FPGAs: A New Point in Database Design Space

Rene Mueller    Jens Teubner
{firstname.lastname}@inf.ethz.ch

Systems Group
Department of Computer Science, ETH Zurich, Switzerland
Heterogeneous Multi-Core Architectures

Computer architectures are undergoing major changes.

- single-core
- multi-core
- heterogeneous multi-core

We see an increasing **diversification** of compute resources:

- floating point units,
- graphics processors,
- Cell’s synergistic processing units,
- field-programmable gate arrays (FPGAs).
Field-Programmable Gate Arrays

Field-programmable gate arrays

- are configurable logic chips ("programmable hardware"),
- can be used to build hardware tailored for your application,
- provide parallelism, low latency, high throughput, and
- excel with power efficiency.

Today:

- FPGAs to assist database processing.
Outline

1. FPGA Basics
   - Technical Background

2. FPGA Design Techniques
   - Exploiting the Parallelism in FPGAs
   - Some More FPGA Features

3. FPGAs in Database Management Systems
   - System Integration

We’ve also prepared some live demos for you.

slides: http://people.inf.ethz.ch/muellren
Who we are

René Müller

- ETH Zurich, Systems Group
- Avalanche: database processing on FPGAs
- background: sensor networks, electrical engineering

Jens Teubner

- ETH Zurich, Systems Group
- Avalanche: database processing on FPGAs
- (Data Cyclotron: distributed databases on high-speed networks)
- background: databases, XQuery processing
FPGA Basics

What is an FPGA?
FPGA Architecture
FPGA — Field Programmable Gate Array

FPGAs
- are semiconductor devices
- can be configured to implement any digital circuit
- can be reconfigured in the field (vs. in the factory)
- are sold by two major vendors: Altera and Xilinx
FPGA—Custom Integrated Circuits

- FPGA intended to prototype Application-specific Integrated Circuits (ASICs).
- Today, used in Reconfigurable Computing.
- First FPGA XC2064 introduced by Xilinx Corp. in 1985.
- FPGA market $2.75 billion expected in 2010

FPGAs vs. ASICs

- FPGAs are more cost-effective
- Lower speed in FPGAs
- FPGAs are less power efficient
- FPGA circuits can be modified (at start time or even at runtime)
Reconfigurable Hardware

FPGA can be thought as two-layered devices:

- **Logic Layer**
  - consists of configurable logic blocks

- **Configuration Layer**
  - Memory that holds “programming” of the FPGA
  - Controls the function computed on the logic layer
  - Allows partial reconfiguration at runtime
Basic FPGA Architecture

- 2D chip layout
- Components
  - **CLB**: Configurable Logic Block implements logic function
  - **IOB**: Input/Output Block
  - **DCM**: Digital Clock Manager
- Interconnect Network
  - Short distance lines
  - Long distance lines
  - Clock lines
  - Switch boxes: programmable switches
Routing signals in an FPGA

programmable Switch Box and bundle of lines

programmable intersection point

programmable switch with memory cell

SRAM cell
Routing in a Xilinx Virtex-5 FPGA

Virtex-5 Routing

- More symmetric pattern, connecting CLBs
- More logic reached per hop
- Same pattern for all outputs

source: Steve Douglass et. al: *Virtex-5, the Next Generation 65 nm FPGA*. In HotChips 18. 2006.
Configurable Logic Block (CLB)

- Implementing \( \{0, 1\}^4 \rightarrow \{0, 1\} \) function
- Stores a single bit
- SRAM cell
- Multiplexer
- Clock
- D Flip Flop
Example—A simple 1-bit Counter

```c
bool accum_mod2(bool x) {
    static bool z = false;
    z = z ^ x;
    return z;
}
```

- 1-bit aggregate sum over a stream
- XOR implemented in LUT
- Two inputs remain unconnected
- z stored in flip flop
- Output feedback to input through interconnect
Structure of the Xilinx Virtex-6 FPGA

Modern FPGA are more complex (partial diagram of a CLB):

