing

by the equation of the ideal line are eliminated in favor of incremental calculations. The resulting algorithm can then be stated as in listing 1.

Further shifting the accumulated error e up by one half eliminates negative error values, and scaling e and the slope m by $2 \cdot dx = 2(x1 - x0)$ results in integer-only arithmetic as illustrated in listing 2.

Finally, it should be noted that the BRESENHAM algorithm for line drawing utilizes three state variables. x holds the current x coordinate during the iteration through the relevant grid points, y and e together represent the corresponding exact y coordinate where y holds the approximation of the plotted pixel.

3 Clock Division

As already mentioned, the BRESENHAM algorithm can be modified for the rational division of clocks. To achieve this, the iteration over x is replaced by an endless loop and only the parity (*being odd or even*) of y is used to determine the state of the generated divided clock. Thus, one loop iteration – now without an x increment – represents one cycle of the input clock whereas two increments of y represent a complete cycle of the divided clock. Consequently, the requirement of y to be the coordinate growing more slowly translates to the restriction that the generated clock can have at most half the frequency of the input clock.

There are a few opportunities for the optimization of the original algorithm in the context of clock division. In particular, the state variable x is fully eliminated and the variable y is reduced to a single bit being alternated instead of incremented. Additionally, the inputs to the algorithm are no longer start and end points of a line but enumerator and denominator of the fraction $\frac{P}{Q}$ the input clock is scaled by.

As will be discussed later, the initial value of the accumulated error e is not critical for the periodic behavior of the generated clock signal. Thus, an initial even value may be assumed. Noting that the inner loop of the algorithm only applies additive manipulation by even values to e allows for a scaling of all involved values in half. Recalling

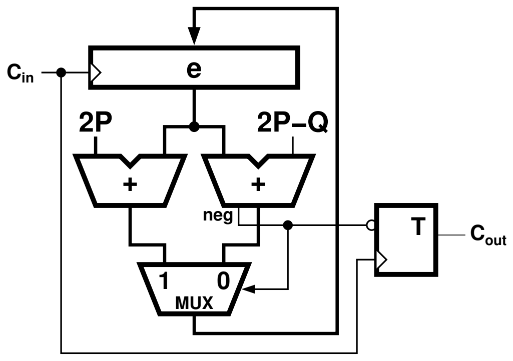


Fig. 1. BRESENHAM Clock Division

that two increments of y are required for a complete cycle of the output clock, $2 \cdot dy$ is scaled down to dy and substituted by $2 \cdot P$ whereas $2 \cdot dx$ is scaled down to dx and substituted by Q. The resulting algorithm can then be cast into structure as depicted in figure 1.

It should be noted that synthesizing this structure for a known fraction $\frac{P}{Q}$ yields further potential for optimization. Arithmetic compaction of the constant adders is easily performed by modern synthesis tools. Another minor reduction in hardware complexity can be achieved when the parity of Q is taken into account. If Q is even, the lowest significant bit of e and all the intermediate arithmetic will never change and can be eliminated. Otherwise, if Q is odd, this bit will only toggle when Q is actually subtracted and the output clock can thus be directly derived from the lowest significant bit of e.

Although the algorithm works for any naturals P and Q as long as $2 \cdot P \leq Q$, it is advisable to ensure that P and Q have no common factor. Dividing both input parameters by their greatest common factor $\mathrm{gcf}(P, Q)$ results in an equivalent output behavior but is likely to reduce the generated logic. Most synthesizers will only be capable of performing this reduction when the common factor is a power of two and can thus easily be identified by constant lowest significant bits within the logic.

