XXIIème Colloque GRETSI - Traitement du Signal et des Images

Plate-forme de conception d’architectures reconfigurables dynamiquement pour le domaine du TSI

Julien Lallet - Sébastien Pillement - Olivier Sentieys

Dijon, 9 septembre 2009
AGENDA

► Context and motivations
► Application/architecture design flow
► Related works
► xMAML Architecture Description Language
► Architecture exploration
  ▶ Dynamic Implementation of a WCDMA receiver
  ▶ eFPGA, coarse-grain reconfigurable processor (DART)
► Conclusions and work in progress
Related works

xMAML language

Architecture exploration

Conclusions and work in progress

Context: VLSI Design of Dynamically Reconfigurable Architectures

Context and motivations of this work:
Context and motivations of this work:

- A **Dynamically Reconfigurable Architectures (DRA)** is
  - a set of **configurable functional units** (e.g. arithmetic operators, logic blocks)
  - interconnected via a **configurable (hierarchical) network**
  - which enables *(partial)* **dynamic reconfiguration management**
Context and motivations of this work:

- **A Dynamically Reconfigurable Architectures (DRA) is**
  - A set of **configurable functional units** (e.g. arithmetic operators, logic blocks)
  - Interconnected via a **configurable (hierarchical) network**
  - Which enables **(partial) dynamic reconfiguration management**

- Trade-off between **performance, energy and flexibility**
Context and motivations of this work:

- **A Dynamically Reconfigurable Architectures (DRA)** is
  - a set of **configurable functional units** (e.g. arithmetic operators, logic blocks)
  - interconnected via a **configurable (hierarchical) network**
  - which enables **(partial) dynamic reconfiguration management**

- Trade-off between **performance**, **energy** and **flexibility**

- **Design of generic DRA ”platforms“**
  - e.g. a DR-FPGA embedded into an SoC, a coarse-grained reconf. arch., etc.
  - (and not the use of commercial platform such as Xilinx Virtex)
Context and motivations of this work:

- A Dynamically Reconfigurable Architectures (DRA) is
  - a set of **configurable functional units** (e.g. arithmetic operators, logic blocks)
  - interconnected via a **configurable (hierarchical) network**
  - which enables **(partial) dynamic reconfiguration management**

- Trade-off between **performance**, **energy** and **flexibility**

- **Design of generic DRA ”platforms“**
  - e.g. a DR-FPGA embedded into an SoC, a coarse-grained reconf. arch., etc.

Contributions:

- Generic architecture model

- A new Architecture Description Language for DRA: xMAML

- A set of CAD tool to synthesize the chip from xMAML
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
**Related Works**

**XMAML Language**

**Architecture Exploration**

**Conclusions and work in progress**

**Context: Generic DRA Platform**

**Generic DRA Platform**

Hardware parameters

1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
2. Partial Reconfiguration management
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
2. Partial Reconfiguration management
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
2. Partial Reconfiguration management
3. Multiple contexts
### Context: Generic DRA Platform

**Generic DRA platform**

**Hardware parameters**
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

**Characteristics**
1. How dynamic (fast) is the reconfiguration
2. Partial Reconfiguration management
3. Multiple contexts
CONTEXT: GENERIC DRA PLATFORM

Generic DRA platform

Hardware parameters
1. Data bus size
2. Number of functional units (FU)
3. Connection schemes
4. FU architectures

Characteristics
1. How dynamic (fast) is the reconfiguration
2. Partial Reconfiguration management
3. Multiple contexts
4. Preemption
Contributions: Application and Architecture Design Flow Using xMAML Model
**Contributions: Application and Architecture Design Flow using xMAML Model**

**Architecture Flow (MOZAIC):**
- **Constraints**
- **Architecture Modeling (xMAML)**
- **Component Library (IP)**
- **Architecture Generator**
- **Bitstream Generator**
  - Bitstream (Binary file)
  - Architecture (VHDL)

**Application Flow:**
- **Application**
- **Synthesis**
- **Compilation**
- **Place/Route**

---

**Related Works**

**xMAML Language**

**Architecture Exploration**

**Conclusions and Work in Progress**
CONTRIBUTIONS: APPLICATION AND ARCHITECTURE DESIGN FLOW USING xMAML MODEL

Application flow
- Application
- Synthesis
- Compilation
- Place/Route
- Bitstream generator

