Integration of Large-Scale FPGA and DRAM in a Package Using Chip-on-Chip Technology

Michael X. Wang, Katsuharu Suzuki,
Wayne W.-M. Dai

Department of Computer Engineering
University of California
Santa Cruz, CA 95064
Tel: 831-459-4954
Fax: 831-459-4829
email: *mwang, suzuki, dai*@ce.ucsc.edu

Yee L. Low, Kevin J. O'conner,
King L. Tai

Bell laboratories
Lucent Technologies
Murray Hill, NJ 07974
Tel: 908-582-2718
Fax: 908-5823834
email: *yll, kjo, kltai*@lucent.com

Abstract— A field-programmable multi-chip module containing one ORCA 3T/125 FPGA and 4 MByte DRAM was built using chip-on-chip technology. Module architecture and physical design issues are presented. A PCI board consisting of four chip-on-chip modules is also built as the test vehivle. The design environment for this multi-chip module, including visual or C++ design entry and bit-serial datapath synthesis system, is also discussed. Some ongoing approaches, like double-flip technology and area IO are also addressed.

I. Introduction

Since Dobbleare and El Gamal first proposed applying Multi-chip Module (MCM) technology to FPGAs by adding additional switchboxes and special drivers to the IO structures of standard FPGAs [DGHK92], a lot of effort has been put on the development of Field-Programmable Multi-chip Module (FPMCM) by several research groups [DGHK92][Ter95][Gar95]. During the last six years, we have designed, fabricated, assembled, and demonstrated two generations of silicon-on-silicon field-programmable multi-chip modules (FPMCMs) including the latest one with 200K gates.

FPMCM-I [DGI+94] was the first field-programmable multi-chip module with silicon substrate. Twelve Xilinx 3042 FPGAs and one Aptix FPIC were flipped on a 30.6 mm square silicon substrate. In total, 44K gates FPGAs were built in a star architecture. FPMCM-II [DD96] includes 4 Altera 10K50 FPGAs. We eliminated the central FPIC and put all interconnections on the substrate. This 200K-gate multi-chip module was the biggest field-programmable logic device at that time.

As the capacity of the FPGA devices grows very rapidly, with the latest one having 500K logic equivalent gates, the band-width between the FPMCM and memory devices has become the bottleneck of the system. However, memory fabrication technology and logic fabrication technology diverge even both are in CMOS. Embedded DRAM would cost more to fabricate than discrete DRAM. Unlike System- On-Chip (SOC) approach, which compromises individual chip technologies, System-In-Package (SIP) approach overcomes formidable integration barriers by the integration of conventional ASIC and memory technologies using existing, individually optimized ICs. Therefore, memory and logic can be integrated at lower cost and reduced size. Based on this concept, we built a new generation of FPMCM, FPMCM III, which integrates large-scale FPGA and multi-banked DRAM in a package to provide high band-width for memory access. FPMCM III is the first multi-chip module integrating FPGA and DRAM using chip-on-chip technology. At the same time, we have developed and demonstrated a high-performance, bit-serial synthesis system for large scale adaptive computing systems using the FPMCMs.

Section 2 introduces our innovative chip-on-chip module, which includes one FPGA and four SGRAM chips. Section 3 describes the prototyping board for this module, its architecture and physical design issues. Section 4 describes our design environment and compilation tool for the multi-chip module. Section 5 concludes the paper with some remarks.

II. Chip-On-Chip Module

To meet the demand for a large amount of memory in field-programmable systems, an innovative multi-chip module, FPMCM III, was built using chip-on-chip technology.

A. Module Architecture

Figure 1 illustrates the architecture of the chip-on-chip module. Each module consists of one ORCA 3T125 FPGA (186K equivalent gates) and four 1MB Micron

SGRAM chips. Before we designed the connection between FPGA and SGRAMs, we studied the implementation of the internal memory controller. There are three possible ways to connect the FPGA and memory: (1) Connect each SGRAM to the FPGA, which requires four internal memory controllers; (2) reorganize SGRAMs into two banks, which requires two memory controllers and some long wires inside the FPGA; (3) group all SGRAMs into one bank, which requires only one memory controller, but a large number of global interconnections. We examined the area efficiency of each implementation. Although the independent memory controller approach consumes a little more chip area, simplifies the memory access and reduces the global interconnection inside the FPGA.

