

# LNS Architectures for Embedded Model Predictive Control Processors

Jesus Garcia  
CSE Department  
Lehigh University  
Bethlehem, PA 18015, USA  
jegj@Lehigh.EDU

Mark G. Arnold  
CSE Department  
Lehigh University  
Bethlehem, PA 18015, USA  
maab@Lehigh.EDU

Leonidas Bleris  
EE Department  
Lehigh University  
Bethlehem, PA 18015, USA  
leb3@Lehigh.EDU

Mayuresh V. Kothare  
ChemE Department  
Lehigh University  
Bethlehem, PA 18015, USA  
mayuresh.kothare@Lehigh.EDU

## ABSTRACT

This paper presents a research on arithmetic units targeted to implement model predictive control (MPC) in a custom embedded processor. A novel hardware implementation of cotransformation for the calculation of addition and subtraction in the Logarithmic Number System (LNS) is proposed. This architecture provides a small ROM-less adder/subtractor, with longer operation latency than other LNS techniques, but easily pipelineable. These characteristics make it very adequate for implementing the datapath of custom MPC embeddable microprocessors. A review of the arithmetic customization process is presented, including: an analysis of the finite precision problem, modifications to the standard MPC algorithm that simplify embedding the application, and the reasons that suggest better performance of LNS over standard floating-point (FP) architectures. The proposed arithmetic unit architecture for 16-bit LNS is fully synthesized for ASIC, and compared with an equivalent FP implementation. Area and clock cycle estimates are compared. Finally, considerations on low-precision implementations of LNS arithmetic units are provided, and an embedded-ROM implementation of addition/subtraction in LNS is proposed and analyzed.

## Categories and Subject Descriptors

B.2.0 [Hardware]: Arithmetic and Logic Structures—*General*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CASES'04, September 22–25, 2004, Washington, DC, USA.  
Copyright 2004 ACM 1-58113-890-3/04/0009 ...\$5.00.

## General Terms

Design, Measurement

## Keywords

FWL, LNS, MPC, cotransformation, reduced precision

## 1. INTRODUCTION

The term “model predictive control” [4] includes a wide range of control techniques that explicitly use the model of a process to obtain the control sequence, by minimizing an objective function. MPC originated in the process industries in the late seventies. The success of MPC is due to: (a) MPC offers an optimal control sequence, and (b) it can be applied to both linear and non-linear multivariable systems with input, output and state constraints. The drawback of MPC is that every control sample time, an optimization problem has to be solved. For a large number of variables, optimization becomes exponentially expensive in terms of computing cost, rendering on-line optimization unpractical. Thus MPC has not been introduced yet in control systems with fast sampling times. But MPC is very desirable for applications with relatively fast control cycles, such as Systems-on-a-Chip (SoC) for medicine and bioengineering (drug delivery, diagnostics), avionics and aerospace (nano and microscale actuators and sensors, smart reconfigurable geometry wings and blades, microgyroscopes), automotive systems and transportation (accelerometers), or microreactors for in-situ and on-demand chemical production. Such applications require a small and power-efficient microcontroller, but capable of solving relatively complex optimization problems at a fixed and possibly high sampling rate. The apparition of embedded MPC technologies is expected to be revolutionary in the target fields described.

Recently, a technique has appeared [2] where the solution of the constrained optimization is shown to be piecewise affine in regions determined by the constraints. The affine laws can be precalculated and the online operations are greatly simplified. However, the growth in the number of regions required is superexponential, and memory re-

quirements for complex problems are extremely high. In order to solve comparatively complex problems, it has been proposed instead customizing the hardware and software to each particular MPC problem [3]. This is very appropriate for a SoC, where the proposed methodology can be used to obtain the minimum precision (and thus wordsize) that guarantees the required control performance for that problem. Reduced precision in the datapath, combined with a Hybrid MPC control algorithm, yields great improvements in computation speed, while decreasing power consumption, circuit area and required data memory. Based on the circuit implementation presented here, a VHDL description of the required arithmetic units can be automatically generated and integrated with the rest of the SoC. Therefore, a complete framework is available that automatizes obtaining a tailored datapath for an MPC problem, and provides the means to estimate the performance and sampling rate of the resulting MPC system.