Architecture flow (MOZAIC)
- Architecture modeling (xMAML)
- Component Library (IP)
- Architecture generator
- Constraints

Bitstream generator
- Bitstream (Binary file)
- Architecture (VHDL)

FPGA

PLATE-FORME DE CONCEPTION D’ARCHITECTURES RECONFIGURABLES DYNAMIQUEMENT
**Contributions: Application and Architecture Design Flow using xMAML Model**

- **Application flow**
  - Application
  - Synthesis
  - Compilation
  - Place/Route

- **Architecture flow (Mozaic)**
  - Architecture modeling (xMAML)
  - Component Library (IP)
  - Architecture generator

- **Bitstream generator**
  - Bitstream (Binary file)

- **Constraints**

**FPGA**

**DART**
Dynamically Reconfigurable Architectures

- Fine-Grain DRA: AT40k, Virtex
- Coarse-Grain DRA: Systolic Ring [Sassatelli02], DART [David03], etc.
- Multiple contexts: DPGA [Dehon96], PipeRench [Schmit02], PiCoGa [Cappelli04], Adres [Mei05], etc.
 RELATED WORKS: DYNAMICALLY RECONFIGURABLE ARCHITECTURES 1/2

- **Dynamically Reconfigurable Architectures**
  - Fine-Grain DRA: AT40k, Virtex
  - Coarse-Grain DRA: Systolic Ring [Sassatelli02], DART [David03], etc.
  - Multiple contexts: DPGA [Dehon96], PipeRench [Schmit02], PiCoGa [Cappelli04], Adres [Mei05], etc.

- **Tools for reconfigurable platforms**
  - Dedicated tools for dynamic platforms: Xilinx PlanAhead, Dresc [Mei07], DART [David02]
  - Generic tools for static FPGAs: MADEO [Lagadec00], VPR [Betz97]
  - No generic tools for DRA modeling and compilation/synthesis

⇒ Need of Architecture Description Language (ADL) for DRA modeling
RELATED WORKS: ARCHITECTURE DESCRIPTION LANGUAGES 2/2

Architecture description languages
- Structural ADLs: MIMOLA [Bashford94], UDL/I [Karatsu91]
- Behavioral ADLs: nML [Fauth95], ISDL [Hadjyiannis94]
- Mixed ADLs: MADEO [Lagadec00], MAML [Kupriyanov07], ARMOR [Charot99]

⇒ We propose xMAML as an extension of MAML for:
- hierarchical, regular, generic, reconfigurable platform with dynamic reconfiguration features
Architecture Description with xMAML

Reconfiguration Unit
Input/Output Unit
Processing Element Unit
Interconnection Unit

UIo
UIo
UIo
ARCHITECTURE DESCRIPTION WITH xMAML

- Reconfiguration Unit
- Input/Output unit
- Processing Element Unit
- Interconnection Unit

A MAML-ARCHITECTURE EXPLORATION

Conclusions and work in progress

PLATE-FORME DE CONCEPTION D’ARCHITECTURES RECONFIGURABLES DYNAMIQUEMENT
ARCHITECTURE DESCRIPTION WITH xMAML

Reconfiguration Unit
Input/Output unit
Processing Element Unit
Interconnection Unit
Architecture Description with xMAML
xMAML description \texttt{PEinterface} (DUCK) allows fast specification of the reconfiguration parameters of one \texttt{Upe} for the generation of the RTL specification for synthesis

- \texttt{<Reconfiguration>} (cycle, bits, preemption)

\begin{center}
\begin{tikzpicture}
\node[draw] (reconf) at (0,0) {\texttt{Reconfiguration} : cycle, bitwidth, preemption};
\node[draw, below of=reconf] (elements) {\texttt{PElements} : names};
\end{tikzpicture}
\end{center}
xMAML description `PEinterface` (DUCK) allows fast specification of the reconfiguration parameters of one `UPe` for the generation of the RTL specification for synthesis

