Auriga2: A 4.7 Million-Transistor CISC Microprocessor

J.P. Tual, M. Thill, C. Bernard, H.N. Nguyen
F. Mottini, M. Moreau, P. Vallet

Hardware Development Paris & Angers
BULL S.A.
78340 Les Clayes-sous-Bois, FRANCE
Tel: (+33)-1-30-80-7304
Fax: (+33)-1-30-80-7163
Mail: J.P.Tual@frcl.bull.fr

Abstract- With the introduction of the high range version of the DPS7000 mainframe family, Bull is providing a processor which integrates the DPS7000 CPU and first level of cache on one VLSI chip containing 4.7M transistors and using a 0.5 µm, 3Mlayers CMOS technology. This enhanced CPU has been designed to provide a high integration, high performance and low cost systems. Up to 24 such processors can be integrated in a single system, enabling performance levels in the range of 850 TPC-A (Oracle) with about 12 000 simultaneously active connections. The design methodology involved massive use of formal verification and symbolic layout techniques, enabling to reach first pass right silicon on several foundries. An architectural overview of the CPU with emphasis on several original aspects of the design aspects (synthesis, verification, symbolic layout) will be discussed in this paper.

I. INTRODUCTION

A CISC microprocessor containing 4.7 M transistors has been developed. This circuit is used in high-end mainframes designed for large transactional and business applications. The clock frequency of the chip is 66 MHz with less than 5W dissipation. The chip includes a private data cache containing 64 KB SRAM (DAT) and its directory (DIR) and a CPU made of three cooperating microprogrammed units, Effective Address Development Unit (EAD), Binary and Decimal Processing Unit (BDP), double-precision Floating-Point unit (FPP). The Auriga2 microprocessor supports a set of 300 instructions implemented in on-chip microcode with complex addressing modes (indexing mode, 16 levels of indirection mapped through 4 levels of converting tables). The instructions range in execution time from 1 cycle to 500 cycles. The chip supports built-in self test, partial scan, in-circuit hardware and firmware for code debugging, remote maintenance and diagnostics, as well as hardware to support parallel multi-processor architecture. It is used in a family of systems able to handle up to 24 such microprocessors, capable to support 10 000 simultaneously connected users. For the development of this complex circuit, a system level design methodology has been put in place, putting high emphasis on high-level verification issues. A lot of home-made CAD tools were developed, to meet the stringent performance/area constraints. In particular, an integrated Logic Synthesis and Formal Verification environment tool has been developed, to deal with complex circuitry issues and to enable the designer to shorten the iteration loop between logical design and physical implementation of the blocks. In particular, all blocks of the chip (including ROMs), representing over 1.3 MT, were formally verified before complete chip simulation.

The low maturity of target technologies (0.5µm CMOS with triple-level-metal) available at the time the design started (late 91) led the design team to adopt a symbolic layout methodology enabling to easily port the design across several foundries, offering comparable processes. In fact, the difficulty to obtain good yields during the design and production phases led the design team to submit the chip to four different manufacturers (2 inUS, one in Asia [A,B,C], one in Europe [D]). The typical porting effort represented a few men-month in average. The die size is 15x15 mm2, with 320 active pins and 88 power/ground pins. The chip is packaged in a custom 419 pad CPGA.

II. CHIP ARCHITECTURE OVERVIEW

A. First level cache

The Auriga2 private cache is part of a two-level cache/three-level busses hierarchy implemented at system level. The second level cache is shared by four Auriga2 microprocessors connected though a 16 B wide, 1 GB