Assuming P and Q are free from common factors, the initialization condition inherited from the line drawing algorithm for the e register can be relaxed. The addition of $2P$ and conditional subtraction of Q is easily identified as addition modulo Q. When Q is odd, $2P$ and Q are free from common factors since P and Q are. Consequently, the subsequent addition of $2P \pmod{Q}$ will cycle the remainder held in e through all Q naturals in $[0, Q)$. Likewise, e will cycle through all $\frac{Q}{2}$ even or odd naturals in $[0, Q)$ when Q is even – never touching the least significant bit. As a consequence, the initial value of e does not matter in terms of the periodic behavior of the approximated generated clock.

The BRESENHAM clock division ensures that there is no long-term phase drift as the phase error is accumulated to zero over $\frac{Q}{\mathrm{gcf}(Q,2P)}$ cycles. During such a cycle, however,

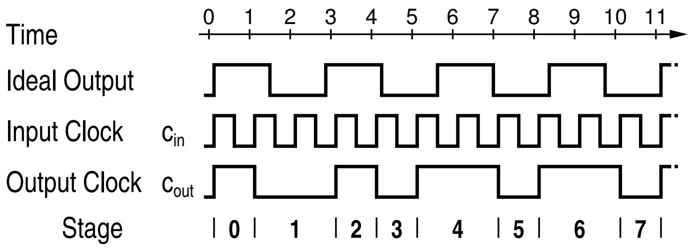


Fig. 2. Waveform for Clock Scaling by $\frac{4}{11}$

some jitter is incurred on the output clock as its edges are merely approximated at edges of the input clock. It is thus vital to evaluate the quality of the generated clock signal.

From the previous discussion, the periodic behavior of the clock division is independent from the choice of the initial value of e. So assume e to be initialized to $e_{\text{init}} = Q - P$. Further, let a single cycle of the input clock serve as a time unit and call a run of subsequent zeros or ones on the output clock a stage. Let the first stage numbered 0 begin at time $t = 0$ on a rising edge of the input clock. Any transition thereafter on the output clock will start a new stage. It is now proven that any edge generated for the output clock occurs at its ideal edge position rounded to the closest edge of the input clock. The situation is illustrated for a $\frac{4}{11}$ scaling in figure 2.

For determining the edges of the generated clock, note that e is loaded with the value $e_i = (2i + 1) \cdot P \pmod{Q}$ at time i. The number of the corresponding stage is determined by counting the edges of the output clock since time $t = 0$. As an edge is generated if and only if the addition of $2P$ modulo Q wraps, the stage number σ_i is determined as:

$$\sigma_i = \left\lfloor \frac{(2i+1) \cdot P}{Q} \right\rfloor$$

Substituting $f := \frac{2P}{Q}$ with rational $0 < f \leq 1$ (from $2P \leq Q$) yields:

$$\sigma_i = \left\lfloor \left(i + \frac{1}{2}\right) f \right\rfloor$$

An edge occurs at time i if and only if a new stage is started, i.e.:

$$\begin{aligned}
1 &= \sigma_i - \sigma_{i-1} \\
&= \left\lfloor \left(i + \tfrac{1}{2}\right) f \right\rfloor - \left\lfloor \left(i - \tfrac{1}{2}\right) f \right\rfloor \\
\Leftrightarrow &\ \exists n \in N.\ \left(i - \tfrac{1}{2}\right) f < n \leq \left(i + \tfrac{1}{2}\right) f \\
&\qquad i - \tfrac{1}{2} < \tfrac{n}{f} \leq i + \tfrac{1}{2} \\
&\qquad \tfrac{n}{f} - \tfrac{1}{2} \leq i < \tfrac{n}{f} + \tfrac{1}{2} \\
\Leftrightarrow i &= \left\lceil \tfrac{n}{f} - \tfrac{1}{2} \right\rceil \\
&= \left\lceil n \cdot \tfrac{Q}{2P} - \tfrac{1}{2} \right\rceil \\
&= \text{round}_{\text{halves\_down}} \left(n \cdot \tfrac{Q}{2P} \right)
\end{aligned}$$