- `<Reconfiguration>` (cycle, bits, preemption)
- `<IOPort>` (name, bitwidth, direction, type)
  - `clk, rst, data, ctrl, ScanIn, ScanEn, RamConfAddr, RamConfIn, RamConfEn, IR, priority`

```
xMAML description
PEinterface (DUCK) allows fast specification of the reconfiguration parameters of one UPe for the generation of the RTL specification for synthesis

- `<Reconfiguration>` (cycle, bits, preemption)
- `<IOPort>` (name, bitwidth, direction, type)
  - `clk, rst, data, ctrl, ScanIn, ScanEn, RamConfAddr, RamConfIn, RamConfEn, IR, priority`
```

```
Reconfiguration : cycle, bitwidth, preemption

IOPort : name, bitwidth, in, type

PElements : names

IOPort : name, bitwidth, out, type
```

---

**Related Works**

**XMAML Language Architecture Exploration**

**Conclusions and work in progress**

---

**Reconfiguration Unit: DUCK (Dynamical Unifier and reConfiguration bloCk) 1/2**

---

**Note:**

- `PEinterface`: Dynamical Unifier and Reconfiguration Block
- `UPe`: Unit Processing Element
- `IR`: Instruction Register
Reconfiguration Unit: DUCK (Dynamical Unifier and reConfiguration blocK) 2/2

```xml
<PEInterface name="LB">
  <Reconfiguration cycle = "1" bits = "17" preemption="no"/>
  <IOPorts>
    <Port name="output_0" bitwidth="1" direction="out" type="data"/>
    <Port name="input_0" bitwidth="1" direction="in" type="data"/>
    <Port name="input_1" bitwidth="1" direction="in" type="data"/>
    <Port name="input_2" bitwidth="1" direction="in" type="data"/>
    <Port name="input_3" bitwidth="1" direction="in" type="data"/>
    <Port name="rst" bitwidth="1" direction="in" type="rst"/>
    <Port name="clk" bitwidth="1" direction="in" type="clk"/>
    <Port name="ScanIn_0" bitwidth="1" direction="in" type="ScanIn"/>
    ...<Port name="ScanIn_16" bitwidth="1" direction="in" type="ScanIn"/>
    <Port name="ConfEn" bitwidth="1" direction="in" type="ScanEn"/>
  </IOPorts>
</PEInterface>
```
INTERCONNECTION UNIT: DyRIBox 1/2

DyRIBox parameters

- Reconfiguration (cycle, bitwidth)
**INTERCONNECTION UNIT: DyRIBox 1/2**

**DyRIBox parameters**

- **Reconfiguration (cycle, bitwidth)**
- **DBPorts**
  - Inputs: number, width
  - Outputs: number, width

![Diagram of DyRIBox parameters]
**INTERCONNECTION UNIT: DyRIBox 1/2**

DyRIBox parameters

- **Reconfiguration** (cycle, bitwidth)
- **DBPorts**
  - Inputs: number, width
  - Outputs: number, width

- **PElementsPorts**
  - Inputs: number, width
  - Outputs: number, width

\[
\text{DBPorts Inputs : number, width} \\
\text{PElementsPorts Inputs : number, width} \\
\text{DyRIBox} \\
\text{DBPorts Outputs : number, width} \\
\text{PElementsPorts Outputs : number, width}
\]
**Related Works**

**XMAML Language**

**Architecture Exploration**

**Conclusions and work in progress**

**Interconnection Unit: DyRIBox 1/2**

DyRIBox parameters

- **Reconfiguration** (cycle, bitwidth)
- **DBPorts**
  - Inputs: number, width
  - Outputs: number, width
- **PEElementsPorts**
  - Inputs: number, width
  - Outputs: number, width
- **AdjacencyMatrix**
  - Doutputs: idx, row
  - Poutputs: idx, row