bandwidth bus (ACB) operating at 16 ns. It is physically implemented under the form of 4MB fast access SRAM (< 10 ns access time) and three custom ASICS external to the microprocessor (one chip for data RAM management, consistency checking and address control on the system (ASB) and CPU control (ACB) busses, and two identical chips for data buffering). The Auriga2 private cache is 64 KB 16 levels set-associative, implementing a store-into write strategy. Addressability is 2^{36} Bytes at system level and replacement algorithm is pseudo-LRU. Data consistency is ensured for a fixed 64 B block size. The DIR unit is in charge of address management (ACB and CPU interface) and address storage. The DIR unit includes 2 dedicated blocks (SCACB and SCEAD) for managing the requests coming from the outside and the Auriga2 CPU (though the ACB bus). It also contains an address interface block with both the ACB bus and the CPU, and the cache directory (RAMDIR) together with its LRU block. DAT performs a 16 Bytes read per cycle and communicates with the external world through a 16 bytes bus (1GB/s). The DIR unit is able to handle 2 operands of variable formats and lengths concurrently with the instruction prefetch. Two misses can be handled at a time and immediate move can be performed without request to the executive units. The DAT unit consists on a 64 KB SRAM and two identical control and operative parts. The SRAM is organised into 16 times 256*16B words. The operative parts include barrel shifters, write operator and 11 registers (3 instruction registers [16B], 2 operand registers [64B and 48 B respectively], 1 trap register [16B] and 5 bus emitter/receiver [64B]registers). The control part decodes the commands coming from the DIR unit. DIR sends to DAT all signals enabling to access or to control the access to the SRAM or to control the various registers in the DAT operative parts.

B. Executive unit

Three microprogrammed units compose the executive parts of the CPU : The Effective Address Development (EAD) unit computes virtual and real addresses with checking of the read/write/execute rights and performs general functions such as instruction or operand fetch scheduling, instruction counter updating, control of the instruction bus; it contains a 32x36 bit fast associative memory, with three levels of precharge, fully addressable in 5.5 ns (hit plus read); the Binary and Decimal Processor contains the main 32 bits data path of the chip : shifter, binary and decimal adder, logical operator, scanner, binary to decimal converter and length counter; the Floating Point Processor is executes floating point operations and contains a 64-bits adder and a 56 x 16 bits Booth-Wallace fast multiplier (96 ns). Those executive processors communicate through three local busses (instruction, operand and result see Fig. 1 below).

The chip clocking is done using a two-phase non-overlapping clocking strategy. Instruction processing consists on an 3-stage pipeline that executes short instructions with operand in memory in one cycle. The principles of the pipeline are the following: a) The microprograms in BDP and FPP start and terminate the execution of an instruction synchronously; b) The EAD unit can anticipate up to 3 instructions on BDP and FPP; c) The last microinstruction of a given instruction triggers the decoding and execution of the next instruction; d) The last microinstruction of an instruction allows EAD to terminate before BDP and FPP. The pipeline is decomposed into 3 stages : address computation (3 steps) in EAD followed by data fetch (2 steps) in the cache and execution (3 steps) performed by BDP. This firmware-controlled pipeline greatly contributes to simplify the hardware and provides a high level of flexibility for the debugging. For example, five cycles are requested to execute the instruction (Mem + Register) ⇨ Mem : EAD and BDP execute in one microprogrammed cycle with EAD starting one cycle ahead of BDP. In addition, on an anticipation of at least 3 cycles by EAD, BDP can shunt any unconditional branch instruction. As a result, the unconditional branch has a 0 cycle visibility.

In total, 300 instructions are implemented in on-chip microcodes of variable length. The execution of each instruction is controlled by three cooperative microprograms, one for each executive unit (ie. EAD, BDP and FPP). Those microprograms execute concurrently, synchronising themselves on inter-unit signals such as instruction ending, data or bus ownership exchanging. Within each unit, the microprograms are coded in two levels: compacted m-words located in control stores and expanded m-words implemented in mROMs. The control stores are implemented in the Auriga2 chip (4K x 48 bits for BDP, 4K x 40 bits for EAD and 1K x 40 bits for FPP) with an external 32K x 48 bits RAM extension. Analyses of million of executed instructions are performed in order to dispatch the instruction set between the ROM and RAM implementations (in average the RAM access is twice longer than the ROM access).

Table 1 hereafter summarizes transistor count and area of the five major units in the chip.

TABLE I

TRANSISTOR COUNT AND AREA SUMMARY OF THE AURIGA2 CHIP MAIN UNITS

Unit	Transistor count	% Area
EAD	350 K	15,9
DIR	350 K	12,8
DAT	100 K + 3 400 K (SRAM)	38,7
BDP	350 K	10,5
FPP	250 K	9,6
ROUTING	-	12,2

Fig. 1 below is a block diagram of the complete Auriga2 microprocessor.

