USING 3D INTEGRATION TECHNOLOGY TO REALIZE MULTI-CONTEXT FPGAS

Alesandro Cevrero[1], Panagiotis Athanasopoulos[1] Hadi Parandeh-Afshar[2], Maurizio Skerlj[2], Philip Brisk[2], Yusuf Leblebici[1], Paolo Ienne[2],

[1]School of Engineering [2]School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland CH-1015
email: {first_name.last_name}@epfl.ch

ABSTRACT

This paper advocates the use of 3D integration technology to stack a DRAM on top of an FPGA. The DRAM will store future FPGA contexts. A configuration is read from the DRAM into a latch array on the DRAM layer while the FPGA executes; the new configuration is loaded from the latch array into the FPGA in 60ns (5 cycles). The latency between reconfigurations, 8.42μs, is dominated by the time to read data from the DRAM into the latch array. We estimate that the DRAM can cache 289 FPGA contexts.

1. INTRODUCTION

Reconfigurable computing is a paradigm in which a hardware platform dynamically adapts itself to an application's behavior. Historically, the latency of dynamic reconfiguration has been prohibitive; nonetheless, interest in techniques to speed up the reconfiguration of FPGAs remains. *Multi-context FPGAs* store several configurations on chip, switching between them at runtime [3, 9, 10, 16-18]; however, this reduces logic density and increases power consumption and wirelength.

To address this concern, we advocate the use of 3D integration technology to store FPGA contexts in an on-chip DRAM. The proposed system is an island-style FPGA with a configuration DRAM stacked using *"face-to-face" (f2f)* metallic bonding [11], in which the tops of two adjacent tiers are stacked facing one another. F2f bonding provides the greatest layer-to-layer via density as the inter-layer vias do not punch through active silicon.

2. DRAM ARCHITECTURE

Fig. 1 shows a digitline DRAM architecture [7, Ch. 2]. The sense amplifier is a collection of circuit elements pitch-matched to the digitline (bitline) of a DRAM array; they amplify and store the value of an row when selected. A subset of the digitlines is connected to the I/O lines using an I/O transistor embedded in the sense amps and driven by the column decoder. The select I/O lines are amplified and latched using a *Helper Flip Flop (HFF)* [7].

Fig. 2 shows a DRAM array. At the intersection between the wordline and column select (CSEL) line, three bits of data can be written to the HFF, but only one block can write to the HFF at a time. The HFF latches the bits selected by the column decoder. We want to extract as many bits in parallel as possible to facilitate fast reconfiguration; the signals are first latched into the HFF because the sense amps do not restore the signals to their full level. We extract the data after it has been stored in the HFF via the *global I/O (GIO)* lines. A subset of row bits selected by the column decoder is read into the HFF; subsequent reads to this row can access the sense amps directly. As reading a row discharges the capacitors in the array, there is a 10-12ns latency to close the row and write the data back before the next row can be read.

DRAM timing characteristics are provided by Park et al. [13] for an 8-bank 512MB DDR3. We refer to data extracted by *Semiconductor Insight* for the 1Gbit DDR3 die manufactured by *Micron Technologies* in a 78nm DRAM process for the area of the cell array and peripheral circuitry [15]. Table 1 lists additional information related to a complete 512MB DDR3 chip, and Table 2 summarizes the key parameters of a 32MB array block of one 512MB DDR3 chip. In Table 1, the minimum column cycle time is the time taken to read data from the sense amp, restore it in the HFF, and read it on the GIO lines; the row precharge time includes the time to store data in a sense amp in preparation for a read/write operation; and the RAS to CAS delay is the latency before a new row can be activated. Table 3 summarizes the parameters of the 2D FPGA on which we based our design.

Each DRAM bank is a standalone 64MB memory with 128 I/O lines and a row size of 2KB; it caches configurations for several FPGA tiles, as the bank area far exceeds the tile area. One configuration bit per 3D interconnect is moved from the FPGA layer to the DRAM layer to equalize the area. When a row is opened, the data is read into the row buffer; the column address is incremented and 128 bits of data are read, each taking 4.2ns, until the row is finished. A latch array located inside the DRAM layer buffers the data as it is read; a multiplexor placed before each 3D interconnect selects a latch during each reconfiguration cycle.