The rest of this paper is organized as follows. Section 2 briefly introduces MPC and LNS, and the requisites for embedding MPC in the target applications. Section 3 reviews the steps in the customization design of the microprocessor's datapath, the modifications proposed to the standard MPC algorithm to improve the accuracy under low-precision arithmetic, and how this leads to LNS arithmetic. A new implementation of cotransformation, using a multipartite table to store the required values, is proposed in Section 4. The small tables required for limited precision LNS can be fully implemented in logic, and the resulting circuit is fully synthesizable for ASIC technologies, as opposed to most LNS-related works, relying on FPGA-based synthesis. This allows a comparison with FP arithmetic units more adequate for embedded systems. Estimations of the achievable clock cycle are derived both for LNS and FP. In Section 5, estimations of area and delay from state-of-the-art embeddable ROM are used to compare clock cycle and circuit size for straight-table lookup LNS implementation. This is an alternative often ignored in the literature. Finally, Section 6 summarizes the contributions of this paper and extracted conclusions, and proposes future research directions.

## 2. CONCEPTS OF MPC AND LNS

### 2.1 Model Predictive Control

Generally, controllers belonging to the MPC family are characterized by the following steps. Initially the future outputs of the controlled plant are calculated at each sample interval over a predetermined horizon  $P$ , the *prediction horizon*, using the process model. These outputs  $y(t+k|t)$  (for  $k=1\dots P$ ) depend up to the time  $t$  on the past inputs and on the future signals  $u(t+k|t)$  (for  $k=0\dots P-1$ ), which are those to be sent to the system. The nomenclature  $a(b|c)$  refers to a future (and therefore, predicted) value of the variable  $a$  at future time  $b$ , predicted by the control algorithm at time  $c$ . The next step is to calculate the set of future control moves by optimizing some criterion in order to keep the process as close as possible to a predefined reference trajectory. This criterion is usually a quadratic function of the difference between the predicted output signal and the reference trajectory. In some cases the control moves  $u(t+k|t)$  are included in the objective function in order to minimize the control effort. Finally, the first control move  $u(t|t)$  is sent to the system while the rest are rejected. At the next sam-

pling instant the output of the system  $y(t+1)$  is measured and used to correct the state estimation in the controller, creating a feedback loop. The procedure is repeated with the new values to get an updated control sequence with all horizons reaching one timestep further.

For a state space model, formally we have:

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (1)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \quad (2)$$

where  $\mathbf{x}$  is the state and  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{C}$  are the matrices storing the system's model. The prediction model is given by

$$\hat{\mathbf{y}}(t+k|t) = \mathbf{C}\hat{\mathbf{x}}(t+k|t) \quad (3)$$

$$\hat{\mathbf{y}}(t+k|t) = \mathbf{C}[\mathbf{A}^k\mathbf{x}(t) + \sum_{i=1}^k \mathbf{A}^{i-1}\mathbf{B}\mathbf{u}(t+k-i|t)]. \quad (4)$$

Quadratic cost functions are typical, having the form:

$$J_P(k) = \sum_{k=0}^P \{[\mathbf{y}(t+k|t) - \mathbf{y}_{\text{ref}}]^2 + \mathbf{R}\mathbf{u}(t+k|t)^2\} \quad (5)$$

$$|\mathbf{u}(t+k|t)| \leq \mathbf{b} \quad , \quad \mathbf{k} \geq \mathbf{0} \quad (6)$$

where  $\mathbf{y}(t+k|t)$  are the predicted outputs,  $\mathbf{y}_{\text{ref}}$  is the desired set reference output,  $\mathbf{u}(t+k|t)$  is the control sequence and  $R$  is the weighting on the control moves, a design parameter. Usually only the first  $C$  control moves are optimized ( $C$  is the *control horizon*), and the following  $(P-C)$  moves are assumed to be zero. This system is subject to input constraints given by the vector  $\mathbf{b}$ .

The control loop has to be executed every control cycle, and the optimization stage generates most of the operations that will be executed. Optimizing  $\mathbf{u}(t+k|t)$  to minimize equation (5) can be accomplished using several algorithms, but all of them are computationally expensive (in general,  $O(n^3)$  operations, where  $n$  is the number of variables to optimize: the dimension of  $\mathbf{u}$  times  $C$ ). The described hard real-time problem is commonly solved using workstations with double-precision FP units. For an embedded processor, double precision implies a huge cost in terms of circuit area and power consumption at the high clock cycles required (hundreds of MHz). Reducing the precision of the number representation to the minimum acceptable for the particular problem has been proposed in [3] as the main step to reduce these costs and allowing higher clock frequencies.

