A Forth-Based CAD for System-Level Microelectronic Design

Ilya E. Tarasov (Measurement systems), Veniamin G. Stakhin, Anton A. Obednin (IDM-Plus)

Abstract

As part of a Federal project to reconstruct Russian microelectronics, the Zelenograd Innovation and Technological Center was chosen to coordinate R&D works to develop a set of IP cores and an appropriate software platform for cores integration, modeling, and hardware-software co-simulation. This project is carried out under a state contract, supported by the Ministry of Science and Education of the Russian Federation. This article describes features of use of the Forth-based engine for the decision of a problem of creation microelecrtonic CAD.

Overview

The main subcontractor for this project is the IDM-Plus fabless company, located in the town of Zelenograd near Moscow. The company successfully operates at the microelectronic market, including collaboration with China and Taiwan foundries and the Cadence company. Among their products is a family of 16-bit stack processors (defined as 1894xx), so IDM-Plus is experienced with the stack architecture. Since 2004 a partnership of IDM-Plus and Measurement Systems was established, based primarily on a collaborative research in Forth technologies.

Measurement Systems, located in Kovrov, was founded in 2000 as an R&D company. It manufactures the high-intellectual measurement systems and precision sensors with strong signal processing part. The base technologies of the company include FPGA prototyping of system-on-chip devices, original filtering and statistical algorithms, and a complete set of Forth technologies, including original versions of Forth translators for PC (PM-Forth in 1999, Quark-Forth in 2006), several cross-translators for microcontrollers and FPGA-implemented Forth processors.

Partnership of IDM-Plus and Measurement Systems based on architectural research works and system software development, provided by Measurement Systems, resulted in an RTL design and an FPGA prototype of a new system-on-chip. Its further silicon implementation is being performed by IDM-Plus. This is a common practice for microelectronic design, when separate teams work on high-level and topology levels of a silicon device.

Design flow

Within this project it was necessary to develop an original design flow, suitable for both, worldwide foundries and an 'Angstrem' foundry in Zelenograd. Also, since the cost of mask set grows dramatically with each new technology step, more attention should be paid to the modeling stage [1, 2]. After carefully preformed prototyping and algorithms testing, a decision on acceptability of the silicon implementation must be taken. Furthermore, not only technical questions, but also marketing ones must be resolved, because a silicon implementation of a 130-nm (and deeper) chip is profitable only for high volumes; in case of moderate volumes an FPGA implementation should be considered. Such high risks of getting a non-profitable (or even non-working) device turns silicon developers to assembling large systems from pre-designed, verified building blocks (IP-cores). With an appropriate set of cores the electronic engineer or programmer is able to compose an optimal hardware configuration for the device under construction. From this point of view, it is important to carefully separate architectural

solutions from a low-level topology design. This means that the system-level specialist must be liberated from designing cores at the low level and refocused to rapidly assemble a software model or an FPGA prototype of a new device, when evaluating it in the field.

There are design tools in the market, oriented to high-level system design, like Coware Platform Architect, Mentor Graphics Visual Elite, Xilinx Platform Studio, etc. Some common trends can be highlighted for this class of CAD systems:
· Early integration of embedded software, that enables complex hardware and software cosimulation.
· Modeling at the transaction level (TLM, Transaction Level Modeling), that enables performance speedup compared to RTL level [1]. At this level of abstraction, the developer must use only pre-verified blocks, because at the transaction level we are at risk to write TLM, which can't be implemented with the current level of silicon technology (for example, considering too fast or too large circuit may fail at the layout implementation phase, while the transaction level will be passed successfully).
· Integration with industry standard software tools, such as topology-level CAD software, and, from the other side, mathematical and DSP software tools and high-level languages.
· Using script languages for automated creation of a project and running the design flow in a batch mode. As an example, a Tcl scripting language is widely used in Xilinx software tools, enabling to assemble an FPGA project with a single batch command without any user interaction. This greatly helps in running many iterations while creating a large chip, replacing high cost 'silicon' iterations with cheap modeling iterations on a PC.