Additional High-level Primitives in Discrete Silicon

- Simplified Virtex-5 XC5VFXxxxT floor plan
- Frequently used high-level components are provided in discrete silicon
- BlockRAM (BRAM): set of blocks that each store up 36 kbits of data
- DSP48 slices: 25x18 bit multipliers followed by a 48 bit accumulator
- CPU: two full embedded PowerPC 440 cores
Virtex-5 Chip Used in Demo Setup

Simplified Virtex-5 floor plan (without embedded CPU cores):

Selected Characteristics

<table>
<thead>
<tr>
<th>Virtex-5</th>
<th>XC5VLX110T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup Tables (LUTs)</td>
<td>69,120</td>
</tr>
<tr>
<td>Block RAM (kbit)</td>
<td>5,328</td>
</tr>
<tr>
<td>DSP48 Slices</td>
<td>64</td>
</tr>
<tr>
<td>PowerPC Cores</td>
<td>0</td>
</tr>
<tr>
<td>max. clock speed</td>
<td>≈ 450 MHz</td>
</tr>
<tr>
<td>release year</td>
<td>2006</td>
</tr>
</tbody>
</table>

Total on-chip memory = 1,120 kbit in LUTs + 5,328 kbit in BRAM = 6,448 kbit = 806 kB
Development Board Used in Demo Setup

- FPGA: Virtex-5 XCV5LX110T
- 256 MB DDR2 memory
- PCI Express (1x lane)
- 1 Gb Ethernet port
- DVI port
- RS232 serial port
- LCD display
- Price: $1999

FPGA Basics

Programming & Design Tools
Programming FPGAs

- Circuits are specified in Hardware Description Language (HDL) or using Design Schematics
- Schematic designs are easier to visualize
- HDL designs easier for large hierarchical designs
- HDL: Formal description digital logic circuits
  - VHDL
  - Verilog

VHDL:
- VHDL = VHSIC Hardware Description Language
- VHSIC = Very High Speed Integrated Circuits
- VHSIC was a U.S. DoD program launched 1980
- Strongly typed (based on Ada)

Verilog:
- Developed by a company, 1995 IEEE standard
- C-like syntax
- Data types wire and reg
- Gate level modelling possible
FPGA Design Flow

- **Design Entry**: circuit design input of HDL Code (Verilog, VHDL) or schematics
- **Synthesis**: compilation of HDL code into device independent primitives
- **Mapping** to device specific elements (LUTs, DSP48, BRAM, etc.)
- **Placement** of components on chip
- **Routing** signals on chip
- **Device Programming**: Download of bitstream to FPGA
Adding Design Constraints

Design Entry

Synthesis

Mapping

Place and Route

Device Programming

User Constraints

Pin Locations on Chip

Timing and Latency Constraints
Signals on a Chip

Example: 1-bit Counter

\[
x \rightarrow \text{XOR} \rightarrow D \rightarrow y
\]

- \( x \rightarrow \) input
- \( \text{XOR} \rightarrow \) XOR gate
- \( D \rightarrow \) D Flip Flop
- \( y \rightarrow \) output

\[\text{Flip Flop}\]

VHDL Code:

```vhdl
process(clock,x) is
begin
  if rising_edge(clock) then
    y <= x xor y;
  end if;
end process;
```

- **Signals do not change immediately**
- **Delays must be considered in design**
- **Propagation Delay** from input to outputs
- **Setup Time**: Signals must be stable before clock beat.

