Dynamically Reconfigurable FIR Filter Architectures with Fast Reconfiguration

Martin Kumm, Konrad Möller and Peter Zipf

University of Kassel, Germany
FIR FILTER

- Fundamental component in digital signal processing
- Computationally complex due to numerous multiply/accumulate operations

![FIR Filter Diagram]
WHY RECONFIGURATION?

- Many applications require the change of coefficients...
- ...but only from time to time

⇒ Possibility to reduce complexity
METHODS OF RECONFIGURATION

1. Integrating multiplexers into the design
2. Partial reconfiguration (e.g., using ICAP)
3. Reconfigurable LUTs
MULTIPLEXER BASED RECONFIGURATION

- Multiplexers are integrated in add/shift networks
- Extremely fast reconfiguration (single clock cycle)
- Only a limited set of coefficients possible!

\[ x \cdot \{815,621,831,105\} \]

[Faust et al. ’10]
PARTIAL RECONFIGURATION

- Partial regions of the FPGA are reconfigured via ICAP

😊 Least resources

😊 Arbitrary coefficients...

😢 ... but synthesis needed for each coefficient set

😢 Slow reconfiguration ($\approx$μs/ms)!
RECONFIGURABLE LUTS

- Changing the LUT content only
- Routing has to be fixed
- First academic tool available (TLUT flow, [Bruneel et al. ’11])

😊 Fast reconfiguration (a few clock cycles, \(\approx\text{ns/\mu s}\))

😊 Arbitrary coefficients...

😊 ... but (again) synthesis needed for each coefficient set

↔ Not, if a generic architecture is transformed to fixed routing
RECONFIGURABLE LUTS

- FPGA components to realize reconfigurable LUTs
  - Older Xilinx FPGAs (Virtex 1-4): Shift-Register LUT (SRL16)
  - Newer Xilinx FPGAs (Virtex 5/6, Spartan 6, 7-Series): CFGLUT5 (similar to SRLC32E but with two output functions)
  - Other FPGA vendors: Distributed RAM or block RAM
METHODS OF RECONFIGURATION

1. Integrating multiplexers into the design
   ⇒ Logic fixed, routing flexible

2. Partial reconfiguration (e.g., using ICAP)
   ⇒ Logic flexible, routing flexible

3. Reconfigurable LUTs
   ⇒ Logic flexible, routing fixed
LUT BASED FIR FILTER

- Two well-known methods that employ LUTs in a fixed structure, suitable for FIR filters:

1. Distributed Arithmetic [Crosisier et al. ’73] [Zohar ’73] ...
   ... [Kumm et al. ’13]

2. LUT based multipliers [Chapman ’96] [Wiatr et al. ’01]
The main question is:

"Which architecture performs best?“
DISTRIBUTED ARITHMETIC

- Main idea is rearranging the underlying inner product

- Resulting function (realized as LUT) is identical for each bit $b$

$\Rightarrow$ Less configuration memory

$$y = c \cdot x = \sum_{n=0}^{N-1} c_n x_n$$

$$= \sum_{n=0}^{N-1} c_n \sum_{b=0}^{B_x-1} 2^b x_{n,b}$$

$$= \sum_{b=0}^{B_x-1} 2^b \sum_{n=0}^{N-1} c_n x_{n,b}$$

$$= f(\tilde{x}_b^N) \text{ (LUT)}$$

$$\tilde{x}_b^N = (x_{0,b}, x_{1,b}, \ldots, x_{N-1,b})^T$$
DISTRIBUTED ARITHMETIC
OVERALL ARCHITECTURE

Pre-processing to exploit coefficient symmetry

Reconfigurable LUTs

Output adder tree

Reconfiguration circuit

Filter Select

Reconf. Circuit
DISTRIBUTED ARITHMETIC MAPPING TO CFGLUT5
**Basic Idea:** Split a multiplication into smaller chunks which fit into the FPGA LUT:

\[
\begin{align*}
\underbrace{c_n \cdot x_n}_{B_c \times B_x \text{ mult.}} &= \underbrace{c_n \sum_{b=0}^{L-1} 2^b x_{n,b} + 2^L}_{B_c \times L \text{ mult.}} \\
&\quad \underbrace{c_n \sum_{b=0}^{L-1} 2^b x_{n,b+L} + \ldots}_{B_c \times L \text{ mult.}}
\end{align*}
\]
LUT MULTIPLIER MAPPING TO CFGLUT5
LUT MULTIPLIER
OVERALL ARCHITECTURE