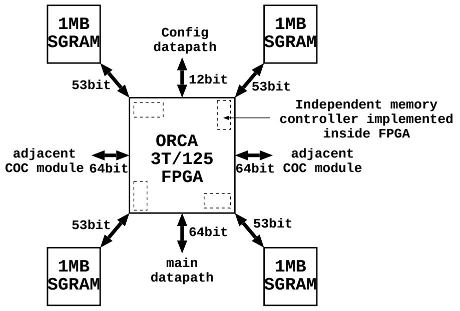

Fig. 1. Chip-on-chip (COC) module architecture. There are one ORCA 3T/125 FPGA and four SGRAM chips in the module. The SGRAMs have been flip-chip mounted on the FPGA. The FPGA can access the four SGRAMs independently and simultaneously. Four pre-designed memory controllers can be downloaded to the FPGA when the modules are powered up. Each module has one 64-bit bus to the main datapath and 2 64-bit buses to adjacent COC modules.

The SGRAMs are connected to the FPGA in a star architecture with a 32-bit datapath. There are two 64-bit buses connected to adjacent chip-on-chip modules. This architecture can easily achieve parallel and pipeline computation. In addition, a 64-bit bus is connected to the main datapath, providing high bandwidth communication between the chip-on-chip module and the system controller.

Figure 2 illustrates the pin assignment of the module, where pins of the FPGA have been divided into eight groups, and pins in each group match half of the pins of one SGRAM.

B. Physical Design

The chip-on-chip technology was developed at Bell Labs, Lucent Technologies [LFO97], initially for a three-chip module consisting of one AT&T DSP-1610 digital signal processor chip and two 256K SRAM chips[TFH+95].

Figure 3 illustrates chip-on-chip technology. Surface chips (Micron 1MB SGRAM) are flipped on the substrate chip (ORCA 3T125 FPGA) with evaporated solder bumps. IO redistribution layers are added on the surface

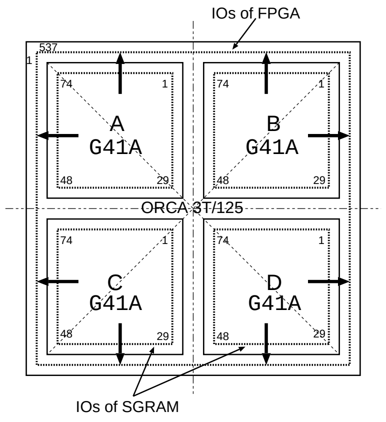

Fig. 2. Pins of FPGA have been divided into eight groups, and the pins in each group match half of the pins of one SGRAM chip.

of each chip. Three additional layers, two on the FPGA and one on the SGRAM, are used to route the connections. The IO redistribution was made on the SGRAM, as illustrated in Figure 4(b). Eighty-four pads of the SGRAM are selected to connect to the solder bumps. To provide more mechanical strength and thermal contact, nine extra solder bumps were built between the FPGA and SGRAM.

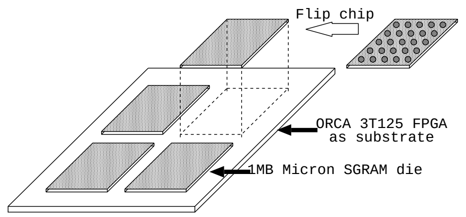

Fig. 3. The FPMCM III is formed by chip-on-chip technology. Four SGRAM dies are flipped on a 3T125 FPGA to achieve high bandwidth and high density.