Newly developed CAD software, named 'Quark CAD', belongs to the class of system-level design tools and is intended to design a processor-based chips, including multiprocessor system-on-chip (SoC) devices. This project utilizes a software Forth machine, based on the 'Quark Forth' translator, previously developed at Measurement Systems. After analyzing the successful usage of the PM-Forth translator, several features was selected as a basis for this new implementation of the Forth machine:
· Native machine code of x86 processor with separate code and data spaces.
· DLL implementation, interaction with the shell program via the 'Evaluate' word and several exported functions, providing access to stack, memory, and a special ' virtual screen'.
· Output from the Forth-machine, based on a virtual screen in the main memory, similar to a Canvas
object in some object-oriented software development tools. This eliminates the necessity of wm_paint message handling for an application program, while the shell program is responsible for proper rendering the contents of a virtual screen.
· Wide usage of vector words, replacing nearly all system interface words (PIXEL, EMIT, INTERPRET, OK, etc.) for maximal portability.

While developing the Quark CAD system, a baseline version of Quark Forth was adapted to the new task. First, its assembler version was replaced with a C++ single quark.h source text for further build-in into the gcc software tool. Due to the Linux compatibility requirement, the Qt library was chosen as the main GUI engine, providing virtual screen visualization, file and GUI operations, and handling the main part of Forth source texts, represented as scripts in the CAD being developed (Fig. 1)

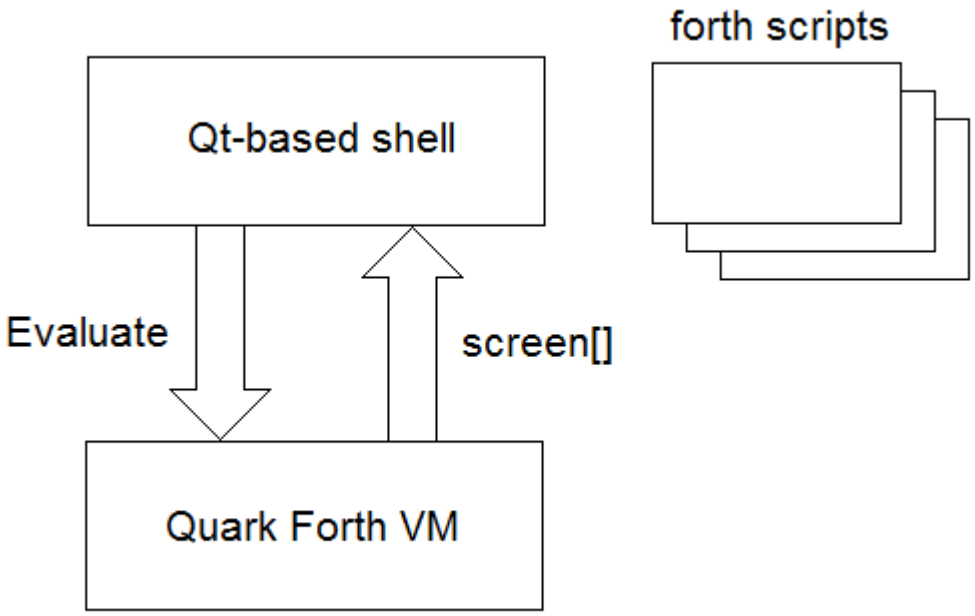

Fig.1. Interaction between C++ Qt and Quark Forth VM in Quark CAD

As shown in Fig. 1, all interactions between the C++ and the Forth parts of the project are strictly limited to a single 'Evaluate' function, which accepts a string to be interpreted by the Quark VM. This VM also is able to open, create, and read/write files without the shell, because basic POSIX-compatible file operations are wrapped by Forth words. So, simple replacing the scripts to load the VM can change the whole functionality of this software. This looks very useful for such complex area as microelectronic design, because many software components (RTL representations, models, etc.) may require an upgrade or reviewing after CAD release. Representing the main part of CAD functionality as Forth scripts, evaluated by VM at the runtime brings great flexibility to this software and makes the component upgrade process transparent and cheap, because no more replacements of executable files are needed. Indeed, it is a common and predictable feature of any Forth-based software, which has been exploited for newly developed CAD.

