Retiming Synchronous Circuitry with Imprecise Delays

I. Karkowski and R.H.J.M. Otten
Delft University of Technology
Faculty of Electrical Engineering
Mekelweg 4, 2628 CD Delft, The Netherlands

Abstract

Often, and certainly in the early stages of a design, the knowledge about delays is imprecise. Stochastic programming is not an adequate means to account for this imprecision. Not only is a probability distribution seldom a correct translation of the designer's delay knowledge, it also leads to inefficient algorithms. In this paper possibilistic programming is proposed for handling the retiming problem where delays are modelled as (triangular) possibilistic numbers. Beside the capability of optimizing the *most possible* clock cycle time and generating its possibility distribution, it allows for trade-offs between reducing clock cycle time and chances for obtaining worse solutions. It is shown that the computational complexity is the same as for retiming with exact circuit delays.

Introduction

Most synthesis methods use estimated values for the coefficients of the constraints and cost functions guiding the design. Especially in the early design phases these estimates may be far from the values ultimately realized. Effort devoted to obtain globally optimal solutions with respect to these cost functions is therefore of doubtful use. Yet wrong decisions made there may cause the necessity of a complete redesign, longer times to market, and thus reduced revenue.

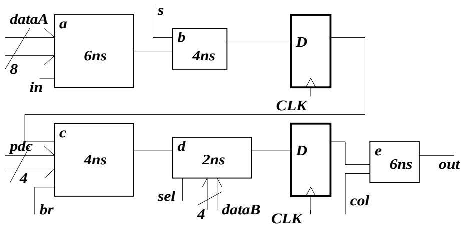


Figure 1: "Neuzel" before retiming.

Example 1 *We want to minimize the clock cycle ϕ of a circuit called "neuzel", given in figure 1. Initially $\phi = 10ns$, which can be calculated by finding the longest path between flip-flops or primary inputs/outputs. A faster circuit can be obtained by shifting both latches backward (see figure 2). Now the circuit should have $\phi = 8ns$. After applying logic synthesis tools to the combinatorial parts of the circuit, and placement and routing, ϕ turns out to be $14ns$. This is mainly due to the delay of the combinatorial block* c *that came out at 10ns instead of 4ns. The original configuration would then have $\phi = 12ns$.*

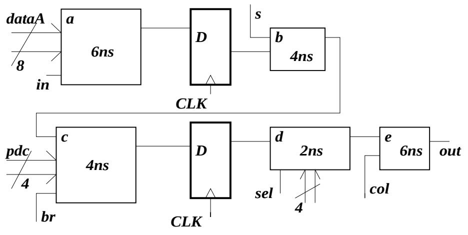


Figure 2: "Neuzel" after retiming.

It may seem reasonable to assume parameters to be random variables with an carefully derived probability distribution. Stochastic programming [Sen72] can then be applied. Unfortunately, stochastic programming methods are computationally inefficient, and generating good probability distributions of circuit parameters is difficult and in most cases not adequate (see example 3).

The approach we propose here is based on possibilistic programming [Zad78]. Methods belonging to this category have their roots in the theory of fuzzy sets and solve optimization problems on fuzzy numbers. These fuzzy numbers are easy to generate for most design parameters. Whenever mathematical programming formulations of problems are practical a straight-forward transformation into a possibilistic programming problem deserves consideration. Recently, such a straightforward application of the possibilistic approach to the high-level synthesis problem of *simultaneous scheduling, selection and allocation of functional units*, a problem that can be solved efficiently with general integer programming methods [GE90], was presented [Kar95]. Leiserson's approach to the retiming of very large networks [LS91] does not allow such a straightforward transformation, although its kernel, the feasibility check for cycle times, is often formulated as an integer program. In this paper we will show how to extend their algorithm to handle imprecise delays of combinatorial units.

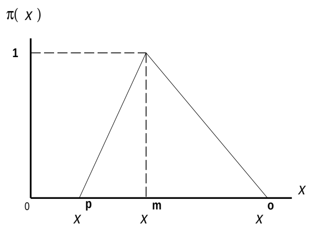


Figure 3: The triangular possibility distribution of fuzzy number x.