### 2.2 Logarithmic Number System

The Logarithmic Number System (LNS) represents the value of the real number  $X$  using a sign bit  $S_X$  and the base- $b$  logarithm,  $x = \log_b |X|$ , where  $x$  is a fixed-point word, consisting of a sign bit  $S_x$ ,  $K$  integer bits and  $F$  fraction bits [11]. The value of  $K$  controls the *dynamic range* of the representation (largest, in absolute value, and closest to zero values), while  $F$  controls the *precision* (quantization of the representable values between two consecutive powers of the base  $b$ ). The represented number  $X$  can be expressed as

$$X = (-1)^{S_X} b^x. \quad (7)$$

With this representation, multiplication, division and square root operations consist in fixed point addition, subtraction and right-shift respectively, due to the properties of the logarithms. These operations can be implemented with simpler,

faster circuits, and consume less power than an equivalent FP circuit. On the other hand, addition and subtraction are more complicated in LNS, and commonly implemented using the transcendental functions  $s_b(z)$  and  $d_b(z)$ , defined as:

$$\log_b(1 + Z) = \log_b(1 + b^z) = s_b(z), \quad z < 0 \quad (8)$$

$$\log_b(1 - Z) = \log_b(1 - b^z) = d_b(z), \quad z < 0. \quad (9)$$

Let  $X \neq 0$  and  $|X| \geq |Y|$ . Depending on the signs  $S_X$  and  $S_Y$ , and substituting with equations (8) and (9), addition of  $X$  and  $Y$  can be computed using:

$$\begin{aligned} S_X = S_Y : \quad & |R_1| = |X| + |Y| \rightarrow \\ & r_1 = x + s_b(z), \quad S_{R_1} = S_X \end{aligned} \quad (10)$$

$$\text{where: } \log_b |R_1| = \log_b |X(1 + \frac{Y}{X})|$$

$$\begin{aligned} S_X \neq S_Y : \quad & |R_2| = |X| - |Y| \rightarrow \\ & r_2 = x + d_b(z), \quad S_{R_2} = S_X \end{aligned} \quad (11)$$

$$\text{where: } \log_b |R_2| = \log_b |X(1 - \frac{Y}{X})|$$

Subtraction of  $X$  and  $Y$  is computed using equation (10) for  $S_X \neq S_Y$  and equation (11) for  $S_X = S_Y$ . The functions  $s_b(z)$  and  $d_b(z)$  are usually implemented using look-up tables, combined with various methods like interpolation or cotransformation to reduce the required data storage. The cost of the table increases quadratically for linear increments of precision, making LNS less efficient compared with FP for wordsizes above 32 bits. But for decreasing sizes, LNS becomes more competitive, reducing the cost of addition of subtraction (the disfavorable operations) quadratically as well.

### 2.3 Design choices

The optimization process represents the main computing load of the MPC algorithm. Many algorithms exist to minimize a quadratic cost function, but they can be classified in Interior Point and Quadratic Programming (QP) methods. The latter have been chosen due to a generally better performance for the problem sizes considered as target applications (in the order of 100 variables to optimize). QP algorithms require matrix inversion and matrix-matrix multiplication in every case. Addition and multiplication are balanced, not favoring LNS nor FP (roughly 50% of the computing load corresponds to each operation). But as suggested in [7], using Householder's transformations to invert a rectangular matrix is a very adequate method to treat the optimization constraints. This algorithm requires abundant square roots. This operation has a extremely simple implementation in LNS (simple right-shift). Considering also the quadratic reduction in LNS required tables as precision decreases, this alternative to floating point is potentially more efficient. This motivates the development of appropriate custom LNS architectures and subsequent comparison with FP. Given the simplicity of the LNS implementation for multiplication, division and square root (Section 2.2), the adder/subtractor represents almost the whole design effort.