**DBPorts Inputs:** number, width

**PEElementsPorts Inputs:** number, width

**DBPorts Outputs:** number, width

**PEElementsPorts Outputs:** number, width
INTERCONNECTION UNIT: DyRIBox 2/2

```xml
<PEInterconnectDyRIBox name="DBox4">
  <Reconfiguration cycle = "1" bitwidth = "8" />
  <DBPorts>
    <InOut number="16" bitwidth="1" />
  </DBPorts>
  <PEElementsPorts>
    <Inputs number="1" bitwidth="1" />
    <Outputs number="4" bitwidth="1" />
  </PEElementsPorts>
  <AdjacencyMatrix>
    <DInOut idx="0" row= "00001000001000010" />
    <DInOut idx="1" row= "00001000100000100" />
    <DInOut idx="14" row= "00100010010000001" />
    <DInOut idx="15" row= "00010001100000001" />
    <POutput idx="0" row= "00000000111100000" />
    <POutput idx="1" row= "00000000111110000" />
    <POutput idx="2" row= "00000000011110000" />
    <POutput idx="3" row= "00000000001111110" />
  </AdjacencyMatrix>
</PEInterconnectDyRIBox>
```
RELATED WORKS

XMAML LANGUAGE

ARCHITECTURE EXPLORATION

Conclusions and work in progress

RECONFIGURATION DOMAIN SPECIFICATION

```xml
<DBDomain name="DPR_1">
  <ReconfigurationParameters preemption="disable" domainCtrl="shared"
    partialReconfiguration="enable" IRPriorityLevel="3" taskNumber="10"
    confBusWidth="8"/>
  <Interconnect type="manual">
    <Instantiation name="MultBus" instanceOf="DBox"/>
    <InternalConnections>
      <MultBus:in(1) = DataMem1:output_0(0:15)/>
      <MultBus:in(2) = DataMem2:output_0(0:15)/>
      ...
      <AG4:output_0(0:15) = DataMem4:input_0(0:15)/>
    </InternalConnections>
    </Interconnect>
  </DBDomain>
```

PLATE-FORME DE CONCEPTION D'ARCHITECTURES RECONFIGURABLES DYNAMIQUEMENT
Synthesis (CMOS 130nm, Synopsys DC) on several DUCK samples with different data size ($Rs$) and register numbers ($Rn$)

$$SD_{\text{para}} = 563 + Rn \times Rs \times 50 \ [\mu m^2]$$

$$SD_{\text{serial}} = 174 + Rn \times Rs \times 28 \ [\mu m^2]$$
WCDMA APPLICATION

1. SoC design for 3G/UMTS telecommunications
WCDMA APPLICATION

1. SoC design for 3G/UMTS telecommunications
2. Dynamic implementation of a WCDMA receiver
WCDMA APPLICATION

1. SoC design for 3G/UMTS telecommunications
2. Dynamic implementation of a WCDMA receiver
1. SoC design for 3G/UMTS telecommunications
2. Dynamic implementation of a WCDMA receiver
3. DRA exploration
   - eFPGA based on XC4000 logic blocks (LB)
   - DART coarse-grain dynamically reconfigurable processor
Estimation of the DUCK size for LBs

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>SIZE OF THE CONFIGURATION (IN BITS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two LUTs (four inputs)</td>
<td>32</td>
</tr>
<tr>
<td>One LUT (three inputs)</td>
<td>8</td>
</tr>
<tr>
<td>Input/Output connections</td>
<td>5</td>
</tr>
<tr>
<td>Multiplexer connections</td>
<td>10</td>
</tr>
<tr>
<td>Carry chain resources</td>
<td>8</td>
</tr>
<tr>
<td>RAM configuration</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>66</td>
</tr>
</tbody>
</table>
Estimation of the DUCK size for DyRIBox

- 40 outputs ↔ 7 inputs: 120 configuration bits per DyRIBox
eFPGA RESOURCE EXPLORATION 3/5

RTL and logic synthesis (Quartus/ABC Berkeley) of the application on the eFPGA

<table>
<thead>
<tr>
<th>Task</th>
<th>FIR (F1)</th>
<th>Searcher (F2)</th>
<th>Rake Receiver (F3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (LB)</td>
<td>1117</td>
<td>1235</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Finger</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Symbole</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>245</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total : 1030</td>
</tr>
</tbody>
</table>

Total configuration bits per context:
- 89k configuration bits per context
- 148k bits per context
- 237kbits (30kbytes) per configuration for the whole eFPGA
### eFPGA Resource Exploration 3/5

