An ASIC for Die-Sinking Spark Erosion Simulations

Mark Genoe **Wim Ploegaerts** **Luc Claesen** **Hugo J. De Man**
Interuniversity Micro Electronics Center (IMEC), Leuven

Claude Tricarico **R. Delpretti** **Dirk Dauw**
Charmilles Technologies S.A., Geneva

Abstract

Electrical Discharge Machining (EDM) simulation is always been considered as a very hard problem, for reasons that such simulations are extremely computation intensive. However, this paper presents a wide range of implementations for the Die-Sinking Spark Erosion Process, which prove that EDM simulations are today executable in an acceptable time. All the implementations we propose here are designed on a full automatic way, using the Cathedral Silicon Compilers System developed at IMEC-Heverlee. The results are compared with similar implementations on a TMS320 domain specific commercial signal processor. From the viewpoint of EDM-users, these simulations are very useful because the final results of this kind of thermo-electrical processes are in no way predictable.

I. Introduction

In Electrical Discharge Machining (EDM) [1], material removal is achieved by the thermal action of electric discharges occurring between a tool-electrode and the workpiece, in contrast with the conventional metal-removing methods using high mechanical forces. By each discharge a small volume of material is melt away from workpiece as well as from the tool-electrode. The electric discharges are produced by a generator applying voltage impulses between electrode and workpiece, at a rate of 10 to 1000 Khz.

The gap between electrode and workpiece tends to increase as material removal goes on. If the tool-electrode was fixed in its original position the machining process would stop as soon as the gap would reach a size too large to allow for further discharge breakdown. In order to maintain the 'sparking'-process, the electrode has to be fed towards the workpiece in such a way as to keep the gap-width within some narrow limits and to avoid contact between the electrode and the workpiece. This is achieved by a servo-controlled feed mechanism. By this way the tool-electrode is sunken gradually into the workpiece. The complementary shape of the electrode is than reproduced into the workpiece with a small oversize of 0.01 to 0.02 mm (figure 1).

The EDM operation is carried out in a dielectric fluid intended primarily to concentrate the discharge energy of a single discharge. Forced circulation of the fluid is often used to remove the small metal chips.

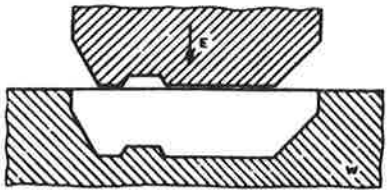

figure 1: Die-Sinking EDM-process.

Besides the Die-Sinking process, there are two other important Spark Erosion processes: wire-cutting and grinding. However, the Die-Sinking process is the only one worthy to simulate, because for this process the final results after a large amount of discharges depend largely on the electrode wear.

The applied Research Department of CHARMILLES TECHNOLOGIES S.A. Geneva has developed a mathematical model to simulate the intermediate and final results of the processes on their Spark Erosion machines, in particular for Die-Sinking [2]. The original implementation was in PASCAL, but the execution time on a PC took several hours. So, other and faster solutions had to be found.

In this paper a comparison is presented between different implementations for the rather complex simulation model for the Die-Sinking Spark Erosion process. Therefore, we considered three kind of processors: a general purpose processor (1), a domain specific signal processor (2), and an application specific signal processor (3). We compare the final execution time after one hundred cycles for a specific example, chosen as reference.

Each implementation is obtained on a nearly full automatic way, starting from the same high level behavioural description for each of the three processor types.

II. Algorithm description

The simulation model described in this context does not apply for all imaginable three dimensional (3D) geometries. Only those 3D geometrical shapes of tool and workpiece which can be generated by a 2D geometric function are considered here, i.e. geometries with a rotational or a longitudinal symmetry.

TH0316-0/90/0000/0236/$01.00 © 1990 IEEE

Thus, we are talking here about 3D shapes which can all be transformed into a 2D concept. Other 3D shapes which do not match the above definitions are not considered in this study though, the applied techniques and algorithms can be extended for these applications.

An important point in the model was the way to describe both the workpiece and the tool-electrode. This is done by two discrete functions: for the workpiece W[i]:i=1..m, and for the tool-electrode T[j]:j=1..n. Each discrete function contains two coordinates, one for the x-direction and the other for the y-direction. These discrete points have to be successively filled along both contours in the same direction. Special attention must be given on the fact that the computing time will increase with the square of the number of considered points, but the accuracy of the final results is of course also be related to that number.

One can imagine that metal removal and tool wear consist of removing thin surface layers successively at the tool and workpiece. Based on this assumption, the spark erosion removal can be decomposed into five basic steps:

1. Workpiece removal
2. Tool feed (servo)
3. Tool wear
4. Tool feed (servo)
5. Shape regeneration

Each simulation cycle involves the removal of a slight surface layer at the workpiece and the tool-electrode respectively. The new coordinates of the points are calculated in the first and the third step, respectively for the workpiece and the tool-electrode. This works as follows: for each point first the distance to the nearest point of the opposite shape has to be found. With this distance we can read from an experimental measured curve the local removing volume.