Figure 4(a) is the layout of additional layers on the FPGA. Two mental layers were used to route the solder joints to the pads of FPGA. The memory chips are treated as surface components whereas the logic chip is considered as a virtual substrate. Its bondpads are each treated as a single-pin bottom surface component. With this approach, we are able to route over 95% of the connections with the automatic router and the rest are easily routed manually. The routing result illustrates that the connections are well distributed on the FPGA dies, which means the selection of module architecture and pin assignment are good.

The thickness of the SGRAM die is about 12.0 mils, the diameter of the bump is about 8 mils, and the thickness of all other added layers is less than 1 mil. So the height of the module is about 21 mils more than the original FPGA die. It is possible for us to put this chip-on-chip

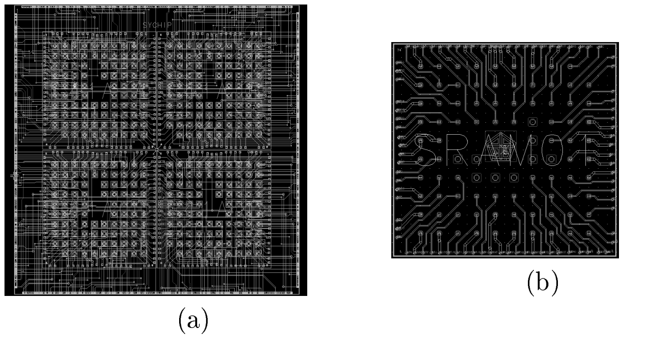

Fig. 4. Module layout (a) and IO re-distribution on the SGRAM side (b).

module into standard package to reduce the engineering cost. We selected a 600-pin EBGA package to maximize the number of user IOs. Table.I gives some features of the FPMCM III module. There are 452 user IOs, 212 of which are reserved for the internal SGRAMs. The estimated speed of this module is 100MHz.

	FPMCM III
Total chips	5
System gates	186K
Memory size	4MByte
User IO	452
Substrate area	144 sq mm (FPGA die size)
Package	600-pin EBGA
Speed	100MHz

TABLE I
FEATURES OF FPMCM III MODULE

C. Further Improvement

Packaging limits the chip-on-chip module in many ways, like cost-effectiveness, area-efficiency, and heat dissipation. The cost of custom-designed packages is one of the major parts of engineering cost in the multi-chip module design process. When the chip is packaged, it might occupy an order of magnitude more board area than the bare die. For example, the ORCA 3T/125 bare die is 1.2cm x 1.2cm, while the package is 4.5cm x 4.5cm. This increase lowers the area-efficiency and causes some routing difficulties for the board design. This large package also prohibits the module from fitting into small cards, like the PCMCIA card. Package is also an obstacle for heat dissipation, which limits the clock frequency of the module, for its power consumption depends on the speed.

All these problems can be solved by introducing double-flip technology, which is illustrated in Figure 5. In our case, the memory chips are flipped on the substrate chip with flip-chip technology. Then the module will be flipped on the board without packaging. This technology greatly reduces the area occupied by the chip and significantly improve the speed when the heat sinks are attached to the dies.

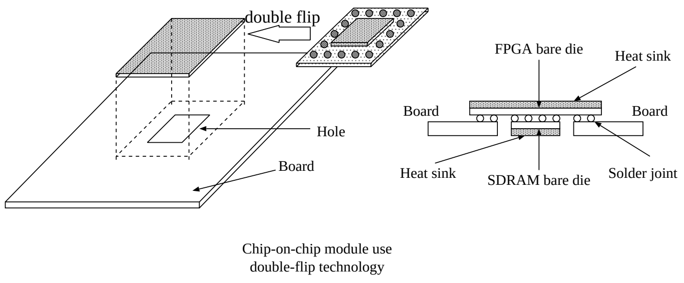

Fig. 5. The chip-on-chip module is flipped on the board without packaging. Solder joints are used to make the contact between the substrate chip and the board. Heat sink materials are put on both sides of the module to enhance the module's thermal dissipation.

Flip-chip technology makes high bandwidth communication between logic chips and memory chips possible, but commercial FPGAs and memories are not designed for multi-chip module technology. There is a mismatch between the ability of perimeter bonds to provide IO and the propensity of logic to demand it.