RTL and logic synthesis (Quartus/ABC Berkeley) of the application on the eFPGA

<table>
<thead>
<tr>
<th>Task</th>
<th>FIR (F1)</th>
<th>Searcher (F2)</th>
<th>Rake Receiver (F3)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (LB)</td>
<td>1117</td>
<td>1235</td>
<td>245</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total: 1030</td>
</tr>
</tbody>
</table>

⇒ eFPGA architecture (36 × 36 matrix of 1296 LBs)
Related works

XMAML language

Architecture exploration

Conclusions and work in progress

E FPGA resource exploration 3/5

RTL and logic synthesis (Quartus/ABC Berkeley) of the application on the eFPGA

<table>
<thead>
<tr>
<th>Task</th>
<th>FIR (F1)</th>
<th>Searcher (F2)</th>
<th>Rake Receiver (F3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (LB)</td>
<td>1117</td>
<td>1235</td>
<td></td>
</tr>
<tr>
<td>FINGER</td>
<td>245</td>
<td>50</td>
<td>Total : 1030</td>
</tr>
</tbody>
</table>

⇒ eFPGA architecture (36 × 36 matrix of 1296 LBs)

- 66 configuration bits per LB
- 1235 LB ⇒ 89k configuration bits per context
- 120 configuration bits per DyRIBox
- ⇒ 148kbits per context
### eFPGA Resource Exploration 3/5

RTL and logic synthesis (Quartus/ABC Berkeley) of the application on the eFPGA

<table>
<thead>
<tr>
<th>TASK</th>
<th>FIR (F1)</th>
<th>SEARCHER (F2)</th>
<th>RAKE RECEIVER (F3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST (LB)</td>
<td>1117</td>
<td>1235</td>
<td>245 50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total : 1030</td>
</tr>
</tbody>
</table>

⇒ eFPGA architecture (36 × 36 matrix of 1296 LBs)

- 66 configuration bits per LB
- 1235 LB ⇒ 89k configuration bits per context
- 120 configuration bits per DyRIBox
- ⇒ 148kbits per context

⇒ 237kbits (30kbytes) per configuration for the whole eFPGA
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
- Number of configuration bytes: $T_{dc} = 30 \times 10^3 \text{bytes}$
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
- Number of configuration bytes: $T_{dc} = 30 \cdot 10^3 \text{ bytes}$
- Access frequency of embedded SRAM memory: $V_m = 300 MHz$
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
- Number of configuration bytes: $T_{dc} = 30.10^3 bytes$
- Access frequency of embedded SRAM memory: $V_m = 300 MHz$

\[
N_D = \left[ \frac{T_{dc}}{V_m} \frac{V_m}{T_i} \frac{T_i}{N_c} \right]
\]
How many domains ($N_D$) to fulfill the timing constraints?

- **Number of context:** $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- **Time for slot sample acquisition:** $T_i = 65 \, \mu s$
- **Number of configuration bytes:** $T_{dc} = 30.10^3 \, bytes$
- **Access frequency of embedded SRAM memory:** $V_m = 300MHz$

\[
N_D = \left[ \frac{T_{dc} V_m}{T_i N_c} \right] = 5
\]
Dynamic implementation gain

<table>
<thead>
<tr>
<th>FONCTION</th>
<th>FIR</th>
<th>SEARCHER</th>
<th>RAKE RECEIVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST (LB)</td>
<td>1117</td>
<td>1235</td>
<td>245  50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total : 1030</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>3382 LBs</td>
</tr>
</tbody>
</table>

Synthesis results
- One CLB: 1539 $\mu m^2$
- One switch-box (SB): 11653 $\mu m^2$
eFPGA RESOURCE EXPLORATION 5/5

Dynamic implementation gain

<table>
<thead>
<tr>
<th>FONCTION</th>
<th>FIR</th>
<th>SEARCHER</th>
<th>RAKE RECEIVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>COST (LB)</td>
<td>1117</td>
<td>1235</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total : 1030</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>3382 LBs</td>
</tr>
</tbody>
</table>