## 3. CUSTOMIZATION METHODOLOGY

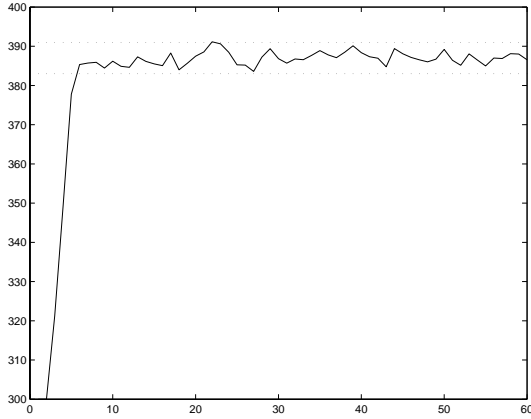
As mentioned in Section 1, the proposed approach to embedded MPC is reducing the precision and customizing the datapath, to obtain throughputs in the order of hundredths of

MIPS (million instructions per second). For wordsizes sufficiently smaller than double precision, the power and area savings, combined with high achievable clock cycles, will reasonably embedding the controller with reasonable costs. Section 4 addresses if the precision reduction achieved is sufficient with a positive answer.

Reducing the precision required to obtain correct results from an algorithm is always desirable for a custom system, and works such as [9] have proposed it for DSP algorithms (of much less complexity than MPC). Works in the control field [12, 10] have applied error analysis to state-space control, trying to find the minimum word length that guarantees the system's stability. The resulting problem is found to be open (not solved yet), and thus either computationally tractable upper-bounds (largely overestimative), or non-rigorous assumptions, are used to simplify the original problem. Also, only the initial representation of the problem data in finite precision is considered, implicitly assuming infinite precision for all subsequent operations in the control algorithm. Actually, every particular operation contributes with a potential roundoff error to increase the inaccuracy of the results. To avoid the limited-precision problem, MPC is usually implemented in double-precision FP, and problems that still present numeric instability are designated as *ill-conditioned*.

The proposed methodology uses bit-accurate hardware emulation in software to obtain the exact results for a given algorithm using a given hardware implementation. To obtain bit-accurate results, the particular hardware arithmetic unit is first implemented in VHDL. The required  $s_b(z)$  and  $d_b(z)$  tables are obtained from the hardware description, by running an appropriate simulation. The software library uses these real data obtained from the hardware description to emulate hardware results. A rigorous relationship between the control problem's parameters and the minimum required precision is very desirable, but has been shown to be an unsolved problem until now. To overcome this, in the proposed methodology statistical measures are obtained with hardware emulation for a large set of tests. These can guarantee realistic worst-case performance with a confidence that grows with the number of tests. The characteristics of this methodology make the results particular for the problem under study. However, the nature of the application (processors for embedded control) justifies this approach: the controller can be defined as a hardware description language (HDL) core with some parameters, such as the word length, and these can be found for each particular SoC to be developed.

The statistical tests allow characterizing the response of the plant using reduced-precision MPC. The errors (considered as the difference with an ideal control decision, computed in double precision) are bounded. Analyzing a step response, the behavior of the plant is qualitatively ideal during the initial stage (while approaching the optimal state). But when a double-precision implementation would completely stabilize the plant, reduced-precision MPC introduces a random noise due to precision errors. Therefore, instead of bringing the system to the optimum state, it stays in an bounded neighbourhood of the optimum state. A typical reduced-precision response to a step reference is shown in Figure 1. These conclusions motivate proposing a Hybrid MPC algorithm, where a simpler state-space control is combined with MPC. Using the results from the statistical tests,