1 Possibilistic programming

1.1 Possibility distributions

To take uncertainty into account we use possibility distributions[1], as introduced by Zadeh [Zad78]. The interpretation of such functions is that a higher measure is assigned to more likely events. Among the various types of distributions, triangular and trapezoid are the most common ones in solving possibilistic mathematical problems. We have chosen for using fuzzy numbers with triangular possibility distributions $\pi(.)$ only (see e.g. figure 3). Such a number is denoted by $X = (x^m, x^p, x^o)$, where x^m, the *most possible* value has possibility measure 1 ($\pi(x^m) = 1$), and is between the bound values x^p and x^o ($x^p \le x^m \le x^o$). These bound values x^p and x^o can be interpreted as the most pessimistic and the most optimistic values. Which one is pessimistic and which optimistic depends on the context.

Example 2 *The delay of unit "c" in example 1 was apparently difficult to estimate. Therefore, instead of a poor approximation, we may represent the delay as the triangular fuzzy number* $\tilde{d}(c) = (4, 3, 10)$. *This means that the most possible delay of the unit "c" is 4ns, we do not expect it to be smaller than 3ns, nor larger than 10ns. In this context* $x^p = 3$ *is the most optimistic value,* $x^o = 10$ *the most pessimistic.*

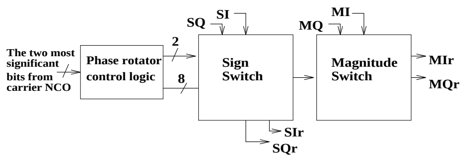
Values for x^m, x^o and x^p can be derived from technology parameters, experience with the synthesis methods used, and information about the structure of the circuit.

The following example shows that fuzzy numbers often can represent the meaning of a constraint better than probability distributions.

Example 3 *The timing specification for the phase rotator (figure 4) contains the following timing constraint:*

> *The delay between arrival of the signal on the "control" input and the time when the data becomes available at the "MIr" output should not be shorter than 5ns or longer than 15ns.*

[1]in fuzzy mathematical programming we use membership functions instead.



The two most significant bits from carrier NCO
Phase rotator control logic
2
8
SQ
SI
Sign Switch
MQ
MI
Magnitude Switch
MIr
MQr
SIr
SQr

Figure 4: The phase rotator.

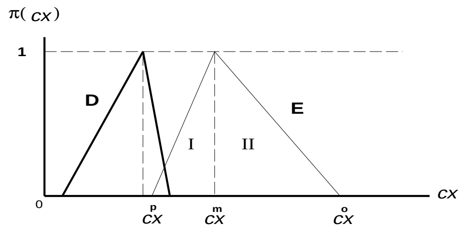


Figure 5: The strategy to solve " $\min \tilde{c}x$". We prefer the possibility distribution of D to that of E.

A straightforward application of the synthesis tools yields a delay of 10ns. One way to express this is as the following fuzzy constraint: $delay_{control,MIr} \le X = (10, 5, 15)$

On the other hand if we were using stochastic programming, the delay should be a random variable with a given probability distribution. It is clear that such a model would not represent the real meaning of our constraint.

Of course, this is not to imply that possibility theory can be a substitute in all applications for probability theory.

1.2 Linear programming with imprecise objective coefficients

We define the following Possibilistic Linear Program (**PLP**):

$$\min \tilde{c}x \qquad s.t. \quad \{Ax \le b, and \;\; x \ge 0\} \tag{1}$$

where $\tilde{c}$ may consist of imprecise numbers with possibilistic distributions. Replacing A and b with a fuzzy matrix $\tilde{A}$, and a fuzzy vector $\tilde{b}$ is tedious, but straightforward (see [LH92]). For given x, the value of the fuzzy objective function (eq. 1) is a fuzzy number defined by three corner points ($c^m x$,1),($c^p x$,0) and ($c^o x$,0). Thus, minimizing the fuzzy objective by pushing these three points to the left may not yield a valid possibility distribution. Therefore, instead of minimizing these three objectives independently, we rather simultaneously minimize $c^m x$, maximize $[c^m x - c^p x]$ and minimize $[c^o x - c^m x]$ (see figure 5). In this way we obtain the following Multiple Objective Linear Program (**MOLP**):