March 25, 2010

René Müller, Jens Teubner · ETH Zurich · Systems Group
Long Signal Paths in Combinatorial Circuits

\[ f_{\text{clk}} < \frac{1}{L} \]

- stages: 1
- latency: \( L \)
- throughput: \( \frac{1}{L} \)
Reducing Path Delays by Inserting Registers

\[ \frac{1}{L} < f_{\text{clk}} < \frac{1}{L'}, \quad L' < L \]

- **Stages:** 3
- **Latency:** \( \frac{3}{f_{\text{clk}}} > L \)
- **Throughput:** \( \frac{1}{2} f_{\text{clk}} \)
Fully Pipelined Signal Path

\[ \frac{1}{L} < f_{\text{clk}} < \frac{1}{L'}, \quad L' < L \]

- stages: 3
- latency: \( \frac{3}{f_{\text{clk}}} > L \)
- throughput: \( f_{\text{clk}} \)
Demo 1

“Hello World” Circuit in VHDL — A simple “0 to 9” Counter
Illustration of Design Flow
Demo: “0 to 9” Counter

- Development board connected to notebook through serial port
- Button clicks are fed to counter
- Counter values are displayed in text form on terminal
VHDL Example: Counter

-- entity specification
entity counter is
port (  
  clk      : in STD_LOGIC;
  enable   : in STD_LOGIC;
  reset    : in STD_LOGIC;
  count    : out STD_LOGIC_VECTOR(  
    3 downto 0));
end counter;

-- behavioral implementation
architecture Behavioral of counter is
begin
  process(clk, reset, enable) is
    variable counter : integer
    range 0 to 9 := 0;
  begin
    if reset='1' then
      counter := 0;
      elsif clk'event and clk='1' and enable='1' then
        if counter = 9 then
          counter := 0;
        else
          counter := counter+1;
        end if;
        count <= std_logic_vector(  
          to_unsigned(counter, 4));
    end if;
  end process;
end Behavioral;
Programming FPGAs using high-level languages (HLL)

- Writing circuits in VDHL or Verilog more difficult than writing programs for a CPU in a HLL
- Different level of abstraction
- In hardware time is a functional property
- ⇒ Automatic generation of HDL code

Examples that extend HLL:
- Handel-C (Oxford): C-like, sequential w/ parallel blocks, channels between parallel blocks
- Bluespec: Based on Haskell.

Examples that use libraries in HLL:
- SystemC: Macro/Libs in C++
- JHDL: Based on Java
- ImpulseC: C Library
FPGA Design Techniques

Exploiting the Parallelism in FPGAs
Parallelism

- FPGAs essentially provide configurable chip space.
- This enables true parallelism in a natural way.

Sub-circuits can operate fully independently.

**Example:** evaluate multiple WHERE predicates in parallel.

We look into data parallelism and pipeline parallelism now.
Data Parallelism

▶ A particular use is data parallelism.

Same operation on multiple input data
Often referred to as SIMD (Single Instruction Multiple Data)
SIMD-Aware Algorithms

The availability of SIMD in mainstream systems made it attractive to design SIMD-aware algorithms.

- **Database Tasks:**

- **XML Processing, Pattern Matching:**
SIMD-Aware Algorithms (cont.)

- **Sorting:**

SIMD-Aware algorithms can often guide FPGA designs.
SIMD Algorithms on FPGAs

- Example: Parallelized sort kernel

- Sort pairs of numbers in parallel.
SIMD Algorithms on FPGAs (cont.)

- Sort (slightly) larger input sets using sort networks.

Here: even-odd merging network [Batcher 1968]

- 2–4 parallel operations at every stage; sort 8 items in 6 stages
Pipeline Parallelism

- But we can also do **pipeline parallelism**.

- All tasks run truly in parallel.
- Simple and efficient **communication** between tasks
  - Contrast to CPU-based setups, where communication overhead dominates when tasks are simple.
Pipeline Parallelism—Example

▶ Example: sorting network (again)

▶ All comparators operate concurrently.

▶ **Throughput**: one 8-set per clock cycle

▶ **Latency**: 6 clock cycles
Pipeline Parallelism / Systolic Arrays

Another way of looking at such circuits is to see them as **systolic arrays** or **wavefront arrays**.

- Very successful VLSI design technique, established in the 80s.
- Processing is driven by **data** that travels through the array.
- Very successful for **matrix multiplication**.
Example: Apriori Algorithm [Baker et al. 2005]

Here: support calculation

1. Load processing units with candidate sets.
2. Stream data through array and count support.

Baker and Prasanna. Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs. *FCCM 2005.*
Which Type of Parallelism When?