In our multi-chip module, many IOs are not connected to external pins and are not exposed to the Electro-Static Discharge (ESD). These protections become redundant and degrade the performance.

Area IO[MDRD95] provides a way to eliminate the IO bottleneck of FPGA and reduce the work of IO redistribution in the MCM design process. FPGAs and DRAMs are naturally fit in the area IO architecture since their block or array structure. When the ESD protections are completely or partially removed, less power consumption, less chip area, and higher clock frequency could be anticipated.

III. PCI Board with COC Modules

To implement the chip-on-chip module in some real applications and to test its efficiency, a test vehicle has been built.

A. Board Architecture

We have designed and fabricated a PCI plug-in board which contains four chip-on-chip modules. The board architecture is illustrated in Figure 6. Four chip-on-chip modules are connected in a mesh architecture. Each module can talk to adjacent modules with a 64-bit bandwidth direct connection. All modules share a 64-bit bus to communicate with the system controller, a SUN microSPARC controller. This controller is also used as the PCI interface to a host computer. An ORCA 3T/30 FPGA is used as a configuration controller. It obtains the program from main datapath and sends to the COC module through the slave-parallel configuration path. The program for the microSPARC controller and ORCA 3T/30 are stored in two 4Mbit flash memory chips. A Pentium II based PC is used as the host computer to control the operation of the board.

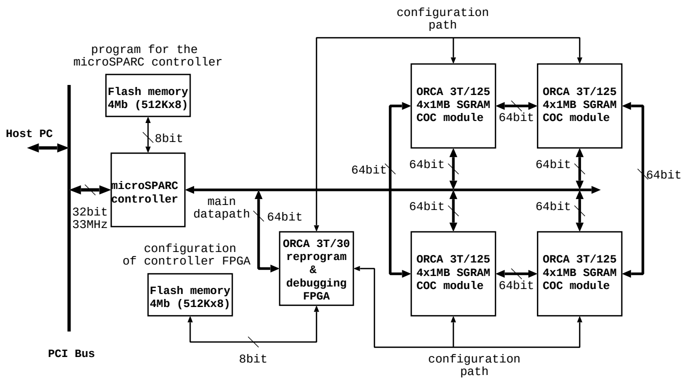
configuration path
program for the microSPARC controller
Flash memory 4Mb (512Kx8)
Host PC
8bit
microSPARC controller
32bit 33MHz
main datapath
64bit
configuration of controller FPGA
Flash memory 4Mb (512Kx8)
ORCA 3T/30 reprogram & debugging FPGA
ORCA 3T/125 4x1MB SGRAM COC module
64bit
PCI Bus
8bit
configuration path

Fig. 6. Board Architecture and datapath specification

Before we laid out this board, we studied the power consumption of each component on it. Table.II represents the worst-case power consumption of each part at the frequency of 100MHz. We find that the power dissipation is mainly in the multi-chip modules, about 6W each in the worst case. The microSPARC controller will consume up to 4W power. The total power consumption is more than the PCI bus can supply, which is about 25W in the 3.3V PCI environment. To put the system on the safe side, we decided to add an external power supply on the board, instead of using the PCI bus. This power connector can provide up to 40W, which will satisfy the power demand even for maximum utilization of the multi-chip modules.

Component	Quantity	Power Consumption
SGRAM	16	<800mW
3T/125	4	<2.8W
3T/30	1	<0.9W
microSPARC	1	<4.0W
Flash Memory	2	<0.06W
Total	–	29.02W

TABLE II
Power estimation for the FPMCM board.

A programmable phase lock loop (PLL) and a 50MHz oscillator were put on the board to achieve different system clock frequencies. When the PCI bus clock is selected to input to the PLL, the output clock frequency could be 33MHz or 66MHz, dependent on the program of the PLL. If the oscillator is selected to input to the PLL, the system could run at 100MHz.