$$\begin{aligned} \max \quad & z_1 = (c^m - c^p)x \\ \min \quad & z_2 = c^m x \\ \min \quad & z_3 = (c^o - c^m)x \end{aligned} \tag{2}$$

$$s.t. \; x \in X = \{x : Ax \le b \;,\; x \ge 0\}$$

This (**MOLP**) would be equivalent to minimizing the *most possible* value of the imprecise cost (at the point of possibility degree = 1), if we ignored the first and third objective function. By including the other objectives we enable a trade-off between this goal and reducing the “risk of paying higher cost” (see region II in figure 5), and improving “the possibility of the lower cost” (region I).

To solve problem of eq. 2 any **MOLP** technique can be used (e.g. utility theory, goal programming, fuzzy programming or interactive programming).

2 Retiming synchronous circuitry with imprecise propagation delays

2.1 Preliminaries

Classical retiming applies to any edge-triggered, single-phase, synchronous circuit. This can be modelled as a finite, directed multigraph $G =< V, E, d, w >$. The vertices represent the connected components of the circuit after removing all latching elements. These components are acyclic combinatorial circuits, and their maximum input-output delay is added as a weight $d \in \mathbf{R}^+$ to each vertex. The weights w on the edges represent the number of latching elements on the connection between the corresponding combinatorial circuits. When the combinatorial delays are not exactly known, applying the standard retiming algorithm of [LS91] may in the end lead to unnecessary violation of timing constraints and cause expensive redesigns. Instead of a single nonnegative number we assign a fuzzy number to each vertex: $\tilde{d} \in X$, where X is the space of triangular fuzzy numbers. Figure 6 shows an example. The vertices still represent the combinatorial elements of the circuit, but each vertex $v \in V$ now has a triangular fuzzy number, $\tilde{d}(v)$, as its weight, modelling the nonnegative imprecise propagation delay of the corresponding element. The extra vertex v_h represents interfaces with the external world, and it is given zero propagation delay. The interpretation of the edges and their weights is the same as in the classical model. How to handle multiple fanout is described in [LS91].

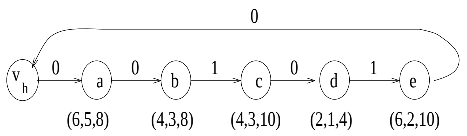


Figure 6: The graph model of the "neuzel" circuit.

The problem of finding a retiming r of the graph $\tilde{G}$ which optimizes its fuzzy/imprecise clock-period $\tilde{\phi}$ can be formulated as the general integer **MOLP** (analogous to the program of eq. 2):

$$\begin{aligned} \max z_1 &= \phi^m - \phi^p \qquad (3)\\ \min z_2 &= \phi^m \\ \min z_3 &= \phi^o - \phi^m \end{aligned}$$

$$s.t.\ \tilde{\phi} = (\phi^m, \phi^p, \phi^o) \in Y,$$

where Y is the set of feasible imprecise clock periods. The method described in section 1.2 can then be used to solve it. However, integer programming is in the general case $\mathcal{NP} - hard$. If it was possible to extend the standard approach while preserving its low computational complexity, an efficient retiming method with reduced risk for redesign would become available. We will show that this is possible. First however, some definitions are in order:

DEFINITION 1 *Let $f : X \mapsto \mathbf{R}^+$ be a linear mapping with positive coefficients, $\tilde{G}$=<V,E,$\tilde{d}$,w> a retiming graph with imprecise delays. Then the crisp graph of $\tilde{G}$ under f is a graph G_f=<V,E,d_f,w>, $d_f \in \mathbf{R}^+$ obtained from $\tilde{G}$ by mapping all fuzzy delays $\tilde{d}$ of $\tilde{G}$ into crisp values: $d_f = f(\tilde{d})$.*