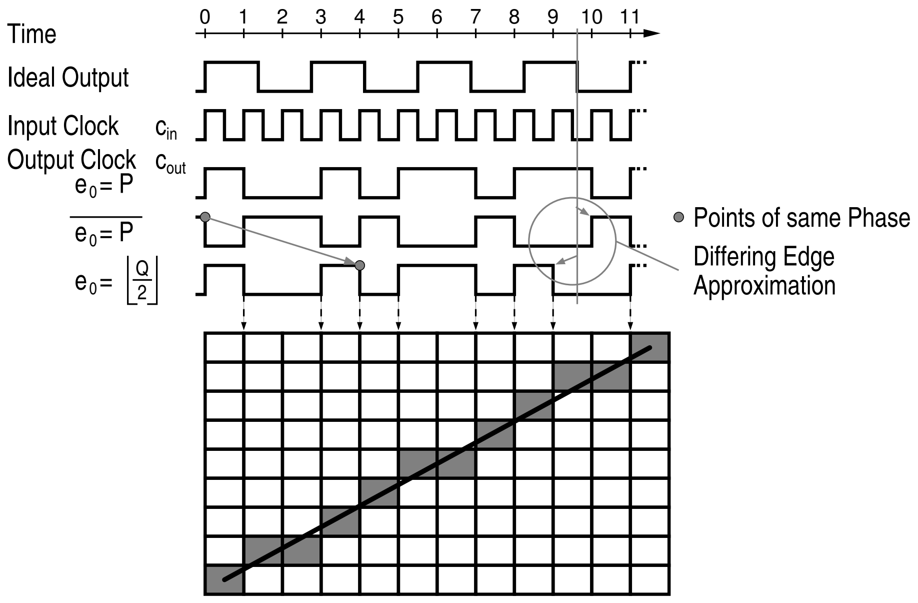


Fig. 3. Initialization and Phase Correspondence

$$(*)$$

The times of the edges of the ideal output clocks are given by $n \cdot \frac{Q}{2P}$ for naturals n. Thus, it follows from $(*)$ that the actually generated clock is optimal in terms of the resolution of the available input clock. The deviation of the generated edges from their optimal point in time is bounded by half a cycle of the input clock.

It should be recalled that the derivation of $(*)$ used the assumption that e be initialized to $Q - P$. As discussed before, using any other initial value will produce the same periodic behavior and thus the same relative jitter of the generated clock signal. Other initial values will, however, yield clock signals that are usually only optimal in the above sense for a different phase.

Consider figure 3 contrasting initializations of e to $Q - P$ and $\left\lfloor \frac{Q}{2} \right\rfloor - 2P$ for the previous example scaling the input by $\frac{4}{11}$. These cases are distinguished by their accumulated errors e_0 in stage 0. The second is chosen such that $e_0 = \left\lfloor \frac{Q}{2} \right\rfloor$, the initial value corresponding to the line drawing algorithm as illustrated immediately below the waveform. It should be noted that in this case, each edge of the generated output clock corresponds to an increment of the y-coordinate.

So far, it has been silently assumed that the initial state of the output clock is $\mathrm{c_{out}} = 0$. As the BRESENHAM algorithm is merely used to determine the times of the clock edges, this assumption is arbitrary. In fact, one of the examples given in figure 3 must be viewed inverted to make the phase difference of the BRESENHAM edge generation correspond directly to a phase difference of the clock signals.