B. Physical Design

Figures 7 and 8 illustrate the board layout and floorplan, respectively. Eight routing layers are used to route the connections. Though this board contains 3.3V components and 5V components, the 5V power supply does not occupy a full routing layer, but was designed to share routing layers with other signals for economic reason. Since this board is for prototyping, the size is not the main constraint. Right now the board is 11 inches by 6 inches, which can not fit into the case of regular PCs. The main reason for this huge size is that we put BGA sockets on the board instead of mounting the multi-chip modules on the board directly. The test ports also occupy some space and contribute to routing difficulty. We expect a board with the size of about 7 inch by 4.5 inch which can perfectly fit into the PC when all debugging components are removed.

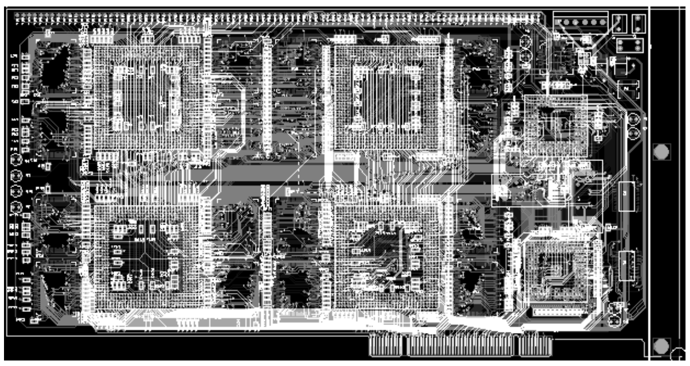

Fig. 7. Board Layout

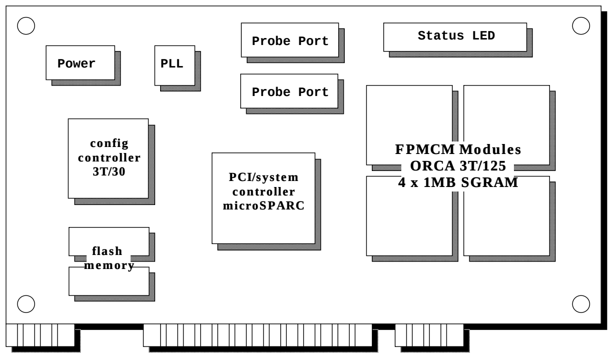

Fig. 8. Board Floorplan

IV. Design Environment for FPMCM-III

We have developed the design environment to enable the users to implement their applications on FPMCM-III in a simple and fast way. The design environment is equipped with features including visual or C++ design entry and bit-serial datapath simulation and synthesis system. Figure 9 shows the design flow in our design environment.

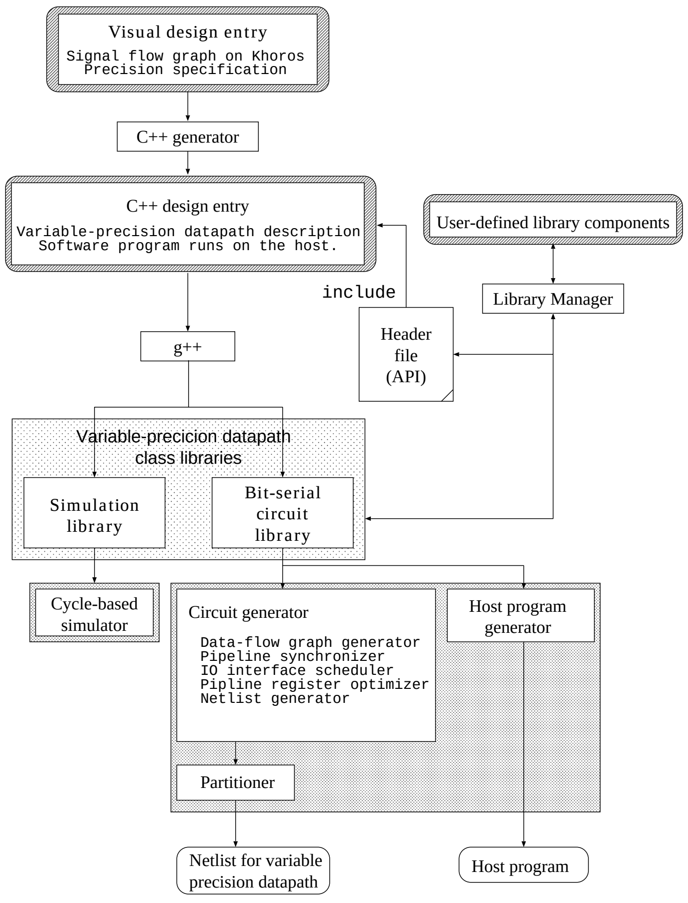
Visual design entry
Signal flow graph on Khoros
Precision specification
C++ generator
C++ design entry
Variable-precision datapath description
Software program runs on the host.
User-defined library components
include
Library Manager
Header file (API)
g++
Variable-precicion datapath class libraries
Simulation library
Bit-serial circuit library
Cycle-based simulator
Circuit generator
Data-flow graph generator
Pipeline synchronizer
IO interface scheduler
Pipline register optimizer
Netlist generator
Host program generator
Partitioner
Netlist for variable precision datapath
Host program