Cu-Cu bonding connects the two tiers; power and I/O signals are connected to the package through wire bonding. The DRAM is configured through a serial interface. The I/O and DRAM power signals also connect to the FPGA bonding pads. We assume the FPGA is fabricated with a 90nm CMOS process and the DRAM with a 78nm process using dual gate triple metal technology [15]. We estimate that two additional metal layers in the FPGA will route signals from the DRAM layer to their locations in the FPGA.

3. ARCHITECTURAL VALIDATION

We wrote a VHDL description of the FPGA logic tile described in the preceding section; we synthesized the design with *Synopsys Design Compiler 2007.12* and place-and-routed it with *Cadence SoC Encounter 6.2*. The architecture was implemented with a 90nm *UMC* low-k logic/MS/RF process. Table 4 lists the architectural parameters that characterize the FPGA and DRAM.

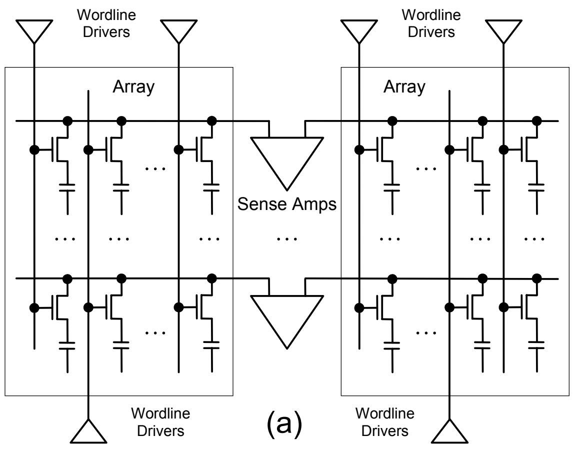

Fig. 1. Open digitline array scheme [7, Fig. 2.6].

Table 1. 32 MB DRAM array block data

Array Block Size (Half bank)	32MB
Process	78nm
Cell Structure	6F^2
Cell Size	0.0365µm
Area Efficiency	74%
I/O Sense Amp Area Overhead	15%
GIO Data Lines	64
Min Column Cycle Time (Tccd)	4.2ns
Row Precharge Time (Trp)	12ns
RAS to CAS delay (Trcd)	12ns

Table 2. 512 MB DDR3 chip data

Bank Size	64MB
Row Size	2KB
GIO Data Lines	128

Table 3. 2D FPGA tile parameters

Logic cells per cluster	8
Intra-cluster routing	Actel-style IIB [5]
X-channel width	128
Y-channel width	128
LABs spanned per wire	4
Type of routing architecture	Direct drive
Configuration bits per tile	1,488
Switch block type	Disjoint

The latch array area is

$$A_{latch\_array} = A_{latch} \times Tile_{num} \times T_{conf} . \quad (1)$$

V_{bank} is the number of vertical interconnects from each bank to the FPGA. We partition the vertical connections within a bank uniformly across the tiles it serves, i.e.,

$$V_{bank} = V_{conf} \times Tile_{num} . \quad (2)$$

Buf_{size} is the area of the buffer containing configuration bits that have been moved from the FPGA layer to the DRAM layer.

$$Buf_{size} = V_{bank} \times A_{latch} . \quad (3)$$

The tile area has a linear correlation to the number of latches that have been moved to the DRAM. A_{tile} can be approximated as

