Customized Exposed Datapath Soft-Core Design Flow with Compiler Support

Otto Esko*, Pekka Jääskeläinen*, Pablo Huerta¤,
Carlos Sanches De La Lama¤, Jarmo Takala*
and Jose Ignacio Martinez¤

* Tampere University of Technology, Finland
¤ Universidad Rey Juan Carlos, Madrid, Spain

Motivation

- Popular way of using high level language programming on FPGA designs is to use soft-core processor
  - No manual RTL coding
  - Fast to implement
- However scalability in the current soft-cores is limited
- We propose the performance can be scaled by using a customizable parallel architecture to exploit the available ILP
  - Without any RTL coding

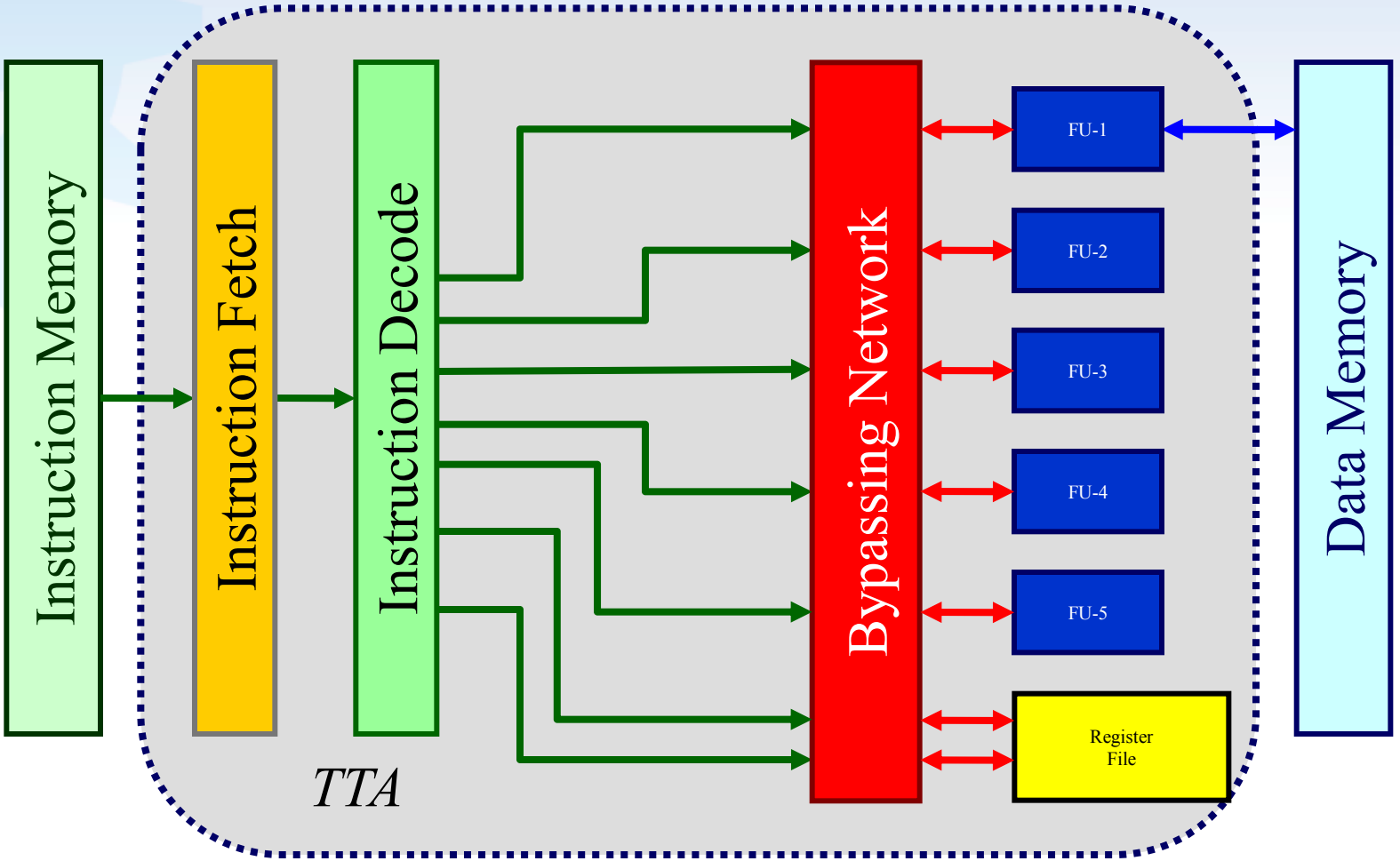
Outline

- Transport Triggered Architecture
- TTA-based Codesign Environment
- Design Flow
- Custom Operation Design Flow
- Benchmark Suite
- Results
- Conclusion

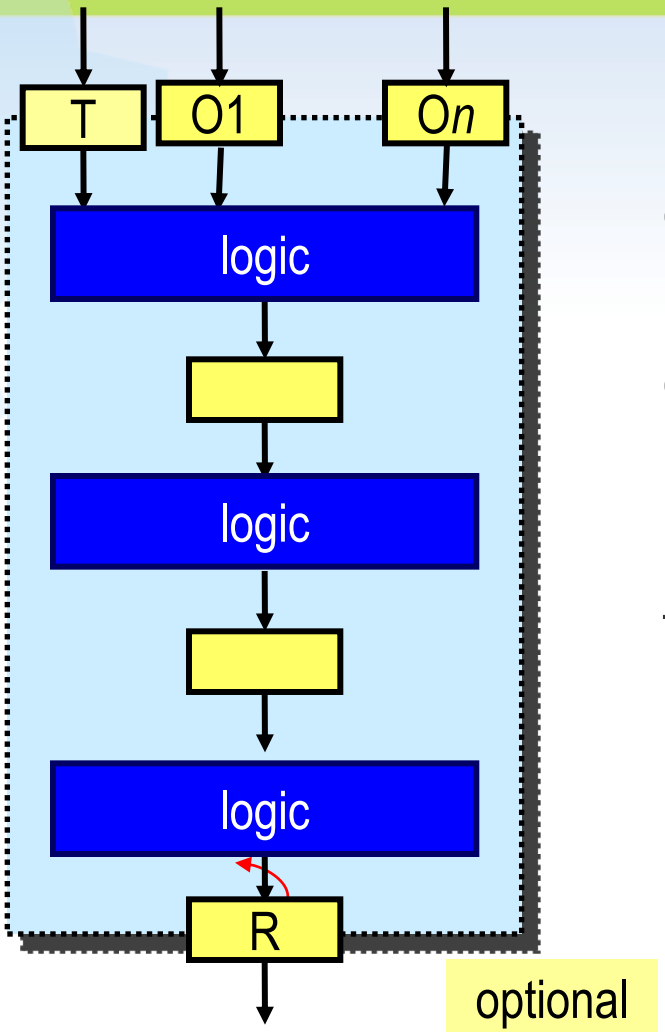
VLIW

Instruction Memory
Instruction Fetch
Instruction Decode
FU-1
FU-2
FU-3
FU-4
FU-5
Bypassing Network
Register File
Data Memory
VLIW