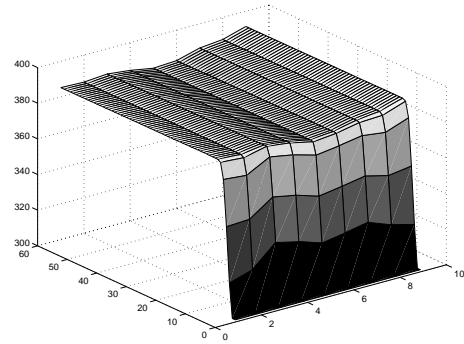


**Figure 1: Typical plant response, using low-precision MPC.**

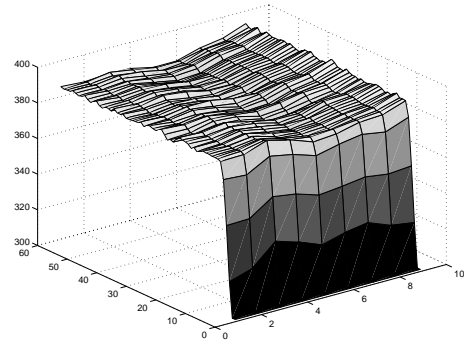
the bounds of the final random error for each controlled variable are known. When those variables are detected to stay within the bounds, it can be assumed that the neighbourhood of the optimum state has been reached. The controller switches to the simpler state-space algorithm, e.g. linear regulator, which requires much smaller precision to stabilize the system (due to the much smaller number of operations required). Since the system is already very close to the optimum, a simpler control method is capable of stabilizing the state in a very short time. Running the hardware emulation on the hybrid MPC algorithm shows a performance very close to that of double-precision MPC, with no qualitative difference and only slightly suboptimal control sequences. Additionally, when using the alternate control, the processor remains idle most of the control cycle. This yields potentially very large power savings (depending on the dynamics of the particular system being controlled). Figures 2 and 3 show a comparison of the responses obtained using double-precision floating point (64 bits), and reduced-precision LNS (16 bits), for a sample problem. This problem consists in a distributed temperature control, using nine resistive heaters and nine sensors to bring the central section of a disc to a desired temperature profile. Notice that, even without Hybrid MPC, the reduced-precision algorithm produces a response very similar to double precision FP. Hybrid MPC practically makes the differences almost unnoticeable.

#### 4. REDUCED-PRECISION LNS: COTRANSFORMATION AND MULTIPARTITE TABLE

The proposed LNS ALU has to take into account the vectorial nature of the algorithm, where the most common operations are multiply-and-accumulate (MAC) for dot products, and vector-vector addition/subtraction. The inherent parallelism of these operations makes aggressive pipelining reasonable. Data dependencies are rarely present, and a pipeline will be fully occupied the majority of the execution time. This makes reasonable trading an increase in addition/subtraction delay, for reduced area. Another reason for searching this tradeoff is the desire of comparing ASIC-based implementations for LNS and FP. This is much more



**Figure 2: Plant response using 64-bit double-precision FP.**



**Figure 3: Plant response using 16-bit reduced-precision LNS.**

appropriate than FPGA-based implementations for a circuit to be embedded in a SoC. The difficulty of synthesizing ROM tables causes a lack of such studies in the literature. But for small enough tables, these can be directly implemented in combinational logic. While this is not the most efficient approach, it serves the purpose of comparing ASIC technologies.

The parametric library provided in [5] generates small multipartite tables for  $s_b$  and  $d_b$ . As pointed out in LNS literature,  $d_b(z)$  is more difficult to interpolate due to the singularity at  $z = 0$ , and usually requires large tables. A smaller table is obtained relaxing the precision of the subtraction operation. In this way, an error equal as the worst case error in FP subtraction (when catastrophic cancellation occurs) is tolerated. But this has shown, using hardware emulation, negative effects on the accuracy of reduced-precision MPC (where unaccurate subtraction is problematic).

All the reasons presented above suggest cotransformation to trade extra delay for adequate accuracy and small tables. Cotransformation avoids computing  $d_b(z)$  for  $z$  “close” to zero. The cotransformation proposed in [1] allows computing the transcendental  $d_b(z)$  function from  $s_b(z)$  and two small tables,  $F_1(z_h)$  and  $F_2(z_l)$ , using the relationship:

$$d_b(z) = z + d_b(-z_h - \delta_h) + s_b(d_b(z_l - \delta_h) - (z + d_b(-z_h - \delta_h))), \quad (12)$$

where  $z_h + z_l = z$  is obtained splitting the bits of  $z$ , and  $F_1(z_h) = d_b(-z_h - \delta_h)$ ,  $F_2(z_l) = d_b(z_l - \delta_h)$ . The para-

metric libraries in [5] have been modified for ASIC implementation, and the LNS adder/subtractor substituted by the circuit shown in Figure 4.  $F_1$  and  $F_2$  are implemented with combinational logic.  $s_b$  is computed using the multipartite table from [5], which in the case of addition is accurate enough. Each of the subtables is also synthesized to gates. The initial stage computes in parallel  $(x - y)$  and  $(y - x)$  trading an extra fast adder for reduced delay.