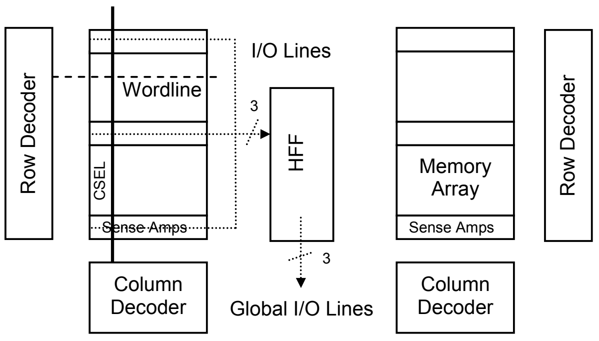

Fig. 2. Array block diagram (based on [7, Figs. 8.5-8.7])

Table 4. DRAM and FPGA architectural parameters

A_{latch}	Latch area
A_{latch_array}	Latch array area
A_{bank}	Bank area
A_{tile}	Tile area
$Tile_{num}$	Tiles per DRAM bank
$Tile_{conf}$	Configuration bits per tile
Buf_{size}	Configuration buffer area
V_{bank}	Vertical connections per bank
V_{conf}	Vertical connections per tile
L_{config}	Reconfiguration latency
$L_{DRAM\text{-}to\text{-}latch}$	DRAM-to-latch data transfer latency
L_{row}	Latency to read a row into the latch array

$$A_{tile} \cong (25{,}600 + 3.46 \times T_{conf})\ \mu m^2 . \quad (4)$$

If there are fewer 3D interconnects than configuration bits, the reconfiguration latency is

$$L_{reconfig} = \left\lceil \frac{Tile_{conf}}{V_{conf}} \right\rceil + 1 . \quad (5)$$

The +1 term accounts of an extra cycle to write data to the buffer.

A grid of Cu bonding pads connects the two layers. Placement and routing experiments (Section 4) determined the number of 3D connections, yielding V_{conf} = 298 and A_{tile} = 29,717µm^2. From the data in Table 1, we estimate that $A_{bank} \approx$ 3,900µm^2, and from our standard cell library, we estimate that $A_{latch} \approx$ 3.14µm^2.

The goal is to balance the area of the two layers such that

$$A_{latch\_array} + A_{bank} + Buf_{size} = Tile_{num} \times A_{tile} . \quad (6)$$

Substituting Eqs. (1) and (3) into Eq. (6) and solving for $Tile_{num}$ yields

$$Tile_{num} = \left\lceil \frac{A_{bank}}{A_{tile} - A_{latch} \times Tile_{conf} - A_{latch} \times V_{conf}} \right\rceil , \quad (7)$$

the number of tile configurations cached by one DRAM bank.

Next, we estimate L_{config} and $L_{DRAM\text{-}to\text{-}latch}$. We assume Cu pad dimensions of 5µm × 5µm with 5µm spacing, yielding a pitch of 10µm [12]. Such a system could be fabricated using existing f2f bonding with a high yield. From Eqs. (6) and (7), we determine that L_{config} = 6 cycles and $Tile_{num}$ = 156. Each DRAM caches data for 156 tiles, i.e., 2,323,128 configuration bits. Since each row contains 2KB of data, 15 row accesses are required. We estimate L_{row}, the latency to move data from each row into the latch array, to be 561.6ns; repeating this operation 15 times yields $L_{DRAM\text{-}to\text{-}latch}$ = 8.42µs.

4. PLACE-AND-ROUTE EXPERIMENTS

This section verifies that our proposed design is routable in the nine metal layers provided by our process technology. We synthesized the design with *Synopsys Design Compiler 2007.12* and place-and-routed it with *Cadence SoC Encounter 6.2*. We introduced a new standard cell with dimensions of 1.120μm × 2.800μm, representing an SRAM cell that holds configuration bits. With this cell, we estimate that the tile area is 30.751μm^2. Each configuration bit moved to the DRAM layer is replaced with a Cu pad on metal-10 to contact a vertical interconnect; specialized metal-9 contacts connect to each Cu pad. We varied the metal-9 contact area from 1.0μm^2 to 6.25μm^2. Table 5 shows that routability is achieved for contact areas of up to 4.00μm^2.