As known, in EDM the working gap between tool-electrode and workpiece remains almost fixed. In reality the working gap varies slightly around a mean value. In this model it is assumed that the minimum distance between the two electrodes remains fixed. This recalculation is done after each removal step, in step two and four.

The last step of each simulation cycle is the shape regeneration. This means that we will control the two discrete functions, for reasons that their length will surely vary during the simulation, and hence the number of points of both functions has to be adapted. By doing this, we are sure that the point density, and even the accuracy can always be guaranteed.

The execution of this five step will remove one small layer of both electrodes. By repeating this for a great number of times, the tool-electrode will sink gradually into the workpiece.

The most occurring activity in this algorithm is the calculation of something like c=a*a+b*b. Each cycle we have to search, for each point of both electrodes, between all the points of the opposite electrode for that point with the smallest distance. This is done in step one and three. In the fifth step the distance between successively points of each discrete function has to compute for controlling and adapting if necessary.

For the description of our signal flow graph we choose the functional language SILAGE [3][4]. We had three good reasons for doing this: first of all, we can simulate this description by using the s2c-compiler of IMEC, which transforms the non-procedural SILAGE description into a procedural and executable C-description. Secondly, EDC-Heverlee has developed recently a compiler to transform the description in an optimal machine-code for the TMS320C25 and TMS320C30 signal processor. Finally, SILAGE is also be used as description language for the Cathedral Silicon Compilers developed at IMEC, which translates the signal flow graph in a single multi-processor ASIC. Each of this tools gives possible solutions for the Die-Sinking Spark Erosion simulations. In the next sections we will discuss them separately.

III. A General Purpose Processor

This is of course the easiest implementation, with the minimal cost but the highest execution time. The bottle-neck here is of course the fact that both instructions and data are stored in RAM, and that they travel on the same bus. Another important thing is that all the multiplications have to be compute on an iterative way, by shifting and adding on the ALU.

As mentioned an important point in this study was that all implementations started from the same behavioural SILAGE-description. Thanks to the s2c-compiler from IMEC we were able to obtain a first -software- implementation in C [5].

We tried first to optimise the original description: so we changed the floating-point operations in fixed-point operations, we found faster methods to get the nearest distance to the opposite electrode, we removed as much as we could large computations like square-roots, divisions, etc. Simulations on an APOLLO 3500 workstation have shown us that the execution time was reduced from several hours to several minutes, thanks to all these corrections. An example of such simulations is shown in figure 2.

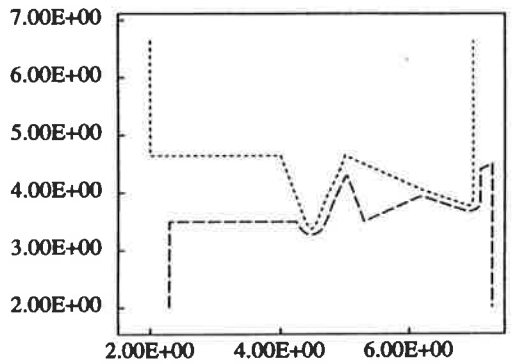

figure 2: Example of EDM Die-Sinking simulations.

IV. Domain Specific Signal Processors

As explained above, the main operation in the algorithm is the calculation of something like c=a*a+b*b. Using a hardware multiplier with accumulator this can we done very fast. Some

commercial processors like e.g. the Motorola56000 family and the TMS320 family make use of this advantage. For this application, we picked out two processors of the last family. This family supports realtime digital signal processing (DSP) and computation-intensive applications in areas of telecommunications, modems, speech processing, instrumentation, numeric processing, machine controlling, etc.

The TMS320C25 [6] is a 16-bit processor with Harvard-type architecture, in which program and data memory reside in separate address spaces. This allows a full overlap of instruction fetch and execution (100 ns cycle time). Internaly, full-speed execution is obtained by maintaining two separate bus structures, for program and data, with the possibility to exchange data between the two buses. When we store the program instructions in the on-chip ROM of 4K words (more than enough), and the experimental measured relations between gap-width and removal volume in the on-chip RAM of 544 words (or 272 words for each relation), than we use this processor in a very efficient way for our application. Only the coordinates of the discrete functions have to be stored out of the chip.