Rather than organizing apriori calculation as a **systolic array**, Baker *et al.* could as well have parallelized processing for each item:

Why did they favor the systolic array?
Which Type of Parallelism When?

Systolic array:

```
    data
  unit 1 ──────────── unit 2 ──────────── unit 3 ──────────── unit 4 ──────────── unit 5
```

MISD:

```
data
 unit 1 ──────────── unit 2 ──────────── unit 3 ──────────── unit 4 ──────────── unit 5
      ↓          ↓          ↓          ↓          ↓
```

- The latter approach leads to significantly longer signal paths.
- Systolic arrays, by contrast, have good scalability properties.
  - Even across chips; Baker et al.: 64 FPGAs
- Systolic arrays also have a simple structure which makes them easier to route (by tool chain).
Example: Frequent Item Problem [ICDE 2010]

Throughput [million items/sec] vs. number of items monitored / length of array.

- Systolic array
- Parallel unit access
Database Queries

- Database queries are naturally pipelineable.
- Contrast to CPU-based systems, operators execute truly in parallel.

Example:

```sql
SELECT Price, Volume
FROM Trades
WHERE Symbol = "UBSN"
AND Volume > 100000
```
Synchronous Circuits

The circuit on the right executes the query step-by-step.

- Each sub-task stores its result in registers (where the next sub-task picks it up).
- One sub-task per clock cycle.
- Fully pipelineable
- throughput: 1 per cycle, latency: 5 cycles.

This is a fully synchronous circuit.
Asynchronous Circuits

Asynchronous Circuits

March 25, 2010
René Müller, Jens Teubner · ETH Zurich · Systems Group
Synchronous or Asynchronous?

In general,

synchronous designs

- are easier to implement,
- may be able to reuse logic for multiple tasks,

asynchronous designs

- can achieve lower latency,
- need less flip-flop registers (no intermediate buffers).

Usually what you want is a hybrid design:

- pack a few operations into one clock cycle.

Note that the clock frequency is a variable, too.
FPGA Design Techniques

Some More FPGA Features
Dual-Ported BRAMs

On-chip memory is provided by so-called **block RAM** (or BRAM):

- numerous **independent** blocks (XC5VLX110T: 148 × 36 kbit),
- configurable **word size** (36 kbit as 4096 × 9 bit, 2048 × 18 bit, . . . ).

All BRAM blocks are **dual-ported**:

- **Two independent** ports to access **same physical data**:¹

  ![Diagram showing dual-ported access to physical data](image)

  - The two ports can be configured to **different** word sizes.
  - **Example**: binary trees: access parent and both children together.

¹Effect of concurrent writes to same location is undefined.
Content-Addressable Memory

Independent word sizes can be used to build content-addressable memory (CAM):

- hardware-implemented key-value store,
- guaranteed constant lookup time (unlike hash tables in software),
- standard device in networking (routing, packet classification).

Idea:

- single-bit writes (address: \textit{key} \textbf{+} \textit{value})
- multi-bit reads (address: \textit{key})
Finite State Machines

**Finite state machines** naturally translate into FPGA circuits.

- States are flip-flops, transitions are logic.
- Available parallelism $\rightarrow$ **non-deterministic automata**.

Example: **XPath evaluation**


Likewise: **pattern detection**, etc.
Clock Regions

Like (almost) everything else, the clock frequency can be configured.

- The longest signal path determines the maximum frequency.
  → Tune size of asynchronous units vs. frequency.

- Different chip regions can be clocked at different rates.
  → Choose optimal clock rate for individual units of a design.

- Use FIFOs to decouple units with different clock rates.

- The clock frequency also influences power consumption.