Morrow et al.'s [12] Cu bonding process can fabricate Cu bonds with dimensions of 5μm × 5μm with a pitch smaller than 10μm, as shown in Fig. 3. We can calculate the area of the tile width in comparison to pitch and copper bond size:

$$pitch\_width = bond\_width + spacing\_width \quad (8)$$
$$tile\_width = 3 \times bond\_width + 2 \times spacing\_width \quad (9)$$
$$tile\_height = 3 \times bond\_width + 2 \times spacing\_width \quad (10)$$
$$A_{tile} = tile\_width \times tile\_height \quad (11)$$

We make the following simplifying assumptions:

$$bond\_width = bond\_height \quad (12)$$
$$spacing\_width = spacing\_height \quad (13)$$
$$num\_horizontal\_bonds = num\_vertical\_bonds \quad (14)$$

From these assumptions, we can replace Eq. (11) with

$$A_{tile} = [(bond\_width) \times (num\_horiztonal\_bonds) + (spacing\_width) \times (num\_horizontal\_bonds-1)]^2. \quad (15)$$

From Fig. 3, *bond_width* = *spacing_width* = 5μm [12]; placing this value into Eq. (15) and factoring yields:

$$A_{tile} = [(100 \times num\_horizontal\_bonds^2) + 25]\ \mu m^2. \quad (16)$$

To find the maximum number of bonds that permit the Cu bond array to fit into the tile area of 25,600 μm^2, we solve Eq. (16) for $num\_horizontal\_bonds^2$, and round up to the nearest integer; this yields 255 bonds^2, which, we approximate as $256 = 16^2$.

A 16 × 16 grid of Cu bonds moves 256 configuration bits per tile to the DRAM layer, leaving 1,232 bits per tile on the FPGA layer. From Eq. (4), the expected tile area is 29,864μm^2. After placement around routing, the measured tile area with a core utilization of 95.2% was 29,901μm^2, close to this approximation.

Equalizing Eqs. (4) and (16) matches the areas of the Cu bond array area and the tile area. First, observe that

$$tile_{conf} = 1{,}488 - num\_horizontal\_bonds. \quad (17)$$

Substituting Eq. (15) into Eq. (4) yields

$$A_{tile} \cong 30{,}748 - (3.46 \times num\_horizontal\_bonds)\ \mu m^2. \quad (18)$$

Next, we substitute Eq. (16) for A_{tile} in Eq. (18). This yields $num\_horizontal\_bonds^2 \cong 325$; since $18^2 = 324$. Removing the ceiling function from Eq. (5) yields 4.59, which is rounded up to 5 cycles for reconfiguration. We could reduce the number of bonds to 298, while still maintaining a 5 cycle reconfiguration since 298 × 5 = 1490 ≥ 1,488 configuration bits. We use a 17 × 18 (= 306) grid and remove 8 Cu bonds from the array. This reduces the tile area to 29,717μm^2 and the area of the Cu bond grid to 28,875μm^2.

Table 5. Core utilization as a function of metal-9 contact ares when all 1,488 configuraiton bits are moved to the DRAM layer

Dimension	Area	Core Utilization
1.0μm × 1.0μm	1.00μm^2	93.50%
1.5μm × 1.5μm	2.25μm^2	90.30%
1.8μm × 1.8μm	3.24μm^2	50.50%*
2.0μm × 2.0μm	4.00μm^2	30.70%*
2.5μm × 2.5μm	6.25μm^2	Not Routable

* Some unresolved DRC errors may persist, even though they can be considered irrelevant to the presence of the contacts.

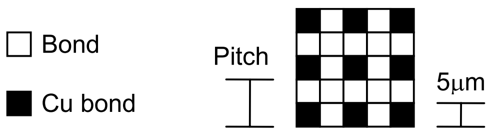

Fig. 3. Cu bond placement and layout

5. CASE STUDIES

We evaluated the multi-context FPGA on an MP3 audio decoder [4, 6] and an MPEG-4 video decoder [19]. The applications are decomposed into a pipeline of compute-intensive kernels. We assume that the size of the FPGA is equal to that of the largest kernel, denoted by *N*, and that the area of the DRAM layer is proportional to the number of cached configurations, *k*. The total FPGA area, including both tiers, is therefore

$$A_{3D\text{-}FPGA} = N \times A_{tile} \times (1 + k / 289). \quad (19)$$

Although we account for data stored in BlockRAMs, we assume that no configuration data is cached on top of them.