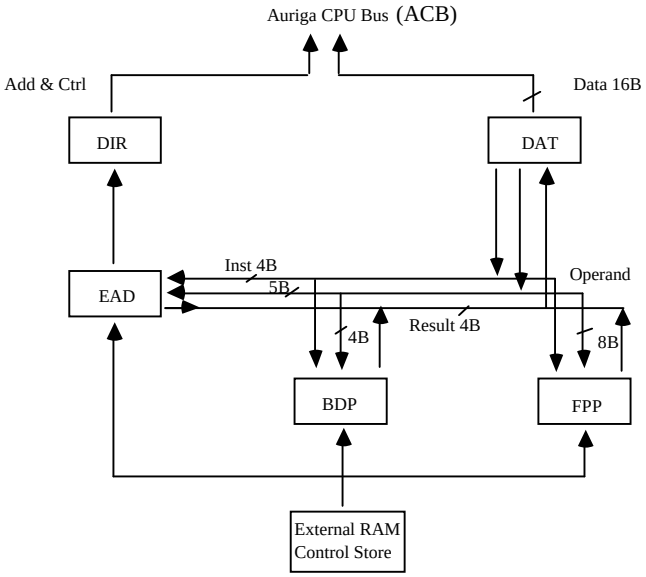

Fig. 1 : Block diagram of the Auriga2 microprocessor

III. DEVELOPMENT METHODOLOGY

A. Global aspects

In many ways, the design process for the earlier VLSI DPS7000 mainframe computer set the tone for all subsequent VLSI designs at Bull and has been reused for the Auriga2 design. The process was characterised by massive simulation of a full system model, running in all modes of operation. Two types of simulation models were used. The first type was designed as a high-level software breadboard used to develop and check out the microcode before the chip hardware was available. The second type was developed as a relatively detailed register transfer level model of the actual physical partitions and design concepts of the chip themselves. This model was dedicated to pilot the synthesis tool suite or used directly by the logic designers to develop the circuit-level representations of parts of the design.

The two types of models were described in a proprietary hardware description language oriented toward synchronous design (called LDS) and where object (signal, latch) names follow an imposed naming convention that guarantees unique identification and improves readability. This language can easily be mapped on to a synchronous subset of VHDL. The detailed level of modelling along with the quality of the simulation test cases provided a high quality of functional validation. Several billion cycles were simulated in the final non-regression steps. In particular, several mechanisms have been defined to provide flexibility in fixing some complex multiprocessing scheme errors which are usually detected rather late; those mechanisms include: usage of an associative memory to inhibit specific microprograms, modification of the microcode contained in the external RAM, definition of "simplified" operating modes to by-pass some complex mechanisms, usage of spare cells for late metal patch (for incremental performance improvements).

Due to the large amount of data, a Zycad SDE hardware accelerator was also used for non-regression simulation.

B Chip level aspects

The CPU and caches of the DPS7000 system are composed of VLSI circuits which are decomposed into macro-blocks during the early floorplanning phase. While structured logic blocks (e.g. operators of the data path, ROMS) are either designed manually or using general purpose [1] or specific module generators, logic synthesis is applied to the design of the all control parts (including sequencers).

In an effort to improve design productivity, rigorous and exhaustive methodology definition as well as search for adequate CAD tools were initiated early in the project (c.f. [2], [3]). Intensive usage was done in the project of new logic synthesis and formal proof methods. In particular, all blocks of the chip (including ROMs), representing over 1.3 MT, were formally verified before complete chip simulation This methodological effort enabled to reach a "first-pass right" silicon and is estimated to have provided over 30% productivity improvement with respect to the previous similar CPU design.

Fig.2 hereafter depicts the overall chip design methodology followed.