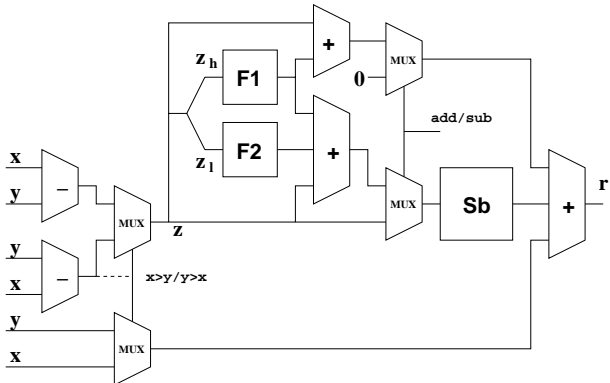


Figure 4: LNS adder/subtractor with multipartite table-based cotransformation .

Precisions around 9 or 10 fractional bits in LNS (mantissa bits in FP) are typical for the target problems. The dynamic range required is comparatively large, typically at least 5 integer bits in LNS (exponent bits in FP). A 16-bit configuration is chosen for the comparison. The parameters for LNS are  $K = 6$ ,  $F = 9$  and one sign bit. The 16-bit FP representation for equivalent precision and dynamic range has 6 exponent bits and 9 mantissa bits. Two ALUs comprising addition, subtraction, multiplication, division and square root (the operations required in MPC) were synthesized for FP and LNS respectively, using TSMC 0.25  $\mu\text{m}$  technology, and the results are presented in tables 1 and 2.

LNS	Area (gates)	Delay (ns)
ADD/SUB	42500	17.92
MUL/DIV	3100	2.51
SQRT	81	0.09
Total	45681	-

Table 1: Area and delay for 16-bit LNS ALU.

FP	Area (gates)	Delay (ns)
ADD/SUB	21071	10.44
MUL	17512	7.81
DIV	27925	30.43
SQRT	13067	19.38
Total	79575	-

Table 2: Area and delay for 16-bit FP ALU.

The first observation that can be made is that the total area of the LNS ALU is just about 60% that of the FP ALU. Attending delay, and even though all operations appear in the MPC algorithm, multiplication and addition (or subtraction) dominate the computing load, appearing both

with roughly the same frequency. Therefore delays in the division and square root operations in FP represent an overhead but not a critical disadvantage. On the other hand, FP has more equilibrated delays for addition and subtraction (same unit) and multiplication. Pipelining is possible in both operations. However, clock cycles under 5 ns may cause the pipeline for FP division to be too deep. In the case of LNS, multiplication is just a fixed-point addition, thus very fast. The adder/subtractor can be pipelined, and 5 ns is a reasonable target clock cycle, giving 4 pipe stages. As noted in [6], multipartite tables (the  $s_b$  block in Figure 4) are very amenable to pipelining. The conclusion extracted is that while achievable clock cycles are similar for pipelined LNS and FP (around 5 ns), LNS requires much less pipelining (just the adder/subtractor unit), and will require only 60% of the area of the FP ALU to be implemented, even implementing the table lookups with combinational logic. To provide further insight in the LNS design, the multipartite table for  $s_b$  requires 12449 equivalent gates,  $F_1$  8207, and  $F_2$  4920 gates. The  $F_1$  function is comparatively the most disadvantageous to implement in combinational logic.

## 5. ESTIMATIONS FOR LNS BASED ON EMBEDDED ROM

Implementing lookup-tables in logic is an unefficient technique, only used to provide a measure of the cost of such tables for ASIC (as opposed to FPGA implementations, where lookup tables are part of the available primitives). However, in a state-of-the-art embedded implementation, silicon intellectual property (IP) generators can be purchased, that generate custom ROM tables in the same technology as the rest of the circuit. The online estimation tools in [8] were used to generate the specifications for custom TSMC 0.25  $\mu\text{m}$  ROMs. As pointed out in Section 2.2, straight table-based lookup of  $s_b$  and  $d_b$  has a cost that decreases quadratically with the precision. For the considered parameters ( $K = 6$  and  $F = 9$ ),  $s_b$  requires less than 8 Kwords of 9 bits, and  $d_b$  under 8 Kwords of 14 bits. The results obtained in [8] for the two memories are summarized in table 3.

ROM	Size	Area ( $\text{mm}^2$ )	Delay (ns)
$s_b$	8K x 9	0.331	4.67
$d_b$	8K x 14	0.517	4.74

Table 3: Area and delay for 16-bit FP ALU.

While judging the merit of the area figures requires owning the proprietary technology to generate the circuit layout (to compare physical areas), the delays are very improved over the cotransformation technique. A circuit using these ROMs for straight table lookup of  $s_b$  and  $d_b$ , like the one depicted in Figure 5, computes addition or subtraction in under 8 ns (even faster than FP).