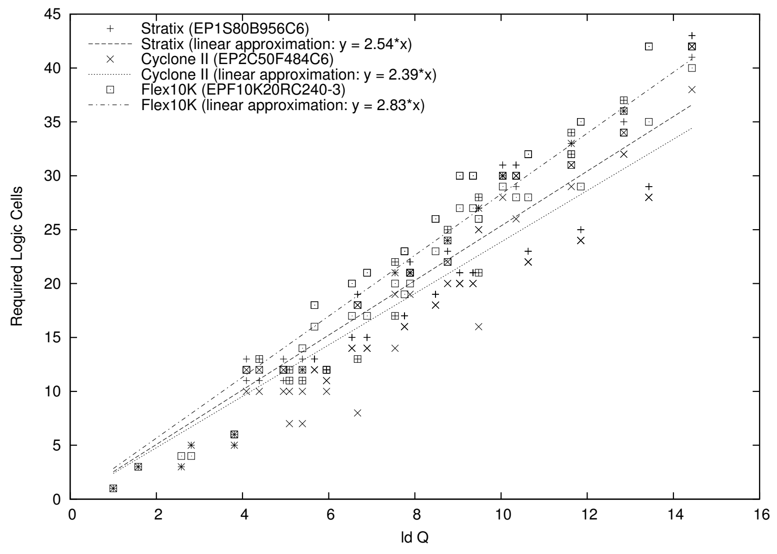


Fig. 4. Complexity of Synthesis Results

There might arise the question why the optimum initialization is different for the line drawing and the clock division applications. The resulting difference is exemplified by the different approximation of the ideal clock edge between times $t = 9$ and $t = 10$. For line drawing, $t = 9$ is chosen as it is closer to the following ideal edge than to the preceding one. For clock generation, $t = 10$ is chosen as this is the closest approximate available.

Finally, it should be noted that the bound on the deviation of each individual output edge immediately implies that the difference of the output stages in duration can be at most one cycle of the input clock. This property is used in the AnyClock$^{\text{TM}}$ chips by Micrel [Mic]. They apply a BRESENHAM-based approach to switch between appropriate $\frac{1}{N}$ and $\frac{1}{N-1}$ divisions to approximate a clock division by a rational between $N - 1$ and N.

4 Design Synthesis

The described BRESENHAM algorithm for clock division has been implemented by a generic synthesizable VHDL model. All optimizations identified for this application domain have been utilized. Assuming linear complexities for the two constant adders, the 2:1 multiplexer as well as the e register, the complexity of the BRESENHAM clock divider also grows linearly with the number of binary digits required for the representation of Q.

The VHDL design has been synthesized for different Altera FPGA architectures [Altd,Alta,Altb] by the Quartus II design software, version 4.2. The synthesis results are summarized in figure 4 confirming the anticipated linear dependency of the design

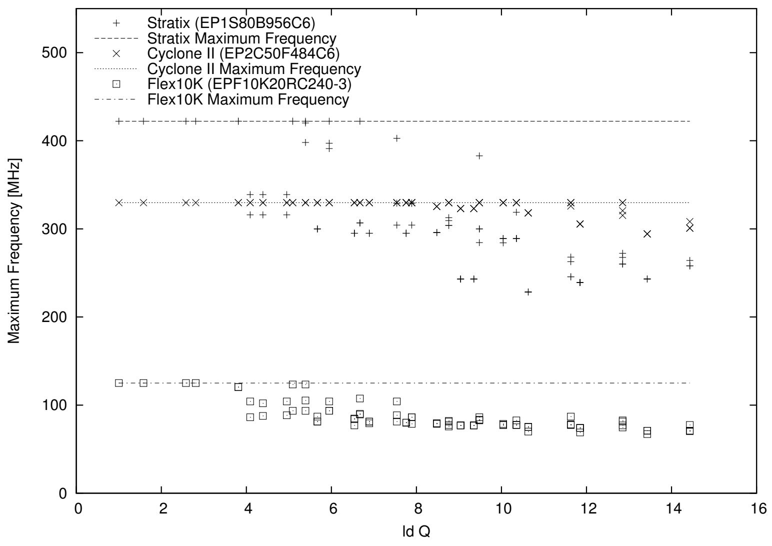


Fig. 5. Maximum Frequencies of Designs

complexity of the bitwidth of Q. The results shown were obtained for $Q = \frac{q}{\mathrm{gcf}(p,q)}$ and $P = \frac{p}{\mathrm{gcf}(p,q)}$ where $q = 2 * 3 * 7 * 17 * 33 = 23562$ and p is one of the elementwise products of $\{1, 5, 19, 97\} \times \mathcal{P}\{2, 3, 7, 17, 33\}$ with $\mathcal{P}$ identifying the power set. Note that the x-axis in figure 4 does not represent the exact bit width of Q but the dual logarithm of Q.