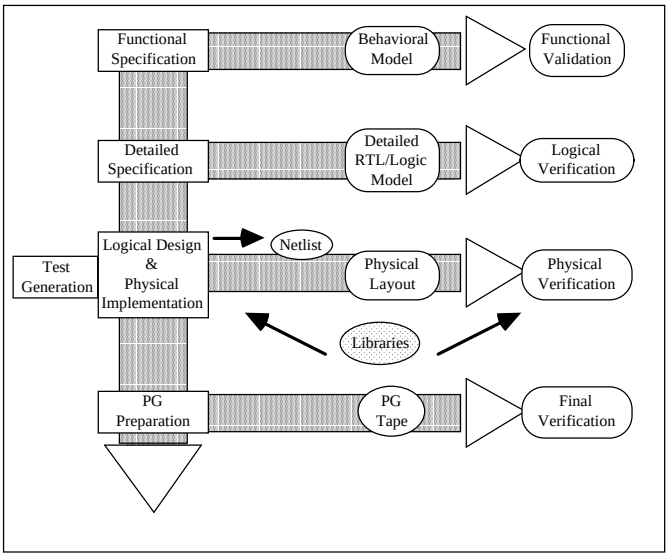

Fig. 2: Complete chip design flow

C. Logic design methodology

The logic design methodology is targeted for a mixed custom and semi-custom design environment (Fig. 3). In the adopted top-down approach, the design starts at the highest level and progresses toward lower levels the upper level in the hierarchy defining the requirements of the design for the lower levels. Performance tuning is achieved during the planning and design phase, providing the ability to perform a one-pass design (indeed, two passes are necessary to meet all the performance constraints). At each level of the abstraction, validation is performed by simulation of the whole model of the CPU.

D. Logic Synthesis Requirements

The logic style of the synthesised blocks is a mixed regular and random logic associated with a multiclocking and multiphase scheme. The usage of multi-input latches, precharged and tristate signals represents another characteristics of the synthesised logic. In addition to these considerations, the main design constraints for these blocks are drastic timing constraints and narrow area margins.
As a result of these considerations, synthesis is a highly iterative process and our main concern has been to ensure a rapid and smooth convergence of this process toward the desired result. This implies the ability of the logic synthesis system to take into account hand tuning needed to improve logic speed in problem areas (e.g. the use of non-standard gates). In addition, logic design was synchronised with the semi-custom physical design process for complete design verification.

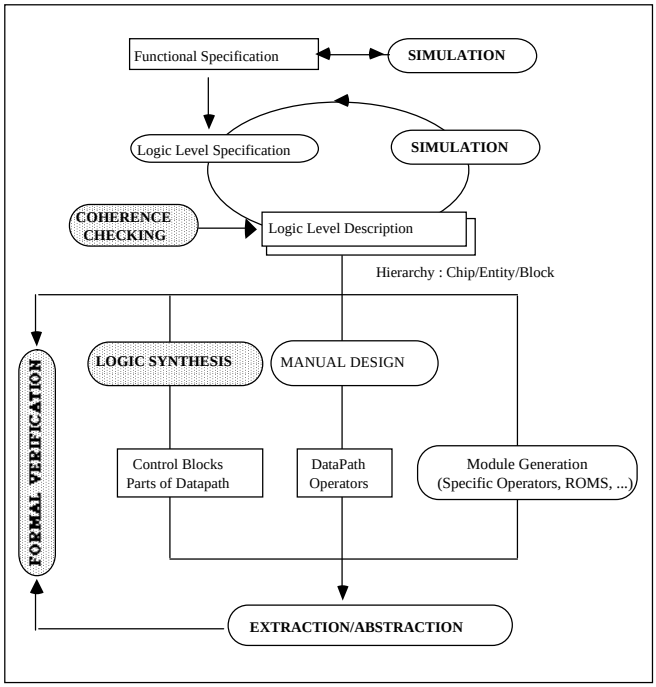

Fig. 3: Logic Design Methodology

E. Formal Verification Requirements

Checking plays a significant role in our methodology. Among the many different checking functions, formal techniques are used to perform logical-to-physical checking and model verification. Logical-to-physical checking is intended to verify that the physical implementation of the blocks matches their logical descriptions while model verification is used to prove the equivalence of models (e.g. different representations or versions of a block) and to check some basic properties (e.g. scan path connectivity, exclusive control signals at a multiplexing node, ...etc.). The main requirements for the usage of formal verification is its ability to handle large and complex logic (e.g. manipulation of Boolean expression with hundreds of variables in its support).

F. The BONSAI system

Both Logic Synthesis and Formal proof have been embedded in a single tool, called BONSAI. The fig. 4 below shows a description of the resulting SW architecture, in interaction with the complete design script.