## 6. CONCLUSIONS AND FUTURE WORK

This work presents a research on the characteristics required by arithmetic units to allow embedding MPC using a small, low-power custom microprocessor. With the use of hardware emulation techniques, and a Hybrid MPC algorithm developed for reduced-precision performance, the traditional double-precision FP implementation can be reduced to just 16 bits. This four-fold wordsize reduction yields

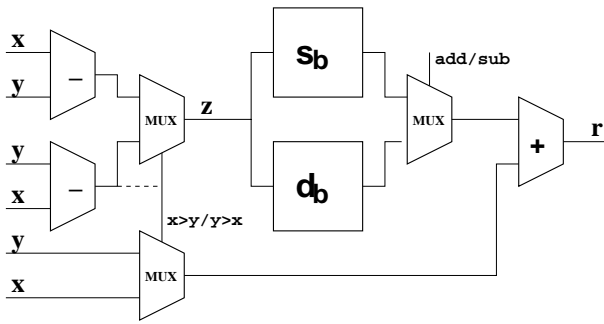


Figure 5: LNS adder/subtractor with straightforward table lookup.

speed improvements, and area and power savings. The latter are further increased by the Hybrid algorithm, which causes power savings by keeping the processor idle most of the control cycle when the controlled system is close to the optimum (desired) state. A ROM-less (fully combinational) 16-bit LNS ALU has been implemented using a novel LNS cotransformation architecture. The synthesis results of the previous circuit are compared with an equivalent 16-bit FP, showing that the LNS ALU requires only 60% of the area of the FP ALU, while pipelining can be applied to both to obtain clock cycles around 5 ns. This will yield close to 200 MIPS in the resulting processor, thanks to the vectorial nature of the operations required by MPC. This throughput is enough to solve the target problems considered. The proposed ALU, combined with the proposed algorithms, enables a viable implementation for embedded MPC, an application expected to be revolutionary for the SoC (especially MEMs) field, and to the best knowledge of the authors, non-existent in practical form yet. Finally, a point is made towards using techniques which require larger tables for increased speed, when reduced-precision LNS is used. The efficiency of embedded ROM implementations is likely to further improve the advantage of LNS over FP at these precisions.

Future research includes a more detailed comparison of LNS circuits using embedded ROM, a comparative study of area and size for different ALU implementations and different precisions (including pipelining), and the integration of the ALU with a microprocessor core, to constitute a parameterized vectorial processor, customizable for each particular MPC problem.

## 7. REFERENCES

- [1] M. G. Arnold. Improved Cotransformation For LNS Subtraction. *IEEE International Symposium on Circuits and Systems*, II:752–755, May 2002.
- [2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, (38):3–20, 2001.
- [3] L. Bleris, M. Kothare, J. Garcia, and M. Arnold. Embedded Model Predictive Control for System-on-a-Chip Applications. In *DYCOPS*, July 2004.
- [4] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, New York, 1999.
- [5] J. Detrey and F. de Dinechin. <http://perso.ens-lyon.fr/jeremie.detrey/FPLibrary/>, 2004.
- [6] F. de Dinechin and A. Tisserand. Some Improvements on Multipartite Table Methods. In *15th IEEE Symposium on Computer Arithmetic*, pages 128–135, June 2001.
- [7] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.
- [8] Dolphin Integration. [http://www.dolphin.fr/flip/ragtime/025/dlm\\_ragtime\\_025.html](http://www.dolphin.fr/flip/ragtime/025/dlm_ragtime_025.html), 2004.
- [9] K.-I. Kum and W. Sung. Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems. *IEEE Transactions on Computer-Aided Design of Integrated Curcuits and Systems*, 20(8):921–930, 2001.
- [10] G. Li, R. Istepanian, W. Mei, and J. F. Whildborne. Finite Precision Digital System Structures with Stability and Roundoff Noise Consideration. In *Proceedings of the American Control Conference*, pages 4333–4337, June 1999.
- [11] E. E. Swartzlander and A. G. Alexopoulos. The Sign/Logarithm Number System. *IEEE Transactions on Computers*, 24(12):1238–1242, Dec. 1975.
- [12] J. Wu, S. Chen, J. Whidborne, and J. Chu. Optimal Floatin-Point Realizations of Finite-Precision Digital Controllers. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 2570–2575, December 2002.