Fig. 9. Hardware software co-design flow in the FPMCM design environment.

A. Visual or C++ Design Entry

The users can specify their design either as a signal flow graph using visual programming interface or as a C++ description with the provided classes. We expanded a visual programming tool, Cantata[YAK95], to support hardware software co-design. Cantata is a tool integrated in Khoros[KR94] which is a common design environment among image processing system designers. The users can describe their design in a form of signal flow graph by selecting glyphs from our libraries and connecting them. This tool generates a C++ description of the whole system including hardware, software, and interfaces from a given signal flow graph.

The C++ description specifies the behavior of reconfigurable hardware on FPMCM, data transfers between FPMCM and host, and software program for the host. The core of the C++ program as a design entry is the hardware description. For hardware design, we provide class libraries, such as a fixed-point class with bit-precision specification. Figure 10 shows an example of a C++ description of a datapath to be implemented on the FPMCM. In the example in Figure 10, the fixed-point class, called `fix`, is used to describe variables.

```
void datapth() {
  vector<fix> inpix(9, fix(8));
  // bitwidth of inpix is 8 bit,
  vector<fix> sum(9, fix(32));
  // bitwidth of sum is 32 bit,
  vector<fix> tmp(9, fix(33));
  // bitwidth of tmp is 33 bit,

  sum[0] = inpix[0] * coeff[0];
  for (int i = 1; i < 9; i++) {
    tmp[i-1] = inpix[i] * coeff[i];
    tmp[i-1] = inpix[i];
    sum[i] = sum[i-1] + tmp[i-1];
   }
}
```

Fig. 10. An example of C++ design entry

The fixed-point class, `fix` supports the variable-precision computation. The bit-width of each variable is specified as an argument of constructor of `fix`. The functions of the hardware are described as overloaded operations such as "+" or "*". With this operation overloading feature of C++, the users can describe their design in the same way used in algorithm description. Because C++ is an object-oriented language, the implementation of those classes are hidden, and the designers only need to include header files into their design and link their programs with the provided libraries. These libraries are also managed by the library manager so that the designers can add new classes and components.

B. Bit-serial Datapath Simulation and Synthesis System

An application program in C++ is compiled and linked to the provided class library. Each class has a simulation kernel and a circuit-generation kernel as the implementation of the class. The application program is compiled into a simulator or a circuit generator by linking to the simulation kernel or the circuit-generation kernel, respectively.

The simulation kernel enables the simulation of a given design. We have implemented a cycle-based simulation kernel for `fix` class. With this feature, the designer can simulate their design at an algorithm level rather than a gate level or lower levels, resulting in faster simulations.