It can be observed that the design complexity in terms of logic cells is approximately $3 \cdot \mathrm{ld}\, Q$. This factor is achieved as multiplexer and e register can be implemented inside the same set of logic cells. In some cases for the Cyclone II and Stratix devices, the synthesizer even managed to integrate one of the adders with the multiplexer and e register by utilizing a synchronous load signal to selectively load the local or the external addition result from the other adder. Thus, the design complexity is occasionally only of the order $2 \cdot \mathrm{ld}\, Q$.

The overall linear trend is interrupted by an upwards step at a bit width of 4. This is likely due to the fact that the logic cells of the considered FPGAs support up to four distinct inputs. The other derivations from the linear complexity dependency are primarily due to simplifications possible by using constant parameters. Furthermore, $\mathrm{ld}\, Q$ is just an approximate of Q's bit width, which must be integer and thus is usually slightly larger.

Figure 5 displays the maximum frequencies of the input clocks achieved for the BRESENHAM dividers for the same selection of parameters as above. For small bit widths, this frequency regularly coincides with the maximum frequency the device itself supports. The performance deterioration on the Stratix and Flex10K devices becomes apparent already at a bit width of 4 whereas the Cyclone II device is further driven at its limit. Altogether, input clock frequencies well above 100 MHz are supported except for the older Flex10K device family.

5 Conclusions

This paper explored the adaptation of the BRESENHAM line drawing algorithm for rational clock division. The optimality of this adaptation in terms of the error of the approximated generated clock edges has been proven. The linear dependency of the structural complexity from the bit width of the fractional denominator has been justified and verified by several synthesis examples. Compared to available non-binary clock generation concepts (e.g. NCO [Altc]), the BRESENHAM approach is especially advantageous in terms of logic cell consumption and achievable reference clock frequency. The encapsulation of the rational clock divider into a parameterizable IP core will be a subject of further investigations.

References

[Alta] Altera Corp., 101 Innovation Drive, San José, CA 95134, USA. *Cyclone II Device Family Data Sheet*. http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1_01.pdf.

[Altb] Altera Corp., 101 Innovation Drive, San José, CA 95134, USA. *Flex 10K Embedded Programmable Logic Device Family*. http://www.altera.com/literature/ds/dsf10k.pdf.

[Altc] Altera Corp., 101 Innovation Drive, San José, CA 95134, USA. *The NCO Compiler V2.2.0*. http://www.altera.com/products/ip/dsp/signal_generation/m-alt-ncocompiler.html.

[Altd] Altera Corp., 101 Innovation Drive, San José, CA 95134, USA. *Stratix Device Handbook*. http://www.altera.com/literature/hb/stx/stratix_handbook.pdf.

[Alte] Altium Ltd., 12A Rodborough Rd, Frenchs Forest NSW 2086, Australia. *System-on-FPGA*. http://www.qa-talk.com/news/alt/alt107.html.

[Bre65] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, Vol. 4(No. 1):Pages 25 – 30, 1965.

[Mic] Micrel, Inc., 1849 Fortune Drive, San José, CA 95131, USA. *3.3V AnyClock^{TM} Fractional N Synthesizer*. http://www.micrel.com/_PDF/HBW/sy87729l.pdf.

[MS98] Donald MacVicar and Satnam Singh. Accelerating dtp with reconfigurable computing engines. In *FPL '98: Proceedings of the 8th International Workshop on Field-Programmable Logic and Applications, From FPGAs to Computing Paradigm*, pages 391–395. Springer-Verlag, 1998.