Skeleton Automata for FPGAs
Reconfiguring without Reconstructing

Jens Teubner
jens.teubner@inf.ethz.ch

Louis Woods
louis.woods@inf.ethz.ch

Chongling Nie
chnie@student.ethz.ch

Systems Group, Department of Computer Science
ETH Zurich, Switzerland
FPGAs are **soft hardware** chips.
FPGAs are soft hardware chips

(Re)programmable: load arbitrary circuit onto FPGA
FPGAs are **soft hardware** chips

- (Re)programmable: load arbitrary circuit onto FPGA
- Programmed FPGA $\equiv$ dedicated hardware (ASIC)
Field-Programmable Gate Arrays

▶ FPGAs are soft hardware chips
  ➔ (Re)programmable: load arbitrary circuit onto FPGA
  ➔ Programmed FPGA ≡ dedicated hardware (ASIC)

▶ FPGAs for database applications
Insert FPGA into the **data path**

source → raw data → **FPGA** → filtered data → CPU

**FPGA Integration into a Database System**
Insert FPGA into the data path

source → raw data → FPGA → filtered data → CPU

simple efficient
Insert FPGA into the **data path**

- **Source**
- **Raw data** → **FPGA** → **Filtered data** → **CPU**

- **Simple and efficient**
FPGA Integration into a Database System

Insert FPGA into the **data path**

source → raw data → FPGA → filtered data → CPU

- simple
- efficient
- complex

Paradigm → process data **near its source**
Expressiveness vs. Reconfiguration Speed

query expressiveness

reconfiguration speed
Expressiveness vs. Reconfiguration Speed

- Query expressiveness
- Reconfiguration speed

Off-line query-to-circuit compilation

Skeleton Automata: High query expressiveness and dynamic query workloads here: Skeleton Automata for XML projection.
Expressiveness vs. Reconfiguration Speed

query expressiveness

off-line query-to-circuit compilation

on-line query workload changes

reconfiguration speed

Louis Woods
Skeleton Automata: Reconfiguring without Reconstructing
Expressiveness vs. Reconfiguration Speed

query expressiveness

off-line query-to-circuit compilation

this work

on-line query workload changes

reconfiguration speed

Skeleton Automata:

- high query expressiveness and dynamic query workloads
Expressiveness vs. Reconfiguration Speed

query expressiveness

off-line query-to-circuit compilation

on-line query workload changes

reconfiguration speed

this work

Skeleton Automata:

- high query expressiveness and dynamic query workloads
- here: skeleton automata for XML projection
XML Projection
for $i$ in //regions//item
    return <item>
        { $i/name }
        <num-categories>
            { count ($i/incategory) }  
        </num-categories>
    </item>

XML Document

A. Marian and J. Siméon, Projecting XML Documents, VLDB03
Projection Paths

XML Document

for $i$ in //regions/item
    return <item>
        { $i/name }
        <num-categories>
        { count ($i/incategory) }
    </num-categories>
</item>

Query

{} //regions/item,
{} //regions/item/name #,
{} //regions/item/incategory
Hybrid XQuery Engine

Projection Paths

\{
  //regions//item,
  //regions//item/name #,
  //regions//item/incategory
\}

Query

for $i in //regions//item
  return <item>
    { $i/name }
    <num-categories>
      { count ($i/incategory) }
    </num-categories>
  </item>
In-Network Filtering

Original Document

<site>
  <regions>
    ...
    <item id="item42">
      <name>foo</name>
      <incategory category="category3"/>
    </item>
    ...
  </regions>
  ...
  <open_auctions>
    <open_auction id="open_auction0">
      ...
    </open_auction>
    <open_auction id="open_auction1">
      ...
    </open_auction>
    ...
  </open_auctions>
  ...
</site>

Filtered Document

<site>
  <regions>
    <item>
      <name>foo</name>
      <incategory category="category3"/>
    </item>
    ...
  </regions>
  ...
  <open_auctions>
    <open_auction id="open_auction0">
      ...
    </open_auction>
    <open_auction id="open_auction1">
      ...
    </open_auction>
    ...
  </open_auctions>
  ...
</site>