TTA

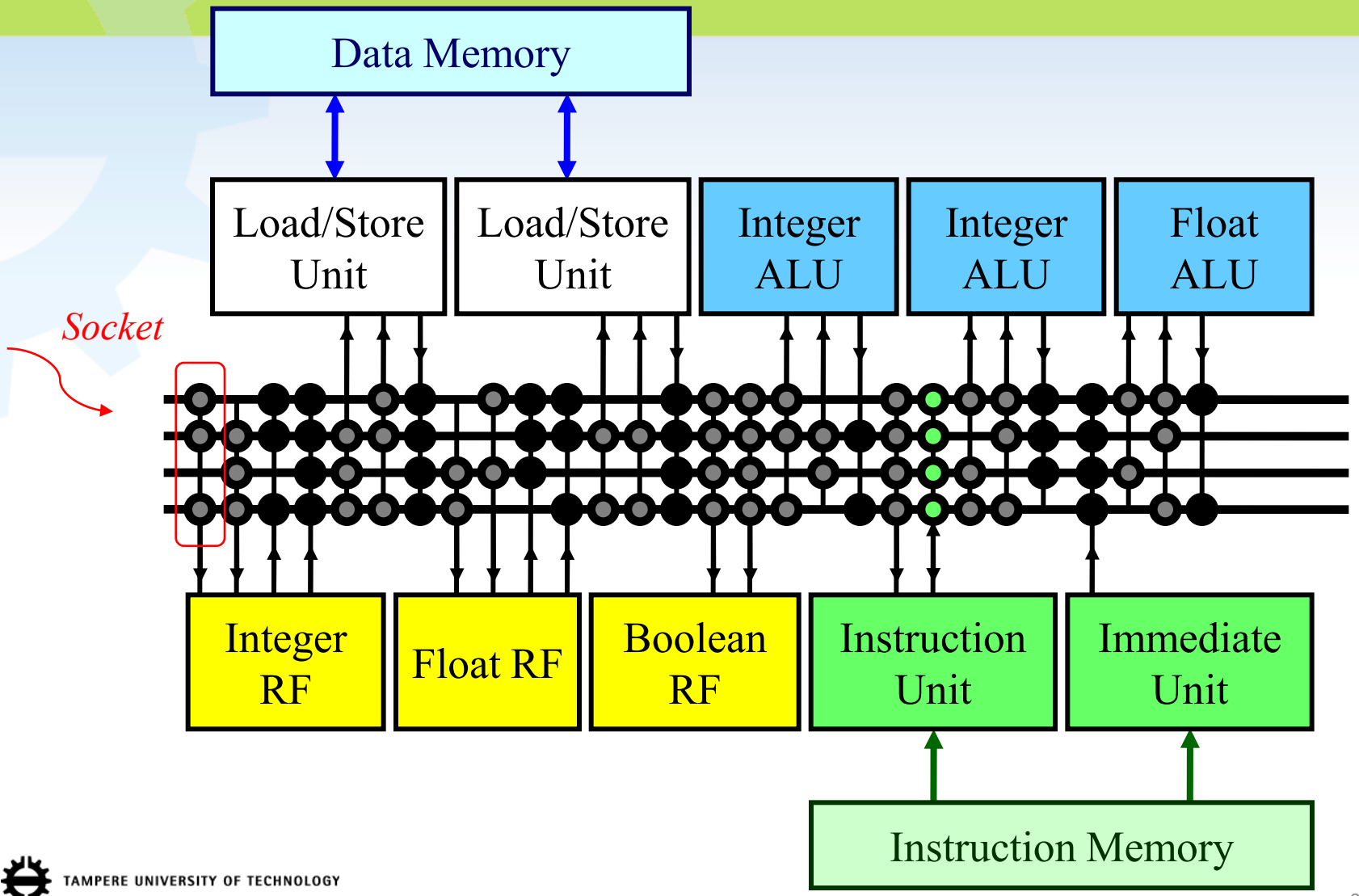

Instruction Memory
Instruction Fetch
Instruction Decode
Bypassing Network
FU-1
FU-2
FU-3
FU-4
FU-5
Register File
Data Memory
TTA

TTA Function Unit



- Operation executed as side effect of operand transports
- Operands are written to operand registers (O)
- Operation performed when last operand written to trigger register (T)
- FUs can be fully pipelined