Replaced by reconfigurable multipliers
CONTROL ARCHITECTURE
## RESOURCE COMPARISON

<table>
<thead>
<tr>
<th>Distributed Arithmetic</th>
<th>LUT Multiplier FIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_x + 1$ LUTs with $M$ inputs</td>
<td>$M$ LUTs with $B_x$ inputs</td>
</tr>
<tr>
<td>CFGLUTs: $\frac{1}{4}(B_x + 1)M(B_c/2 + 1)$</td>
<td>CFGLUTs: $\frac{1}{4}B_x M(B_c/2 + 2)$</td>
</tr>
</tbody>
</table>

$M = \lceil N/2 \rceil$: No. of unique taps

$B_x / B_c$: input / coefficient bit width
## RESOURCE COMPARISON

<table>
<thead>
<tr>
<th>Distributed Arithmetic</th>
<th>LUT Multiplier FIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_x + 1$ LUTs with $M$ inputs</td>
<td>$M$ LUTs with $B_x$ inputs</td>
</tr>
</tbody>
</table>

CFGLUTs:

- Distributed Arithmetic:
  
  $\left( B_x + 1 \right) \left\lceil \frac{M}{4} \right\rceil \left\lceil \frac{B_c}{2} + 1 \right\rceil$

  $\approx \frac{1}{4} \left( B_x + 1 \right) M \left( \frac{B_c}{2} + 1 \right)$

- LUT Multiplier FIR:
  
  $M \left\lceil \frac{B_x}{4} \right\rceil \left\lceil \frac{B_c}{2} + 2 \right\rceil$

  $\approx \frac{1}{4} B_x M \left( \frac{B_c}{2} + 2 \right)$

Surprisingly, CFGLUT requirements are very similar!
**RESOURCE COMPARISON**

<table>
<thead>
<tr>
<th>Distributed Arithmetic</th>
<th>LUT Multiplier FIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adders:</td>
<td>Adders:</td>
</tr>
<tr>
<td>$M + B_x + (B_x + 1) \left\lceil \frac{M}{4} \right\rceil$</td>
<td>$2M - 1 + M \left\lfloor \frac{B_x}{4} \right\rfloor$</td>
</tr>
</tbody>
</table>

$\Rightarrow$ So, LUT multiplier based FIR filters are better when...

$$2M - 1 + MB_x/4 < M + B_x + (B_x + 1)\frac{M}{4}$$

$$\vdots$$

$$\frac{3}{4}M - 1 < B_x$$

...,i.e., the input word size $B_x$ is greater than approximately half the number of coefficients $M = \left\lfloor \frac{N}{2} \right\rfloor$
RESULTS: 1ST EXPERIMENT

- Synthesis experiment for Virtex 6
- Nine benchmark filters with length $N=6...151$
- Input word size $B_x \in \{8, 16, 24, 32\}$

⇒ Very fast reconfiguration times: 49...106 ns
⇒ High clock frequencies: 472 MHz / 494 MHz (DA/LUT mult.)
RESULTS: 1ST EXPERIMENT

LUT Multiplier improvement compared to DA:

As expected, the LUT multiplier architecture is best for low $N$
RESULTS: 1ST EXPERIMENT

LUT Multiplier improvement compared to DA:

Choosing the right architecture can save up to 40% slices
RESULTS: 2ND EXPERIMENT

- Comparison with partial reconfiguration via ICAP

- Ten different filters with $N=41$ were highly optimized using PMCM optimization RPAG [Kumm et al. ’12]