At the European Development Center (EDC -Heverlee-Belgium) they have developed compilers to translate a single flow graph from a behavioural description (SILAGE) into an efficient instruction-list for the TMS320. The efficiency of this compiler can be demonstrated by comparing it with the C-compiler of Texas Instruments: the instruction-list after using this compiler was more than three times longer. The reason is that this compiler first translates the C-code into assembler as an intermediate step. The final results for this processor type is that the execution takes 34.0 seconds for a single processor configuration, and 22.8 seconds for a two-processor system. Using the SWDS simulator [7] we were able to execute and test the obtained instruction-code as a verification step in our design.

The TMS320C30 Digital Signal Processor [8] is a very recent 32-bit third generation micro-processor in the TMS320 family. The 60-ns cycle time allows it to execute more than 33 MFLOPS. High performance is gained through its large on-chip memories, concurrent DMA controller, full floating-point multiplier and ALU, instruction cache, etc. There is a high degree of parallelism and very powerful instruction set.

The instruction-list for our application on the TMS320C30 is also obtained by using a compiler of EDC. Thanks to the parallel-operation instruction set the number of instructions to implement our algorithm on this processor is 2259 vs. 3195 for the TMS320C25, or less than 70 %. This means that the execution of our algorithm takes 16.4 and 10.8 seconds, respectively for a single and a two processor configuration.

We can conclude that a domain-specific signal processor like those of the TMS320 family can be a good solution to execute this application in an acceptable time, without extremely high cost or other disagreeable problems. Comparing it with a general purpose processor, as we saw in the previous section, the execution is an order of 10 times faster.

V. Application Specific Digital Signal Processors

The third type of processors we investigate in this study is the ASIC or Application-Specific Integrated Circuit. This means that datapath and control logic are both designed in an optimal way for each specific application separately. It is obvious that such a solution will lead to the fastest implementation, but also the most expensive. The datapath can execute several operations in parallel on different chosen execution units (EXU's), instructions are stored in a separated ROM, data can travel on several buses, etc.

The VSDM-group of IMEC is intended to study design methodologies for VLSI-systems. The result of this work is 'Cathedral', a set of silicon compilers for the design of integrated digital circuits, starting from a behavioural description of an algorithm. At this moment there are four different Cathedrals, each of them with a target architecture for a target application area:

	CATH-1	CATH-2	CATH-3	CATH-4
Application Domain	linear DSP with simple control structure	complex DSP in the medium throughput	high through-put with irregular signal flow	high through put with regular signal flow
Target. Architecture	hardwired bit-serial architecture	microcoded multi-proc. architecture	hardwired cooperating data-paths	systolic array

The spark erosion algorithm is rather complex, so it was clear that for this application the Cathedral-II Silicon Compiler with microcoded multi-processor architecture [9][10] is the best Cathedral-solution to design an ASIC.

The design methodology which is used in the Cathedral-II Silicon Compiler is based on the so-called 'meet-in-the-middle' strategy.
This means that a distinction is made between the system design and the silicon design. When the system design attains the level of functional building blocks (FBB's), we can make use of the silicon module library [11]. In a full-custom design, more than 60 % of the design time is spent on layout and verification. With the meet-in-the-middle strategy, the reusable silicon modules reduce these design tasks.

As mentioned above, for complex DSP-applications in the medium throughput, a microcoded multi-processor architecture is developed into Cathedral-II. This target architecture gives a fixed framework within the structure of each particular design can be freely chosen. There are three levels of hierarchy :

1. chip level: at this level, the algorithm can be divided into a number of independent processors, with dedicated interprocessor memory for communication. Such a division is not always trivial, but can lead to a great decrease of the cycle-count.

2. processor level: each processor has his own datapath and control logic. The datapath consists of a number of execution units that communicate with each other via buses. Examples of available

EXU's are the ALU, RAM, ROM, multiplier, address computation unit, etc. Each clock cycle the controller generates a microcode word that specifies the function of all parts of the datapath in that particular clock cycle. These words are stored in the instruction ROM. The program counter can be influenced by flags from the datapath via a finite-state machine (FSM).

3. execution unit level: this is the lowest level in the hierarchy. The EXU's are formed from a number of FBB's as adders, multiplexers, register files, etc. These FBB's have parameters such as number of registers, wordlength, adder-type, etc.

In the interactive concept [12] of Cathedral-II the system designer is able to give structural hints to the compiler at the highest level. This can be done by adding high-level directives to the behavioural description (e.g. allocate 3 ALU's, operation a>b on comparator nr 2, all multiplication on a hardware multiplier or on an ALU, etc.). The designer can easily iterate the synthesis process from his workstation until he is completely satisfied with the results. During the interaction the module-generation environment is consulted in depth to give early information about the expected chip area and timing performance. The result of the synthesis is finally sent to the floorplanning environment.