Each application was evaluated under three scenarios:

No Dynamic Reconfiguration (NDRC). One large FPGA without dynamic reconfiguration or configuration caching.

Off-chip Dynamic Reconfiguration (ODRC). The FPGA area is *N*; configurations are cached in an off-chip DRAM.

3D Configuration Caching (3DCC). The FPGA area is *N*; the 3D configuration caching scheme, as described earlier, is used.

The area requirements for ODRC and 3DCC are the same; the DRAM is off-chip in the case of ODRC and stacked on top of the FPGA in the case of 3DCC.

Both applications are soft real-time applications. For the MP3 decoder, we attempt to find the minimum frequency that can meet the constraint without dropping frames. As shown in Table 6 3DCC achieves the smallest frequency and area. ODRC and 3DCC require 99 tiles of FPGA area plus 3 tiles' worth of DRAM area to hold the 7 contexts, plus 5 BlockRAMs.

The MPEG-4 decoder is partitioned into hardware and software tasks. Three stages execute on the FPGA while six execute in software, which masks some of the reconfiguration overhead. The authors of the study report the area of each kernel in terms of logic gates [19]; as we have no way to equate logic gates with FPGA area utilization, we do not report the area of the different designs. Instead, we focus solely on performance issues.

Table 6. MP3 audio decoder with and without dynamic reconfiguration

Method	Min Freq.	Per-Reconfig. Delay	Area (tiles)	BlockRAMs
NDRC	2 MHz	0 ns	373	23
ODRC	4 MHz	2×10^6 ns	102	5
3DCC	2 MHz	60 ns	102	5

Table 7. MPEG-4 video decoder with and without dynamic reconfiguration

Method	Macroblock Delay	Frame Delay	Frame Rate
NDRC	20.42 μs	33 ms	30 fps
ODRC	6004 μs	9,500 ms	0.1 fps
3DCC	20.42 μs	33 ms	30 fps

Table 7 shows that 3DCC achieves the same macroblock and frame delays as NDRC; the reconfiguration delay is masked by the portions of the application that run in software. The performance of ODRC is prohibitive; it achieves an intolerable fps of 0.1.

6. RELATED WORK

The *Dharma* project [1] was an early multi-context PLD. 2D multi-context FPGAs have followed [3, 9, 10, 16-18]. These designs are all 2D, so the extra storage increases the planar area of each tile; this reduces logic density and increases wirelength, thereby degrading performance. Several of these proposals use DRAMs, instead of SRAMs to cache multiple contexts [10, 16].

Chiricescu et al. [2] built a 3D multi-context FPGA using an SRAM layer layer for configuration caching With three tiers, f2f bonding is impossible and through-silicon vias are required. Lin et al. [8] introduced a monolithically stacked 3D FPGA with separate layers for logic, routing, and configuration bits; unlike our work, their proposal was not a multi-context FPGA.

7. CONCLUSION

This paper lays the groundwork for the use of 3D integration technology to enable multi-context FPGAs. Each DRAM bank caches 289 configurations for 156 tiles; 6 cycles are required to reconfigure the device; and there is an 8.42μs latency between configurations. Our case studies show that improvements in area utilization are possible with minimal overhead. In the future, we plan to partition the DRAM layer between context storage and data storage and to increase the number of latch arrays in the DRAM layer to eliminate the 8.42μs latency between configurations. We also intend to analyze the power consumption of the DRAM layer and 3D interconnect structure, and to characterize the device from a thermal perspective, as DRAMs execution breaks down at lower temperatures than CMOS logic.