Forth based product features

Each component in the silicon chip has a several representations for different phases of the design flow (Fig. 2).

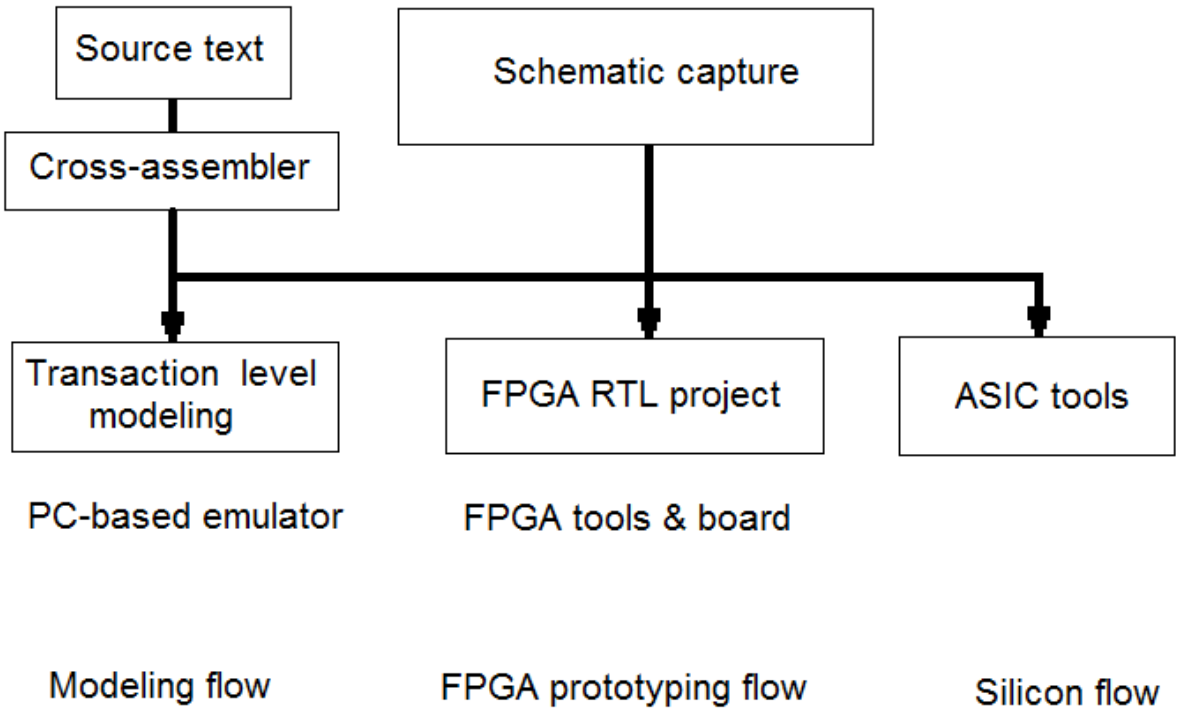

Fig. 2 System-on-chip design flow

As shown in fig.2, there are three main versions of SoC design flow, depending of the specialists involved. At early stages, when project requirements and specifications are still unprecise and need to be reviewed, the long-time FPGA prototyping and especially high-cost silicon implementation is unacceptable. Working with complex models at the hardware and software levels costs less, enables hundreds of iterations, when up to ten iterations of FPGA prototyping may be followed to validate a practical applicability of the developed device. Finally, ASIC implementation will challenge only specific technological, but not algorithmic or marketing issues. In the Quark CAD, all of the design-specific tasks run in Quark VM. There are some types of component representations, used in the design flow, including:

· Structural representation, required for building project graphics and, more important, creating a top-level structural VHDL file. This kind of file is based on a domain-specific Forth extension, which looks like a scripting file for the designer. For example, 'in data_a 32 bit', will create a bus with the name 'data_a' and the 32-bit width for the current component.