Query Result

<item>
  <name>foo</name>
  <num-categories>2</num-categories>
</item>
Projection Path Circuits
Automaton (NFA) for projection path //a/b/c//d
Automaton (NFA) for projection path //a/b/c//d
From Queries to Hardware

XPath

Hardware FSM

/bitstream

FPGA

Louis Woods

Skeleton Automata: Reconfiguring without Reconstructing
From Queries to Hardware

XPath  \(/a//b\)  Hardware FSM  bitstream

\[
\begin{array}{c}
\text{a} \\
\downarrow \\
\text{b}
\end{array}
\]

\[
\downarrow
\]

FPGA

\[\geq\text{several hours}!\]
From Queries to Hardware

XPath

\(/a//b\) → Hardware FSM → bitstream

\(a \rightarrow b \rightarrow *\)

FPGA

≥ several hours!

Digital Circuit

or

and
From Queries to Hardware

XPath

$/a//b$ → Hardware FSM

$\begin{array}{c}
\text{a} \\
\text{b}
\end{array}$ → FPGA

$\geq$ several hours!

Digital Circuit

or

and

Synthesis
Map
Place & Route

Louis Woods
Skeleton Automata: Reconfiguring without Reconstructing
From Queries to Hardware

XPath

/a//b

Hardware FSM

\(a \rightarrow b \ast\)

several hours!

FPGA

bitstream

Digital Circuit

or

and

VHDL

Synthesis

Map

Place & Route

bitstream

FPGA
Skeleton Automata
Structure of Projection Path Automaton

Automaton for //a/b/c//d:

```
\( q_0 \) \quad a \quad \rightarrow \quad q_1 \quad b \quad \rightarrow \quad q_2 \quad c \quad \rightarrow \quad q_3 \quad d \quad \rightarrow \quad q_4
```

XML

```
<xml>
  <tag>
    <a/>
  </tag>
</xml>
```

Decoding

```
<table>
<thead>
<tr>
<th>q_0</th>
<th>q_1</th>
<th>q_2</th>
<th>q_3</th>
<th>q_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>or</td>
<td>and</td>
<td>and</td>
<td>and</td>
<td>or</td>
</tr>
</tbody>
</table>
```

Louis Woods

Skeleton Automata: Reconfiguring without Reconstructing
Automaton for //a/b/c//d:

```
root()/desc:: a/child:: b/child:: c/desc:: d
```

Or

```
tag decod. a
```

And

```
and
```
Automaton for //a/b/c//d:

![Automaton Diagram]

- $q_0$ transitions to $q_1$ on input $a$.
- $q_1$ transitions to $q_2$ on input $b$.
- $q_2$ transitions to $q_3$ on input $c$.
- $q_3$ transitions to $q_4$ on input $d$.
- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.

- $q_0$ transitions to $q_1$ on input *.
- $q_1$ transitions to $q_2$ on input *.
- $q_2$ transitions to $q_3$ on input *.
- $q_3$ transitions to $q_4$ on input *.

- $q_0$ transitions back to itself on input *.
- $q_1$ transitions back to itself on input *.
- $q_2$ transitions back to itself on input *.
- $q_3$ transitions back to itself on input *.
Static and Dynamic Parts

XPath spec. → skeleton

×Path spec.
Static and Dynamic Parts

XPath spec. → skeleton → FPGA

configuration parameters

static part (off-line)
dynamic part (runtime)

minutes/hours

⊥

a∗b
Static and Dynamic Parts

- **XPath spec.**

  - skeleton

  - static part (off-line)
    - user query /a//b
    - configuration parameters

  - dynamic part (runtime)

  - FPGA

  - minutes/hours

Louis Woods  
Skeleton Automata: Reconfiguring without Reconstructing  
14
Static and Dynamic Parts

XPath spec. → skeleton

static part (off-line)

dynamic part (runtime)

Microseconds

configuration parameters

user query

/a//b

minutes/hours

FPGA

⊥

a

b

⊥

*
Multiple Projection Paths

- Exploit pipeline parallelism to evaluate multiple path expressions concurrently
- Propagate global match signal through segment chain
Multiple Projection Paths

- Exploit pipeline parallelism to evaluate multiple path expressions concurrently
- Propagate global match signal through segment chain

XML lexer → segment → ... → segment → ... → segment → segment → serializer

path $p_1$
Multiple Projection Paths

- Exploit pipeline parallelism to evaluate multiple path expressions concurrently.
- Propagate global match signal through segment chain.