<table>
<thead>
<tr>
<th>Method</th>
<th>$S$ [bit]</th>
<th>Slices</th>
<th>$f_{clk}$ [MHz]</th>
<th>$T_{rec}$ [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPAG with ICAP</td>
<td>746496</td>
<td>502…569</td>
<td>386.7…448.8</td>
<td>233280</td>
</tr>
<tr>
<td>Reconf. FIR DA</td>
<td>1920</td>
<td>1071</td>
<td>521.9</td>
<td>61.3</td>
</tr>
<tr>
<td>Reconf. FIR LUT</td>
<td>14784</td>
<td>1108</td>
<td>487.8</td>
<td>65.6</td>
</tr>
</tbody>
</table>

Configuration memory is reduced by a factor of $1/388$ (DA) and $1/50$ (LUT Mult.) 😊
RESULTS: 2ND EXPERIMENT

- Comparison with partial reconfiguration via ICAP
- Ten different filters with $N=41$ were highly optimized using PMCM optimization RPAG [Kumm et al. ’12]

<table>
<thead>
<tr>
<th>Method</th>
<th>$S$ [bit]</th>
<th>Slices</th>
<th>$f_{clk}$ [MHz]</th>
<th>$T_{rec}$ [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPAG with ICAP</td>
<td>746496</td>
<td>502...569</td>
<td>386.7...448.8</td>
<td>233280</td>
</tr>
<tr>
<td>Reconf. FIR DA</td>
<td>1920</td>
<td>1071</td>
<td>521.9</td>
<td>61.3</td>
</tr>
<tr>
<td>Reconf. FIR LUT</td>
<td>14784</td>
<td>1108</td>
<td>487.8</td>
<td>65.6</td>
</tr>
</tbody>
</table>

Slice requirements are roughly doubled 😞
RESULTS: 2ND EXPERIMENT

- Comparison with partial reconfiguration via ICAP

- Ten different filters with $N=41$ were highly optimized using PMCM optimization RPAG [Kumm et al. ’12]

<table>
<thead>
<tr>
<th>Method</th>
<th>$S$ [bit]</th>
<th>Slices</th>
<th>$f_{clk}$ [MHz]</th>
<th>$T_{rec}$ [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPAG with ICAP</td>
<td>746496</td>
<td>502...569</td>
<td>386.7...448.8</td>
<td>233280</td>
</tr>
<tr>
<td>Reconf. FIR DA</td>
<td>1920</td>
<td>1071</td>
<td>521.9</td>
<td>61.3</td>
</tr>
<tr>
<td>Reconf. FIR LUT</td>
<td>14784</td>
<td>1108</td>
<td>487.8</td>
<td>65.6</td>
</tr>
</tbody>
</table>

Performance is similar
RESULTS: 2ND EXPERIMENT

- Comparison with partial reconfiguration via ICAP
- Ten different filters with $N=41$ were highly optimized using PMCM optimization RPAG [Kumm et al. ’12]

<table>
<thead>
<tr>
<th>Method</th>
<th>$S$ [bit]</th>
<th>Slices</th>
<th>$f_{clk}$ [MHz]</th>
<th>$T_{rec}$ [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPAG with ICAP</td>
<td>746496</td>
<td>502…569</td>
<td>386.7…448.8</td>
<td>233280</td>
</tr>
<tr>
<td>Reconf. FIR DA</td>
<td>1920</td>
<td>1071</td>
<td>521.9</td>
<td>61.3</td>
</tr>
<tr>
<td>Reconf. FIR LUT</td>
<td>14784</td>
<td>1108</td>
<td>487.8</td>
<td>65.6</td>
</tr>
</tbody>
</table>

Reconfiguration time is drastically reduced by a factor of $1/3556$! 😊
CONCLUSION

- Two different reconfigurable FIR filter architectures for arbitrary coefficient sets were analyzed.

- Both are implemented using reconfigurable LUTs (CFGLUTs).

- The LUT multiplier architecture typically needs less slices when input word size is greater than approx. half the number of coefficients (and vice versa).

- Both architectures offer reconfiguration times of about 3500 times faster than partial reconfiguration using ICAP.

- This is paid by twice the number of slice resources.
If you have a reconfigurable FPGA circuit which allows a fixed routing:

Use reconfigurable LUTs!
THANK YOU!