- Or turn the clock off for regions to save power (clock gating).
FPGA in Database Management Systems

Integration Aspects
System Architectures — Overview

- On-Chip co-processor to the Embedded CPU
  - Xilinx MicroBlaze CPU
  - PowerPC Embedded Core

- Co-processor in traditional systems
  - PCI Express attachment
  - HyperTransport attachment: FPGA in a CPU socket

- FPGA in I/O data path

![Diagram showing data flow between CPU, network, disk, FPGA, and other elements.]
Co-processor to Embedded CPU Core

- External Memory (DDR2)
- On-chip Memory (BRAM)
- Embedded CPU Core
- Custom HW Circuit

Connections:
- **input data** from External Memory (DDR2) to Embedded CPU Core
- **result data** from Embedded CPU Core to Custom HW Circuit
Designing an On-chip Co-processor

- On-chip components connected by several bus systems
  - Local Memory Bus: dedicated memory bus
  - Processor Local Bus: wide bus for fast components (Memory)
  - On-chip Peripheral Bus: narrow bus for slow components (Peripherals)
- Buses implemented in configurable HW
- Buses require \textit{arbitrated} access → overhead, latency
- Better: Point-to-Point Links → no arbitration needed
- Simplex Links: Unidirectional connections
  - Fast Simplex Link: 32-bit wide, essentially a 16 element FIFO
- Some CPUs have dedicated Co-processor interface → Auxiliary Processing Unit (APU) Interface
Xilinx MicroBlaze CPU

- Soft Intellectual Property (Soft-IP) core: implementation in configurable logic
- 32-bit MIPS Architecture
- Closed source, only compiled form (netlist)
- Supported features chosen at configuration time
- Trade-off: functionality vs. chip space
- Max. Clock Speed: 235 MHz on Virtex-5 (=280 Dhrystone MIPS)

Features:
- Floating Point Unit
- Hardware Divider
- Memory Management Unit
- Exception Support
- Fast Simplex Link Interfaces
- Local Memory Bus to on-chip memory
Fast Simplex Link to MicroBlaze Execution Pipeline

- Fast Simplex Link (FSL) to Pipeline
- Up to 16 FSL
- Put/Get instructions: Register ↔ FSL
  - put %regA, rfs1X
  - get %regA, rfs1X
- Also blocking and non-blocking versions
PowerPC Hard-IP Core

- Virtex-5 XC5VFXxxxT contains two PowerPC 440 cores
- hard-IP core: implemented in silicon, not configurable
- PPC440 core also used in BlueGene Supercomputer
- 32-bit CPU
- Up to 400 MHz clock
- 32 kB Data and 32 kB Instruction Cache
- Memory Management Unit
- Auxiliary Processor Unit Interface for connecting co-processors (e.g., FPU)
PowerPC APU

- Auxiliary Processing Unit (APU) Interface allows extension of the PowerPC instruction
- Most common use: integration of an FPU
- APU controller decodes PPC FPU and VMX (Altivec) vector instructions
- Plus up to 16 User-defined Instructions
- Load/Store instructions
PCI Express Bus

- PCIe: Packet-based, point-to-point serial interface
- Multiple Lanes: x1 – x32

<table>
<thead>
<tr>
<th>Link</th>
<th>Raw Bw per dir.</th>
<th>effective Bw per dir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>2.5 Gb/s</td>
<td>2 Gb/s</td>
</tr>
<tr>
<td>x2</td>
<td>5 Gb/s</td>
<td>4 Gb/s</td>
</tr>
<tr>
<td>x4</td>
<td>10 Gb/s</td>
<td>8 Gb/s</td>
</tr>
<tr>
<td>x8</td>
<td>20 Gb/s</td>
<td>16 Gb/s</td>
</tr>
</tbody>
</table>

- PCIe available in commodity hardware
- Demo Board has PCIe x1 connector
- PCIe defines roles:
  - PCIe Root Complex (Southbridge)
  - PCIe Switch
  - PCIe Endpoint, e.g., NIC
- FPGA takes role of Endpoint
- Endpoint must be implemented on FPGA
  - Virtex-5 hard-IP core plus soft-core wrapper
  - Freely available Implementation of PCIe Endpoint (R. Bittner, MSR Redmond)
FPGA Co-processor on a PCI Express Card