Synthesis results

- One CLB: 1539 $\mu m^2$
- One switch-box (SB): 11653 $\mu m^2$

Static implementation

Static eFPGA (3382 LBs/SBs): 44 $mm^2$
**Dynamic implementation gain**

<table>
<thead>
<tr>
<th>Function</th>
<th>FIR</th>
<th>Searcher</th>
<th>Rake Receiver</th>
</tr>
</thead>
</table>
| Cost (LB)| 1117 | 1235     | Finger (LB)   | 245 | 50  
|          |      |          | Symbol (LB)   |      |     
|          |      |          | Total : 1030  |      |     

**Total**: 3382 LBs

**Synthesis results**

- One CLB: 1539 $\mu m^2$
- One switch-box (SB): 11653 $\mu m^2$

**Static implementation**

Static eFPGA (3382 LBs/SBs): 44 $mm^2$

**Dynamic implementation**

Dynamic eFPGA (1235 LBs/DBs/BUCKs): 26 $mm^2$
### Dynamic implementation gain

<table>
<thead>
<tr>
<th>Function</th>
<th>FIR</th>
<th>Searcher</th>
<th>Rake Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (LB)</td>
<td>1117</td>
<td>1235</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total : 1030</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>3382 LBs</td>
</tr>
</tbody>
</table>

#### Synthesis results
- One CLB: 1539 $\mu m^2$
- One *switch-box* (SB): 11653 $\mu m^2$

#### Static implementation
- Static eFPGA (3382 LBs/SBs): 44 $mm^2$

#### Dynamic implementation
- Dynamic eFPGA (1235 LBs/DBs/BUCKs): 26 $mm^2$

$\Rightarrow$ 40% less silicon area
### Number of configuration bits

<table>
<thead>
<tr>
<th>Reconfigurable Resource</th>
<th>Bits Nb /Resource</th>
<th>Resource Nb /DPR</th>
<th>Bits Nb /DPR</th>
<th>Bits Nb /Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGs</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>Reg</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>FU1&amp;3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>FU2&amp;4</td>
<td>11</td>
<td>2</td>
<td>22</td>
<td>132</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>228</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⇒ 912 bits (four clusters) + 1056 bits (interconnections) = 1968 bits = 246 bytes per configuration for DART

### Overhead of dynamic reconfiguration based on DUCK components

<table>
<thead>
<tr>
<th>DUCK Resource</th>
<th>Register /DUCK</th>
<th>Area in mm² /DUCK</th>
<th>Area ratio DUCK/RESOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FU1&amp;3</td>
<td>2</td>
<td>0.0008</td>
<td>1.08%</td>
</tr>
<tr>
<td>FU2&amp;4</td>
<td>6</td>
<td>0.0018</td>
<td>2.47%</td>
</tr>
</tbody>
</table>
DART RESOURCE EXPLORATION 3/3

How many domains \( (N_D) \) to fulfill the timing constraints?

- Number of context: \( N_c = 3 \) (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: \( T_i = 65 \mu s \)
- Access frequency of embedded SRAM memory: \( V_m = 300MHz \)
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
- Access frequency of embedded SRAM memory: $V_m = 300MHz$
- Number of configuration bytes: $T_{dc} = 246$ bytes
How many domains ($N_D$) to fulfill the timing constraints?

- Number of context: $N_c = 3$ (FIR filter, Searcher, Rake Receiver)
- Time for slot sample acquisition: $T_i = 65 \mu s$
- Access frequency of embedded SRAM memory: $V_m = 300 MHz$
- Number of configuration bytes: $T_{dc} = 246$ bytes

$$N_D = 1$$
CONCLUSIONS AND WORK IN PROGRESS

xMAML: a language for DRA modeling

- Generic specification, supports heterogeneity
  ⇒ from FPGA to coarse-grain reconfigurable (multi-)processors
- Rapid prototyping of DRA platforms
- Automatic generation of the (VHDL) RTL code for synthesis

Exploration results

- Dynamic implementation requires less area than static
- Overhead of reconfiguration management is limited

Work in progress

- VLSI design of the eFPGA ⇒ available Q2 2010
- Generation of optimized full-custom blocks (e.g. intercx resource) from xMAML
- Synthesis and P&R tools for bit-stream generation
Join us at ISCAS 2010 in Paris!

Deadline: october 9, 2009 ;−)