We can of course apply classical retiming to G_f=<V,E,d_f,w>. In particular, if $f(\tilde{x}) = f(x^m, x^p, x^o) = x^m$ the result will be the optimal clock period when all combinatorial delays are equal to the *most possible* value assigned to their vertices. The *imprecise* delay of a path is defined through the usual definition of the addition operation on triangular fuzzy numbers:

DEFINITION 2 *The imprecise delay of a path p in graph $\tilde{G}$ is:*

$$\tilde{d}(p) = \sum_{v_i \in p} \tilde{d}(v_i) = (\sum_{v_i \in p} d^m(v_i), \sum_{v_i \in p} d^p(v_i), \sum_{v_i \in p} d^o(v_i)). \quad (4)$$

To enable optimization we need a *max*-operator over triangular fuzzy numbers:

DEFINITION 3 *The maximum imprecise total propagation delay on any critical path from u to v is:*

$$\tilde{D}(u, v) = \max_{p \in C} \tilde{d}(p) = (\max_{p \in C} d^m, \max_{p \in C} d^p, \max_{p \in C} d^o). \quad (5)$$

where C is the set of paths between u and v with the lowest number total edge weight.

DEFINITION 4 *The imprecise clock period $\tilde{\phi}$ of the circuit modeled by $\tilde{G}$ is a triangular fuzzy number defined by the equation:*

$$\tilde{\phi} = \tilde{\phi}(\tilde{G}) = \max_{p \in W^0} \tilde{d}(p) = (\max_{p \in W^0} d^m, \max_{p \in W^0} d^p, \max_{p \in W^0} d^o), \quad (6)$$

where W^0 is the set of all paths that $w(p) = 0$.

LEMMA 1 *Applying a linear mapping f (as in def. 1) to the imprecise clock period of $\tilde{G}$ yields the same number as determining the clock period of the crisp graph of $\tilde{G}$ under f:*

$$\phi_f = \phi(G_f) = f(\tilde{\phi}). \quad (7)$$

PROOF. Consider all paths p in G_f for which $w(p) = 0$. Because the structure and the edge weights of crisp graphs are identical to those of $\tilde{G}$, this set is the same as in $\tilde{G}$.

For any such path in G_f we have $d(p) = \sum_{v_i \in p} d(v_i) = \sum_{v_i \in p} f(\tilde{d}(v_i))$. Since f is linear and from definition 2 we get $\sum_{v_i \in p} f(\tilde{d}(v_i)) = f(\sum_{v_i \in p} \tilde{d}(v_i)) = f(\tilde{d}(p))$. Thus,

$$\phi_f = \max d(p) = \max f(\tilde{d}(p)) \qquad (\star).$$

For $\tilde{G}$ we have by definition 4 that $f(\tilde{\phi}) = f(\max \tilde{d}(p))$. Again, the properties of f imply that $f(\max \tilde{d}(p)) = \max f(\tilde{d}(p))$. Combining this with ($\star$) we obtain $\max f(\tilde{d}(p)) = \phi_f$. □

THEOREM 1 *Given a crisp graph G_f obtained from $\tilde{G}$ by use of the mapping f (as defined in def. 1) the following holds:*

$$\min_{\phi_f \in K} \phi_f = \min_{\tilde{\phi} \in Y} f(\tilde{\phi}). \qquad (8)$$

where K is the set of all feasible clock periods in G_f.

PROOF. From lemma 1 we know that applying the same legal retiming r to $\tilde{G}$ and G_f yields $f(\tilde{\phi}) = \phi_f$. From this follows that the retiming for which $\phi_f = \phi_{f\,min}$ is also a retiming of $\tilde{G}$ with $f(\tilde{\phi}) = f(\tilde{\phi})_{min}$. □

2.2 The FORTM algorithm

Theorem 1 is the basis of the following algorithm:

ALGORITHM FORTM (fuzzy optimization retiming):

1. Map the fuzzy graph $\tilde{G}$ into the crisp graph G_f using a mapping: $f(\tilde{x}) = a_1 x_p + a_2 x_m + a_3 x_o$ where a_1, a_2 and $a_3 \geq 0$.
2. Run the **LS** algorithm on the graph G_f.
3. Use the resulting configuration as optimal solution to the retiming problem of $\tilde{G}$.