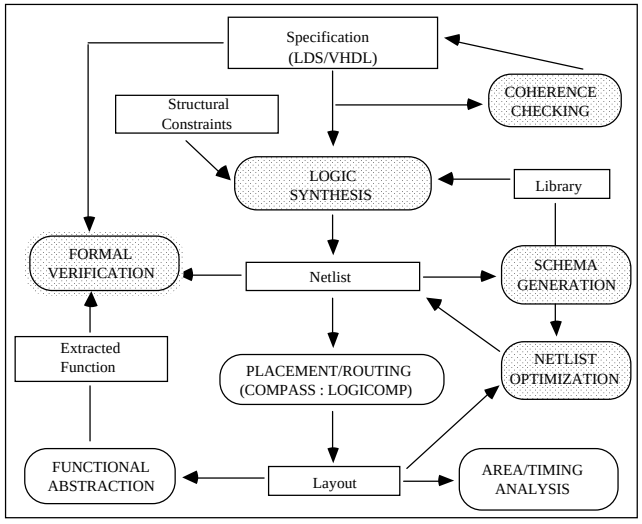

Fig. 4: Features of the BONSAI system

The main feature of the system is built on the concept of a Boolean Network as detailed in [4]. However, it is necessary to extend those definitions in order to type the nodes and to handle sequential components although only the combinational parts interconnecting these components are considered in our work.

Def.1: A Boolean Network N is a 3-uple (**F**,PO,M) where **F** = {**F**j / j=1,...,m} is an incompletely specified function.

With each **F**j is associated a logic variable v_j in the set V = {v_j / j=1,...,m} called the intermediate variable set. The specified Primary Output set PO $\subset$ {1,...,m} identifies the subset {v_j / j $\in$ PO} of the observable outputs of the $\mathbf{F}_j$ while the memory set M $\subset$ {1,...,m} defines the subset {v_j / j $\in$ M} of the memorised variables. In our case, these memoried variables are used to model latches. The behaviour of a latch with input d, output q and control c is defined by the following function :

if c(t+1) then q(t+1) := d(t+1)
else q(t+1) := d(t) endif;

Def. 2: A representation of a Boolean Network is a 4-uple (F,DC,PO,M) such that {F_j / j=1,...,m} (resp. {DC_j / j=1,...,m}) is a set of m given representations of the on-sets (resp. DC sets) of F_j.

Let us note the support set of Fj by SUPP(F_j), then we will have t = / $\cup$j SUPP(F_j) / > m. The last t - m logic variables v_j, j > m have no associated F_j and are identified as the Primary Input set PI. In addition, we can state without lost of generality that for all j=1,...,m, SUPP (DC_j) $\subset$ SUPP(F_j).

Note that the "don't care" set plays an important role in the verification process: for an operator of a data path, the set of control signals is most of the time generated by a control logic block and therefore it is not complete. Besides, the DC set is also necessary to handle tristate signals.

Notation : In the following, we note O = PO$\cup$ M and also I = PI$\cup$M.

Def. 3: For each variable v_j, j=1,...,m+n we define the *Fan-In* set FIj as follows :
if j > m then FI_j = f else FI_j = {k / $v_k \in$ SUPP(FI_j)}.
Similarly, The *Fan-Out* set FOj is defined as:
FO_j = { k / $v_j \in FI_k$ }.

Def.4: The transitive *Fan-In* and *Fan-Out* sets associated to a variable v_j are defined by:
i) TFI_j = { k / $\exists$ [k_1,...,k_r] such that: k_1 = k, k_r = j and for q = 1, ...,r-1, q $\in$ FI_{q+1}
for q, 1<q<r, q$\notin$ M }
ii) TFO_j = {k / j $\in$ TFI_k}

As shown on Fig. 4, the main modules of the BONSAI system are the following :

a) Assistance to specification : The system verifies that the Boolean Network is well-defined, i.e. satisfies the following criterions :
i/ $\forall$ j =1,...,m, j $\notin$ TFIj
ii/ $\forall$ j $\in$I, $\exists$ i $\in$ O such that i $\in$ TFOj.
iii/ The control signals at all nodes of M are mutually exclusive
These rules express the facts that (i) there is no functional loop and (ii) every intermediate variables or primary inputs are effectively used; checking these criterions allows to detect errors in writing the model.
In addition BONSAI provides consistency checking and symbolic simulation, in order to detect incoherence in the specification such as unused or unassigned variables, ill-defined conditional assignment or multiple assignments exclusivity of the control signals at a multiplexing node.