· Transaction level model (TLM) representation, written in Forth for each component. There are no domain-specific extension limitations, like for previous representation, so the component designer may use any Forth words he wants, and freely define all necessary words to properly represent the component model. Each model must provide the 'CLK' word, which is executed by the main modeling engine for each component in the system. Executable addresses of CLK words are collected when component models are loaded.

· Assembler representation, required for any type of processor component. There are no restrictions for assembler format, style, or limitations, so the developer may realize both Intel and AT&T formats, as well as a more compact postfix format. All processor cores, provided in the baseline configuration of Quark CAD, use the postfix format, written in pure Forth. No restrictions are present because only requirement to assembler is to create memory image for each processor component in the system. An external assembler program may be also used for

this purpose.
· RTL representation, written in VHDL, but combines to synthesizeable project by configuration engine, running on Quark VM. Each component must have this type of representation for FPGA prototyping and silicon implementation; however, not for TLM. This allows programmers to write their own TLMs without knowledge of VHDL. After modeling is completed, the TLM representation may be used as a specification to for a corresponding RTL source file.
· GDSII representation required only for chip production.

Design tools are integrated in IDE, shown in Fig.3.

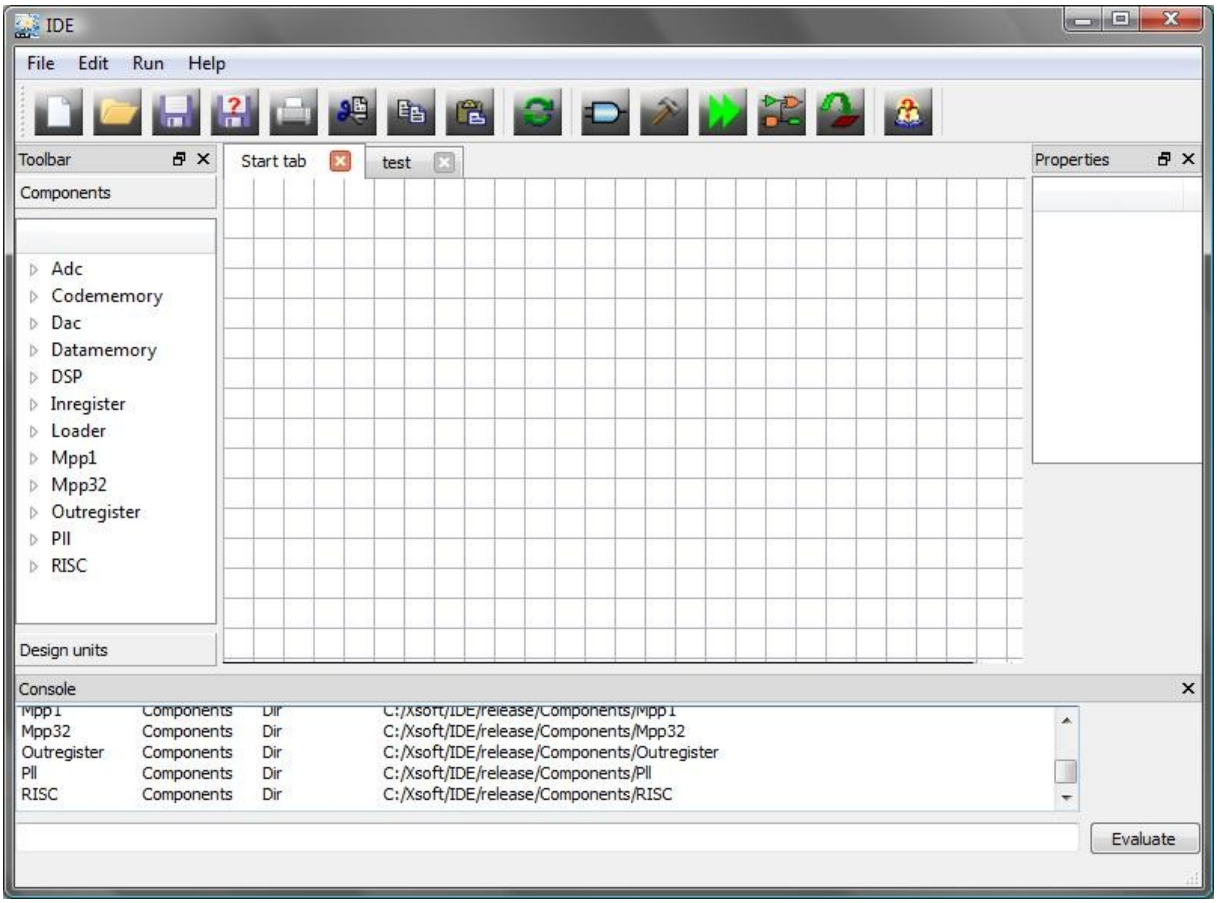

Fig. 3. The Quark CAD design tool

The main window, denoted as 'start tab', shows the content of the Quark VM virtual screen. Any other tabs are text editors from the Qt library, used for viewing and editing source texts and model reports. The 'Evaluate' button at the bottom of the window runs evaluation of the text, entered in the text field near it. The content of this text field is copied directly into the Forth terminal input buffer, when the VM interpreter runs.

Quark CAD stack processor cores

Under the contract conditions, several processor cores must be provided as part of a standard component library of the CAD tool. There are a general-purpose RISC processor, a DSP core, and a matrix of processor elements in this library. All processor cores may be connected to the synchronous system bus for sharing peripheral devices. Fixed priority arbitration is used in

case of more than one processor in the system. Stack operations and a minimal set of Forth words are supported by all cores being developed.