In this ASIC-study we compared many alternative implementations. The best solution was an implementation with two ALU's, one hardware multiplier and only on-chip RAM's. A summary of the architectural results are given below:

EXU	SPEC's(bits)	R1	R2	delay	mm2
ALU_1	X=16-bit;Y=16-bit	11	6	103 ns	1.1*2.7=2.9
ALU_2	X=16-bit;Y=16-bit	9	7	101 ns	1.1*2.5=2.8
MULT	X=16;Y=16 -> 32-bit	2	4	122 ns	2.0*3.4=6.8
ACU	address=12;data=16	7	3	116 ns	2.3*0.8=1.8
ROM_CTRL	in=5;out=16;size=21	-	-	31 ns	0.8*0.7=0.6
RAM_1	1238 words;16 bit	2	4	76 ns	9.4*3.0=28.3
RAM_2	1030 words;16 bit	1	2	73 ns	9.1*3.0=27.7
BUS_1	16 bit	-	-	15 ns	0.2*1.1=0.3
BUS_2	16 bit	-	-	15 ns	0.2*1.1=0.3
BUS_3	32 bit	-	-	15 ns	0.2*2.1=0.5
CONTROLLER	163 words of 112 bit	-	-	160 ns	1.8*3.5=6.3

cycle-count : 552 cycle-time: 160 ns total area: 78.2 mm2

The same configuration but with only one ALU takes 727 cycles. If we remove also the hardware multiplier the total cycle count is 4852. For the above one-processor ASIC with 2 ALU's, 1 hardware multiplier, 1 Address Computation Unit (ACU) and only on-chip RAM's, the total execution time for our spark erosion simulation application is 12.4 seconds. We notice that the timing and area estimations are done for the current 1.6 micron module library. Another, 1.25 library will be installed shortly, and will lead to less area and shorter delays.

It was also possible to partition the algorithm at high level, what resulted in a multi-processor system with switching RAM's as intercommunication memory [13]. We did this for a configuration with two processors (cycle-time: 120 ns), and the resulting execution time decreased in only 6.5 seconds.

VI. Conclusions

In this paper the results of a comparative study between different implementations of a rather complex application are presented, and they show us that Die-Sinking Spark Erosion Simulations don't have to be considered in the future as unrealistic. After having rewritten some parts of the original algorithm, without changing something functional, we worked out several implementations automatically on three different kind of processor-types, all starting from the same behavioural high level description. To achieve this, we used three excellent compilers: the s2c-compiler for the implementation on a general purpose processor, the EDC-compiler for implementation on a TMS320 Digital Signal Processor, and the Cathedral-II Silicon Compiler for the ASIC implementation. We can conclude that the corresponding execution times are respectively from the order of several minutes, a halve minute and several seconds.

VII. Acknowledgements

The author wishes to thank Mr. Claude Tricarico and Dr. Dirk Dauw of CHARMILLES TECHNOLOGIES S.A. for all the information concerning this application, and all members of IMEC's VSDM division for their contributions to the presented work.

References

[1] Kruth J.P., "The EDM process, and its applications", ITB university of Bandung-Indonesia, KUL university of Leuven-Belgium.

[2] Tricarico C., Delpretti R., Dauw D., "Geometrical Simulation of the EDM Die-Sinking Process.", Charmilles Technologies S.A., Geneva,1988.

[3] Hilfinger P.N., "SILAGE: a Language for Signal Processing", University of California, 1984.

[4] Scheers C., Nachtegaele L., "SILAGE to C compiler", IMEC Laboratory, 1989.

[5] Kernighan B.W., Ritchie D.M., "The C Programming Language", Prentice Hall, 1978.

[6] Texas Instruments, "TMS320C25 Digital Signal Processor", Product Description,1986.

[7] Digital Signal Processor Products, "User's Guide: Software Development System (SWDS)", Texas Instruments,1987.

[8] Texas Instruments, "TMS320C30 Digital Signal Processor", Product Description,1988.

[9] De Man H., Rabaey J., Vanhoof J., Goossens G., Six P., Claesen L.," Cathedral-II: a computer-aided synthesis system for digital signal processing VLSI-systems", IMEC, april 1988.

[10] Van Meerbergen J., De Man H., "A true silicon compiler for the design of complex IC's for digital signal processing.", IMEC-Philips, 1988.

[11] De Keulenaer H., "Definition of the Cathedral-II module library", IMEC, 1988.

[12] Goossens G., Lanneer D., Vanhoof J., Rabaey J., Van Meerbergen J., De Man H., "Optimization-based synthesis of multiprocessor chips for digital signal processing with Cathedral-II", IMEC, may 1988.

[13] Decaluwe J., Rabaey J., Van Meerbergen J., De Man H., "Interprocesor Communication in Synchronous Multiprocessor Digital Signal Processing Chips", IMEC, 1988.