b) (Re)Synthesis with structural constraints: The synthesis procedure is classically implemented in two steps: Logic minimisation and technology mapping. While logic minimisation adopts a symbolic method to transform the initial Network into a primary and non redundant Boolean Network [5], the technology mapping is based on a mixed algorithmic and rule-based approach. The rule-based mapping procedure is used improve the netlist produced by the algorithmic step and to solve local problems such as

fitting fan in/Fan out, synchronisation of latch controls, etc ...

c) Formal Proof : The role of the Formal Prover is to check for the equivalence of two Boolean Networks defined as follows :
Two Boolean Networks N1=(F1,PO,M) and N2=(F2,PO,M) are said to be equivalent if :
" $\forall j \in O$ (= PO$\cup$M), $z_1(j) = z_2(j)$ on the definition domain of j"
[where z denotes the IO map of a Boolean Network (as defined in [4]) with the following extension z : I --> O instead of z : PI --> PO)].
The definition domain of a node is defined as the complement of the set ($\cup DC(k)$ / $k \in TFI_j$).

Practically, the adopted approach is based on a canonical representation of Boolean functions called Compacted Decision Diagram [3] which is an outgrowth of the Binary Decision Diagram [6]. Used in conjunction with a swap strategy, this representation allows to handle efficiently large designs by reducing the memory occupation and improving the computation time.

d) Netlist Optimization : The quality of the synthesized netlist is improved, taking into account the interconnect delays extracted from the layout (cf. [2], [3]).

e) Schematic Generation: Most of the time, large schematics are not readable. To solve this problem, the schematic generator provides local views (eg. path between two nodes, logical cone whose output belongs to O and inputs are elements of I), and introduces timing barriers (represented by memory nodes or I/O ports) to decompose the network into successive logical slices .

IV Implementation-second sourcing aspects

The Auriga2 microprocessor is implemented in a 0.5 µm, triple-level-metal 5 Volt CMOS process that allows power savings and superior circuit flexibility. The physical design was done with two main objectives: to allow optimisation when needed, with a minimum loss of space with respect to manual design; to allow easy technology migration within a range of compatible CMOS processes. To achieve these objectives, the physical implementation adopts a symbolic (grid-based) approach with the usage of primitives and the ability to expand into real coordinates with respect to a specific process design rules. The need to rapidly switch within a range of compatible processes was dictated by two main reasons : necessity to work with the most advanced CMOS processes not fully stabilised neither optimised, generating a high risk on prototype silicon availability; necessity to reach the optimal cost/performance ratio for the final circuit in production, which is now effective since mid 1994.

The symbolic approach has proven to be very effective in that case, since the complete CPU has been successfully transferred in four different processes (2 at 5 Volts, 2 at 3.3 Volts) with a minimum of redesign effort needed in each case (typically, a few men-month). Figure 5 shows a microphotography of the chip in the 0.5µm triple-metal-layer salicided CMOS process of founder D characterised in Table II. Table III shows the typical yields measured at wafer level from the 4 different founders and Table IV the performance obtained on various type of oscillators.

TABLE II

Process summary (Silicon founder D)

Lgate	0.5 µm
M1	1.75 µm pitch
M2	2.25 µm pitch
M3	4.5 µm pitch
Contact	0.6x0.6 μm^2
Via1	0.6x0.6 μm^2
Via2	0.6x0.6 μm^2
$Idsat_N$	490 µA/µm
$Idsat_P$	250 µA/µm

TABLE III

Measured yield from the various foundries

	% of good dies (probe yield)
Foundry A	1,8
Foundry B	6,9 to 10
Foundry C	6
Foundry D	13 to 40

TABLE IV

Measured performance (ring oscillators) at 5 Volt

	Inverter (FO=1)	Inverter (FO=3)	Inverter (Metal load: 400µ)
Foundry A	100,5 ps	196,4 ps	253 ps
Foundry B	105,6 ps	199 ps	277 ps
Foundry C	92,3 ps	173,6 ps	241 ps
Foundry D	89,2 ps	164,6 ps	226 ps

V PERFORMANCES

The system level development methodology followed enabled to reach "First Pass Right" silicon on each of the four foundries to which the design was submitted. For the applications in mind, performances have to be measured in Transaction per seconds (TPS-A).