THEOREM 2 *The algorithm* **FORTM** *solves exactly the program of equation 3.*

PROOF. We solve the program of equations 3 by mapping it into intermediate retiming problem, which we solve using an exact method. All our mappings are well defined in the sense of the definition 1. Therefore, from theorem 1 we get that the solution of the temporary problem represents exact solution to the original problem. □

2.3 Choice of the coefficients

We replace the three goal functions of the program 3 by a single one:

$$\min\left[-\beta_1(\phi^m - \phi^p) + \beta_2\phi^m + \beta_3(\phi^o - \phi^m)\right] \qquad (9)$$

where coefficients β_i represent the relative importance of the three subgoals. Without lost of generality we can assume that $\beta_1 = 1$. Then, we can rewrite the equation above as:

$$\min\left[\phi^p + (\beta_2 - \beta_3 - 1)\phi^m + \beta_3\phi^o\right] = \min\left[a_1\phi^p + a_2\phi^m + a_3\phi^o\right] \qquad (10)$$

To justify the application of the **FORTM** algorithm, we have to make sure that all coefficients a_i are positive. Consider therefore the relative importance of the three subgoals of program 3. It is easy to see that in practice every designer will assign β_i values in such a way that $\beta_1 \leq \beta_3 \leq \beta_2$. Otherwise we could obtain solutions with, for example, small risk of obtaining a longer clock period but relatively long most possible clock period. Because of that and since $\beta_1 = 1$ the coefficients a_1, a_3 will be positive. Consider now two "symmetric" fuzzy

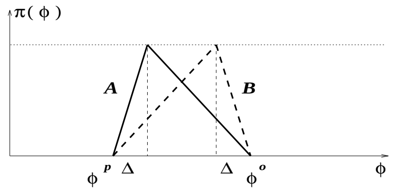


Figure 7: Two fuzzy numbers

numbers as in figure 7. They have the same pessimistic and optimistic values, but **A**'s most possible value is smaller than **B**'s most possible value. If we are to prefer **A** over **B** in such a case we should have

$$\phi^p + (\beta_2 - \beta_3 - 1)(\phi^p + \Delta) + \beta_3\phi^o \leq \phi^p + (\beta_2 - \beta_3 - 1)(\phi^o - \Delta) + \beta_3\phi^o. \qquad (11)$$

This implies that

$$(\beta_2 - \beta_3 - 1) \geq 0, \qquad (12)$$

which is equivalent to the requirement that $a_2 \geq 0$. In this way we have shown that for every practical choice of β_i all coefficients a_i will be positive and therefore the function of eq.10 will be a mapping function in the sense of definition 1, and consequently theorem 2 applies.

2.4 Computational complexity

The complexity of step 1 is $\mathcal{O}(|V|)$. In step 2 we call the standard retiming algorithm **LS**[2] of complexity $\mathcal{O}(|V||E| \lg |V|)$. This complexity remains unchanged because the graph mapping operations do not change the structure of the graph. Also the values of d_f remain well defined. Thus the total complexity is $\mathcal{O}(|V||E| \lg |V|)$.

3 A numerical example

In this section we illustrate with a simple numerical example how to use the obtained results to retime circuits with imprecise delays.

Consider the circuit "neuzel" from figure 1. Its graph model is presented in figure 6. The delay of every combinatorial block of the circuit is a triangular fuzzy number (also shown in figure 6). We use the following coefficients: $\beta_1 = 1$,

[2]or $\mathcal{O}(|V|^3 \lg |V|)$ if we use Bellman-Ford instead of FEAS

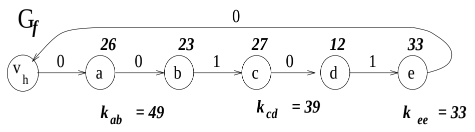


Figure 8: G_f after retiming if $d(c) = (4, 3, 10)$.