8. REFERENCES

[1] Bhat, N. B., Chaudhary, K., and Kuh, E. S. Performance-oriented fully routable dynamic architecture for a field programmable logic device, *Tech. Report UCB/ERL M93/42*, U.C. Berkeley, June, 1993.

[2] Chiricescu, S., et al. Design and analysis of a dynamically reconfigurable three-dimensional FPGA, *IEEE Trans. VLSI*, vol. 9, no. 1, February, 2001, 186-196.

[3] DeHon, A. DPGA-coupled microprocessors: commodity ICs for the early 21st century. *FPGAs for Custom Computing Machines (FCCM '94)* (Napa Valley, CA, USA, April 10-13, 1994) 31-39.

[4] Faltman, I., et al. A hardware implementation of an MP3 decoder. *Digital IC-Project, LTH, Sweden*, May, 2003. http://www.mp3-tech.org/programmer/docs/fpga_report.pdf

[5] Feng, W., and Kaptanoglu, S. Designing efficient input interconnect blocks for LUT clusters using counting and entropy. *ACM TRETS*, vol. 1, no. 1, March, 2008, article. No. 6.

[6] Hedberg, H., Lenart, T., and Svensson, H. A complete MP3 decoder on a chip. *IEEE Int. Conf. Microelectronics Systems Education (MSE '05)* (Anaheim, CA, USA, June 12-14, 2005) 103-104.

[7] Keeth, B., Baker, R. J., Johnson, B., and Lin, F. 2008. DRAM circuit design. Wiley-Interscience, Hoboken, NJ, USA.

[8] Lin, M., et al., A. Performance benefits of a monolithically stacked 3D FPGA, *IEEE Trans. CAD*, vol. 26, no. 2, Feb. 2007, 216-229.

[9] Lodi, A., et al. A VLIW processor with reconfigurable instruction set for embedded applications, *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, November, 2003, 1876-1886.

[10] Motomura, M., et al. An embedded DRAM-FPGA chip with instantaneous logic reconfiguration. *FPGAs for Custom Computing Machines (FCCM '98)* (Napa Valley, CA, USA, April 15-17, 1998) 264-266.

[11] Morrow, P. R., et al. Three dimensional wafer stacking via Cu-Cu bonding integrated with 65-nm strained-Si/low-k CMOS technology. *IEEE Electron Device Letters*, vol. 27, no. 5, May, 2006, 335-337.

[12] Morrow, P. R., et al. Wafer-level 3D interconnects via Cu bonding. *Advanced Metallization Conference (AMC '04)*, 2004, and *Mater. Res. Soc. Symp. Proc.*, vol. 20, 2005, 125-130.

[13] Park, C., et al. A 512-mb DDR3 SDRAM prototype with CIO minimization and self-calibration techniques. *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, April, 2006, 831-838.

[14] Patti, R. S. Three dimensional integrated circuits and the future of system-on-chip designs", *Proceedings of the IEEE*, vol. 94, no. 6, June, 2006, 1214-1224.

[15] Saino, K., et al. A novel W/WNx/dual-gate CMOS technology for future high-speed DRAM having enhanced retention time and reliability. *IEEE International Electron Devices Meeting Technical Digest*, December 8-10, 2003, 17.3.1-17.3-4.

[16] Shibata Y., et al. HOSMII: a virtual hardware integrated with DRAM. *Reconfigurable Architectures Workshop (RAW '98)* (Orlando, FL, UA, March 30, 1998) 85-90.

[17] Sugawara, T., Ide, K., and Sato, T. Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology. *IEICE Trans. Inf. Systems*, vol. E87-D, no. 8, August, 2004, 1997-2003.

[18] Trimberger, S., et al. A time-multiplexed FPGA. *IEEE Symp. FPGAs for Custom Computing Machines (FCCM '97)* (Napa Valley, CA, USA, April 16-18, 1997) 22-28.

[19] Zemva, A., and Verderber, M. FPGA-oriented HW/SW implementation of the MPEG-4 video decoder. *Microprocessors & Microsystems*, vol. 31, no. 5, August, 2007, 313-325.