- FPGA Chip
  - On-chip Memory (BRAM)
  - Embedded CPU Core
  - Custom HW Circuit
  - PCIe Endpoint

- External Memory (DDR2)
- PCI Express link to host system
FPGA Co-processor on HyperTransport Bus

- FPGA directly connected to CPU HyperTransport
- Tight coupling to Server CPUs

- FPGA in CPU Socket → RPRU Module (DRC Computer Corp.)
- FPGA in HTX Socket → HTX Board (U. Mannheim)
FPGAs in the Data Path

- FPGA in Data Path to CPU
  - From Network → Streams on Wires—A Query Compiler for FPGAs (Müller, Teubner, and Alonso) in VLDB09
  - From Disk → Netezza Data Warehouse Applicance
- Off-loading to FPGA for Data Reduction
  - Selection/Projection
  - Aggregation
- Reduces traffic to the CPU
Processing Packets at Wire Speed (ongoing work)

- Stream Processing Application
- Filtering Tuples in UDP datagrams
- IP/UDP Engine implemented in hardware
- Connected to 1 Gb Ethernet MAC
- PC System drops packets (high interrupt rate)
- FPGA Solution allows processing packets at wire speed w/o loss.

March 25, 2010
René Müller, Jens Teubner · ETH Zurich · Systems Group
Demo 2

Data Stream Aggregation on FPGA
Processing of UDP Network Traffic
Demo 2: Data Stream Aggregation

- Stock Ticker Stream
- Symbol: CHAR(4)
- Price: Price in Cents
- Volume: # shares

<table>
<thead>
<tr>
<th>Seqnr</th>
<th>Symbol</th>
<th>Price</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>2245</td>
<td>BAER</td>
<td>3551</td>
<td>75</td>
</tr>
<tr>
<td>2246</td>
<td>UBSN</td>
<td>622</td>
<td>47</td>
</tr>
<tr>
<td>2247</td>
<td>NOVN</td>
<td>4637</td>
<td>403</td>
</tr>
<tr>
<td>2248</td>
<td>NESN</td>
<td>2842</td>
<td>166</td>
</tr>
<tr>
<td>2249</td>
<td>UBSN</td>
<td>608</td>
<td>13</td>
</tr>
<tr>
<td>2250</td>
<td>NOVN</td>
<td>4736</td>
<td>118</td>
</tr>
<tr>
<td>2251</td>
<td>ABBN</td>
<td>2505</td>
<td>27</td>
</tr>
</tbody>
</table>

Aggregation Query:
```
SELECT Symbol, avg(Price)
FROM Trades [SIZE 15
ADVANCE 5 TIME]
GROUP BY Symbol
```

In Demo max. 32 groups
System Setup

- Ethernet (input stream)
- serial line (output)
- serial port
- NIC
- FPGA
Power Consumption

One of the virtues of FPGAs is their **low power consumption**.

This circuit:

- total FPGA power consumption: 6.4 W
- part spent to drive I/O pins (network, serial): 4.5 W
- part spent in co-processor: 0.04 W

For comparison: Intel Core 2 Q6700 (desktop CPU): up to 95 W.
Wrap-Up

Summary and Lessons Learned
Summary

FPGA Basics:
- re-configurable logic
- low latency, high throughput, power-efficient

FPGA Design Techniques:
- flexible types of parallelism: data and pipeline parallelism
- addl. tricks: on-chip BRAM, CAM, flexible clock frequency

System Integration
- co-processor to general-purpose CPU
- FPGA in the system’s data path
Lessons and Guidelines

- **Resource consumption** is an important design factor.
- **Pipeline parallelism** eases scalability and performance.
- **Trade-off**: synchronous designs $\leftrightarrow$ asynchronous designs.
- **System integration** can be decisive.
- Designs can be tuned for **power efficiency**.
A New Point in the Design Space

FPGAs work well for **streaming-style, high-throughput** processing.

- Good fit for many database tasks.

Put the FPGA into a system’s **data path**:

Roles of the FPGA can also include **decoding, compression**, etc.

Benefits include **performance**, but also **power efficiency**.