XML lexer → segment → ... → segment → ... → segment → serializer

Path $p_1$, Path $p_2$, ...
Multiple Projection Paths

- Exploit **pipeline parallelism** to evaluate multiple path expressions concurrently.
Multiple Projection Paths

- Exploit **pipeline parallelism** to evaluate multiple path expressions concurrently
- Propagate **global match** signal through segment chain
Experiments
Effects of FPGA-Based Projection

→ Saxon-EE; 116.5 MB XML

**Total Processing Time**

<table>
<thead>
<tr>
<th>XMark query</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedup</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Effects of FPGA-Based Projection

→ Saxon-EE; 116.5 MB XML

Query Execution Time

<table>
<thead>
<tr>
<th>XMark query</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedup</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Louis Woods
Skeleton Automata: Reconfiguring without Reconstructing
Effects of FPGA-Based Projection

→ Saxon-EE; 116.5 MB XML

Parsing Time

<table>
<thead>
<tr>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
<tr>
<td>Q4</td>
</tr>
<tr>
<td>Q5</td>
</tr>
<tr>
<td>Q6</td>
</tr>
<tr>
<td>Q7</td>
</tr>
<tr>
<td>Q8</td>
</tr>
<tr>
<td>Q9</td>
</tr>
<tr>
<td>Q10</td>
</tr>
</tbody>
</table>

XMark query
Effects of FPGA-Based Projection

→ Saxon-EE; 116.5 MB XML

Memory Consumption Improvement

XMark query

improvement factor

Q1  Q2  Q3  Q4  Q5  Q6  Q7  Q8  Q9  Q10
Scalability

100% chip utilization

clock frequency [MHz]

number of segments $n$
Scalability

100% chip utilization

clock frequency [MHz]

number of segments $n$
Scalable Circuit Design

XML stream → XML lexer → FPGA with projection engine → segment matchers → filtered XML → serializer
Selected Related Work

▶ Static query-to-hardware compilation

Selected Related Work

▶ Static query-to-hardware compilation


▶ Dynamic query work load changes

→ IBM/Netezza’s 1000 (formerly TwinFin)
Selected Related Work

▶ Static query-to-hardware compilation

▶ Dynamic query work load changes
  → IBM/Netezza’s 1000 (formerly TwinFin)

▶ Analysis of existing trade-offs
  → Sadoghi et al. *Towards Highly Parallel Event Processing through Reconfigurable Hardware*. In DaMoN’11, Athens, Greece, 2011.
Static query-to-hardware compilation


Dynamic query work load changes

IBM/Netezza’s 1000 (formerly TwinFin)

Analysis of existing trade-offs

Sadoghi et al. *Towards Highly Parallel Event Processing through Reconfigurable Hardware*. In DaMoN’11, Athens, Greece, 2011.

Avalanche research project

http://www.systems.ethz.ch/research/projects/avalanche

Funded by SNSF and by the Enterprise ComputingCenter (ECC) of ETH Zurich
Conclusions

Setting

Hybrid (FPGA/CPU) XQuery engine with **FPGA in the data path**
Conclusions

► Setting
   → Hybrid (FPGA/CPU) XQuery engine with FPGA in the data path

► Problem
   → Reprogramming FPGA too slow for dynamic query workloads
   → Yet, we do not want to sacrifice expressiveness
Conclusions

Setting

→ Hybrid (FPGA/CPU) XQuery engine with **FPGA in the data path**

Problem

→ Reprogramming FPGA **too slow** for dynamic query workloads
→ Yet, we do not want to sacrifice **expressiveness**

Solution

→ **Skeleton automata** separate the **structure** of finite-state automata from their **semantics**