At 50 MHz and for an 8 processor system, measured performance are in the range of 600 TPS-A (Navigational) and 400 TPS-A (Oracle), with up to 12 000 simultaneous connections. New design tuning at 66 MHz and high-range configurations with 24 processors are expected to bring the complete system performance above the 1200 TPS-A (Navigational) and 850 TPS-A (Oracle) levels.

VI CONCLUSIONS

A complex CISC microprocessor representing more than 4.7 million transistors has been developed. This microprocessor is currently used in a family of complex mainframes. The corresponding system architecture can integrate up to 24 such processors, providing cost-efficient solutions for large Data Base/Transaction processing applications. The performance achieved, measured in TPS-A and for a 8 processor configuration, reached the 600 TPS-A (Navigational) and 400 TPS-A (Oracle) levels. For this complex development, the system-level design script included a lot of innovative, home-made CAD tools. Major points of focus were in high-level complete system simulation, user guided synthesis with integrated formal verification, design migration across comparable processes. The symbolic layout methodology used in the project, allowed to easily port the design on four advanced 0.5µ CMOS processes. Based on this experience, new CPUs will be designed, definitely bringing the power of CMOS technologies in the high-end computer segment.

REFERENCES

[1] H.N. Nguyen, L. Ducousso, "PLAYL : A General-Purpose Data Path Assembler", *Proceedings of EUROASIC'90*, Paris May 29-June 1, 1990.

[2] H.N. Nguyen, L. Ducousso, M. Thill, J.P. Tual, P. Vallet , "Logic synthesis and formal verification of the CPU and caches of a mainframe system", *Proceedings of the European Design and Test 1994 Conference*, IEEE Computer Society Press, Paris, France, 1994.

[3] H.N. Nguyen, L. Ducousso, M. Thill, J.P. Tual, P. Vallet, "The logic CAD suite of the DPS7000 mainframe", *Proceedings of the ICCD 1994 Conference*, IEEE Computer Society Press, Cambridge, MA, 1994.

[4] G.D. HACHTEL and R.M. JACOBY, "Verification Algorithms for VLSI Synthesis, *Design Systems for VLSI Circuits*, Martinus Nijhoff Publishers, pp 249-300, 1987.

[5] R.K. Brayton, G.D. Hachtel and A. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis", *Proceedings of the IEEE*, Vol. 78, N° 2, pp. 264-300, Feb 1990.

[6] R.E. Bryant, "Graph-based Algorithms For Boolean Function Manipulation", *IEEE Transactions on Computers*, 35(8), pp. 677-691, 1986.

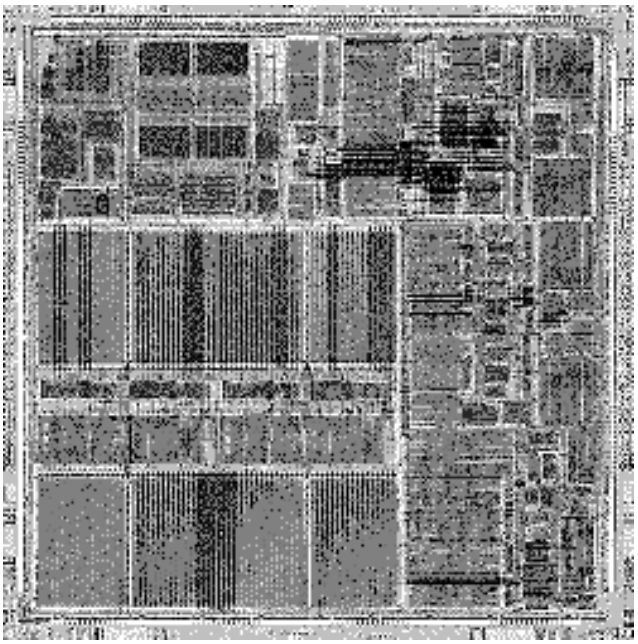

Fig. 5: Photography of the Auriga2 microprocessor