The circuit-generation kernel synthesizes a bit-serial datapath by mapping the data flow in the design to bit-serial datapath on FPGA. We have investigated the advantage of bit-serial architecture on the reconfigurable hardware[ID95]. Bit-serial architectures has significant advantages over bit-parallel architecture in the following aspects: (1) high clock frequency achieved by the reduction of routing delay, (2) high throughput, (3) ease of placement and routing, (4) scalability in routing resource requirement, and (5) suitable to computation in variable precision. The circuit generator synthesizes the signal flow graph onto efficient bit-serial datapath by mapping the operations in data flow to manually-optimized bit-serial components and connecting all of the components according to data flow in the user-specified behavioral description.

After the circuit generation, the synthesized bit-serial circuit is partitioned to fit in our FPMCM architecture. Because of locality of connection in bit-serial circuits, they do not require a complex algorithm for the partitioning. We employ a recursive 2-way partition algorithm[Iss96]. The generated netlists are placed, routed, and downloaded by the Lucent design tool[Inc99]. The configuration and execution process is controlled by the host program.

V. Summary

Integration of field-programmable logic devices and DRAM is a challenging problem in VLSI design. System-In-Package (SIP) provides a cost-effective solution for this problem. Our FPMCM III module demonstrates chip-on-chip packaging technology and leads to a solution for large scale embedded memory integration. The design environment based on bit-serial synthesis system could enhance the performance of adaptive computing systems using FPMCMs. In order to improve the performance of FPMCMs further, double-flip technology and custom designed FPGA and DRAM with area IO should be addressed.

VI. Acknowledgments

The authors would like to thank Dr. Tsuyoshi Isshiki of Tokyo Institute of Technology for his suggestion on FPMCM III board design.

References

[DD96] J. Darnauer and Wayne W.M. Dai. Tradeoffs in chip and substrate complexity and cost for field programmable multichip modules. In *Eighth Annual IEEE International Conference on Innovative Systems in Silicon*, pages 217–27, 1996.

[DGHK92] I. Dobbelare, A. El Gamal, D. How, and B. Kleveland. Field programmable mcm systems- design of an interconnection frame. In *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pages 15.1/1–6, 1992.

[DGI+94] J. Darnauer, P. Garay, T. Isshiki, J. Ramirez, and W. W. H. Dai. A field programmable multi-chip module (fpmcm). In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 1–10, 1994.

[Gar95] Tim Garverick. Field configurable mcm. In *Proceedings of the ARPA eletronic Packaging and Interconnect Meeting*, 1995.

[ID95] Tsuyoshi Isshiki and Wayne Wei-Ming Dai. High-level bit-serial data path synthesis for multi-fpga systems. In *Proc. of International Symposim on Field Programmable Gate Arrays*, 1995.

[Inc99] Lucent Technologies Inc. ORCA foundry develop system, April 1999.

[Iss96] Tsuyoshi Isshiki. *High-Performance Bit-serial Datapath Implementation for Large-Scale Configurable Systems.* PhD thesis, University of California at Santa Cruz, 1996.

[KR94] K. Konstantinides and J.R. Rasure. The khoros software development environment for image and signal processing. *IEEE Transactions on Image Processing*, 3(3):243–252, May 1994.

[LFO97] Y.L Low, R.C Frye, and K.J O'Conner. Design methodology for chip-on-chip applications. In *Electrical Performance of electronic Packaging*, pages 5–8, 1997.

[MDRD95] V. Maheshwari, J. Darnauer, J. Ramirez, and W.M. W. Dai. Design of fpgas with area i/o for field programmable mcm. In *1995 Symposium on Field Programmable Gate Arrays*, 1995.

[Ter95] Richard Terril. A novel 2-sided multi-chip module used to creat a 50,000 gate programmable logic device. In *IEEE Multi-Chip Module Conference*, 1995.

[TFH+95] K.L. Tai, R.C. Frye, B.J. Han, M.Y. Lau, and D. Kossives. A chip-on-chip dsp/sram multichip module. In *International Conference on Multichip Modules*, pages 466–71, 1995.

[YAK95] M. Young, D. Argiro, and S. Kubica. Cantata: visual programming environment for the khoros system. *Computer Graphics*, 29(2):22–24, May 1995.