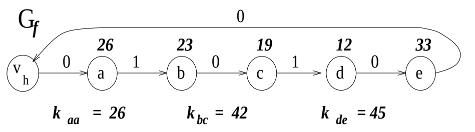


Figure 9: G_f after retiming if $d(c) = (4, 3, 6)$.

$\beta_2 = 3.5, \beta_3 = 2$. The mapping function for the graph G_f thus becomes:

$$f(\tilde{x}) = f(x^m, x^p, x^o) = x^p + 0.5x^m + 2x^o.$$

The minimum clock-period retiming of the graph G_f is shown in figure 8. This is the same configuration as in the initial circuit. As you recall from our discussion of example 1 this is exactly the solution that we then had preferred. The reader can easily check that if we had $d(c) = (4, 3, 6)$ (the risk that the delay of the unit "c" is larger would be smaller) then we would get $d_f(c) = 19$ in G_f (see figure 9). The same retiming results as produced by the **LS** algorithm with exact delays equal to x^m.

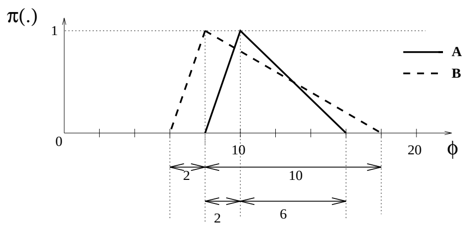


Figure 10: Two solutions to the retiming problem of the "neuzel" circuit.

In figure 10 two triangular fuzzy numbers are depicted. The fuzzy number **A** drawn with full lines is the imprecise clock-period of the configuration that we obtained using the **FORTM** algorithm. The fuzzy number **B** represents the possibility distribution of the clock cycle time in the configuration generated by the **LS** algorithm. Comparing these two solutions we can see that the most possible clock-period in **A** is larger than in **B**, and that the left spreads are the same. **B** does have a much larger right spread however, which "measures" the risk of obtaining a worse solution. We conclude that solution **A** is "better" in the sense of the strategy defined by program 3. There may still be design situations in which the user wants faster circuits and he is prepared to take more risk. In such case he can influence the tradeoff between the three subgoals of eq.3 by appropriately setting the weight coefficients β_i.

It is interesting to point out a very important characteristic of this method. There may exist many solutions with similar *most possible* values of the cost function (very close to the global optimum). The algorithm will be able to choose among these one with a small risk of ending up with a slower implementation and a maximal *possibility* of obtaining an even faster one.

4 Conclusions

In this paper we presented an extension to the "classical" algorithm for retiming synchronous circuits to cope more adequately with the uncertainty in combinatorial delays. We described an algorithm that handles retiming of single-phase, edge-triggered, synchronous circuits with imprecise delays, given as triangular fuzzy numbers. The computational complexity of this algorithm is the same as for single-valued delays, i.e. $\mathcal{O}(|V||E| \lg |V|)$. We believe that similar techniques can be successfully applied to other synthesis tasks. In fact, whenever mathematical programming formulations apply for obtaining efficient optimizations and one has to deal with imprecise data, possibilistic programming deserves consideration. As examples we mention here retiming of circuits with level-sensitive latches [LE92][IL92] or transistor sizing.

References

[GE90] C.H. Gebotys and M.I. Elmasry. A global optimization approach to architectural synthesis. In *IEEE International Conference on Computer Aided Design*, pages 258–261, 1990.

[IL92] A.T. Ishii and C. Leiserson. Optimizing two-phase, level-clocked circuitry. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference*. MIT Press, 1992.

[Kar95] I. Karkowski. Architectural synthesis with possibilistic programming. In *Proceedings of the HICSS-28*, January 1995.

[LE92] B. Lockyear and C. Ebeling. Optimal retiming of multi-phase, level-clocked circuits. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference*. MIT Press, 1992.

[LH92] Y.J. Lai and C.L. Hwang. A new aproach to some possibilistic linear programming problem. *Fuzzy Sets and Systems*, 49:121–133, 1992.

[LS91] C. Leiserson and J. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–25, 1991.

[Sen72] J.K. Sengupta. *Stochastic Programming*. North Holland Publishing Company, 1972.

[Zad78] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.