The RISC-processor, named QuarkR, is a 32-bit general-purpose processor. Its main features are listed below:
· Harvard architecture with a 3-stage pipeline
· 32 general-purpose registers
· 3-address and stack-based register file access, when the top of data stack is initially located in R31
· Independent return stack (32 cells deep), switchable to a stack in the external data memory
· 32-bit wide command
· Base Forth commands implemented in hardware as a command set extension (no mode switching is required)

DSP:
· Independently running MAC engines, 1 or 2 blocks in each DSP core. Each DSP block may work as one 32x32, two 32x16, or four 16x16 independent multipliers with corresponding accumulators
· Control processor unit, 32-bit stack core with 8-bit wide commands; command set is similar to that of QuarkR
Matrix processors have two modifications, one of them with a grid of 32-bit compact stack cores:
· 4 cells deep data stack, 4 cells deep return stack
· Harvard architecture with separate code and data memories for each core
· Up to 32x32 processor unit grid
· Local links, global column buses and system bus interface

Conclusions

From project description above, we can make several assumptions about usage style of integrated development systems based on Forth-engine. They have found preliminary acknowledgement while QuarkCAD was in service at developers laboratories.

- The combined interpreting&compiling nature of Forth makes very useful script-rich style of large, complex system assembling. In fact, C++ code has about nothing influence on the actually characteristic of CAD output, while all of results depends only on scripts being run. This allow a wide range of IP cores to be modeled.
- Possibility of just-in-time defining of additional commands allow deep integration of different CAD flow steps, including flexible and re-engineering interfaces to external tools. For example, a full path to VHDL code generation provided by one of Forth script, when another, added later, scripts are interacts with FPGA downloading tools.
- Working on the scripts, developers team found a little need of complex programming technologies, some of them was added to Forth last time. Nevertheless, Quark VM, which closer to F83 than to F94, provide enough capabilities to solve a hard enough task, such as system-level chip design. We relied less on the built-in possibilities of the Forth, than on added by our scripts. Thus, in the course of project performance even base possibilities of the Forth have allowed to achieve good results.

References
1. C. Rowen. Engineering the Complex SoC
2. The International Technology Roadmap for Semiconductors: 2007 http://public.itrs.net