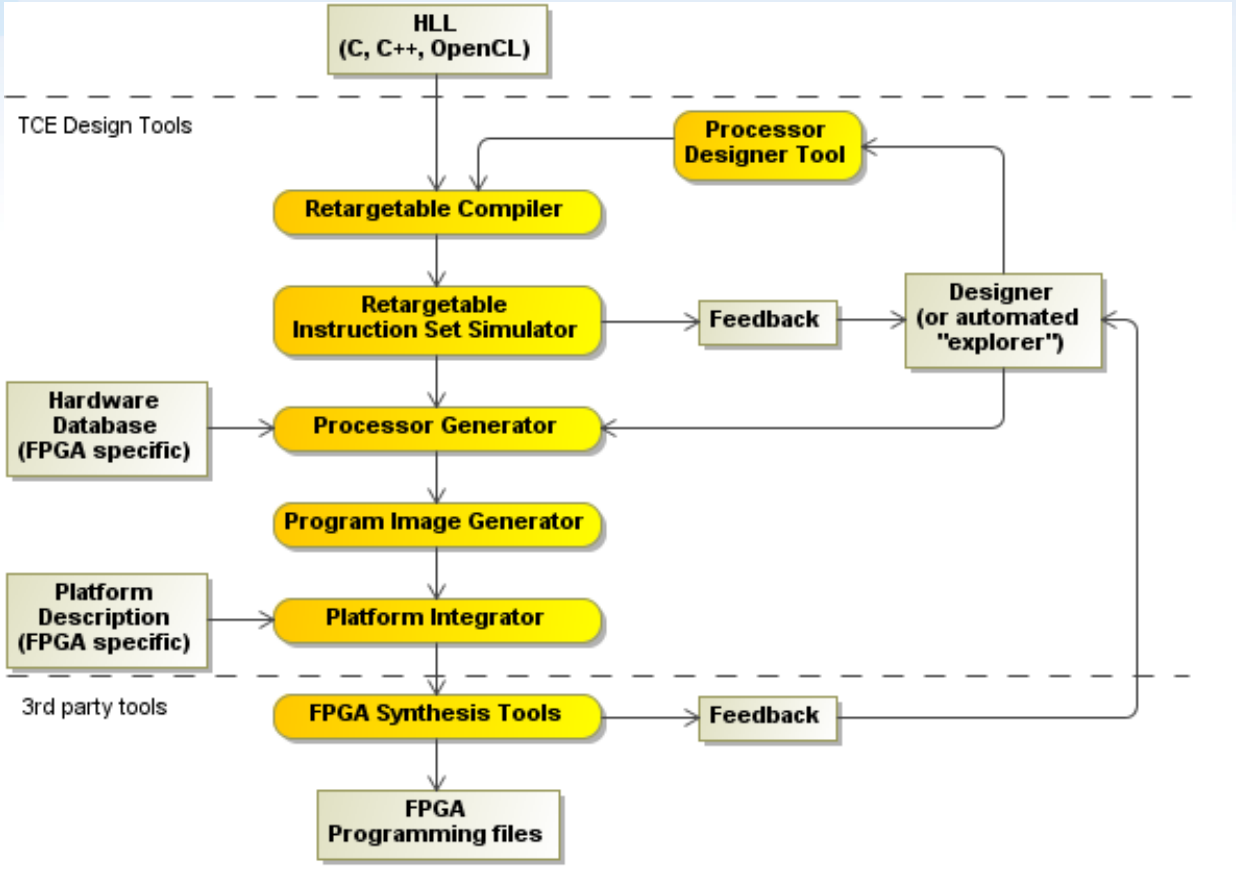
Example TTA Datapath



TTA-based Codesign Environment (TCE)

- TCE is a toolset for designing application specific processors based on TTA processor template
  - Main use case: Fast co-design of processor based accelerators without manual VHDL coding
  - HLL to RTL-flow
- Processor architecture and implementation are separeted in the design process
- Retargetability

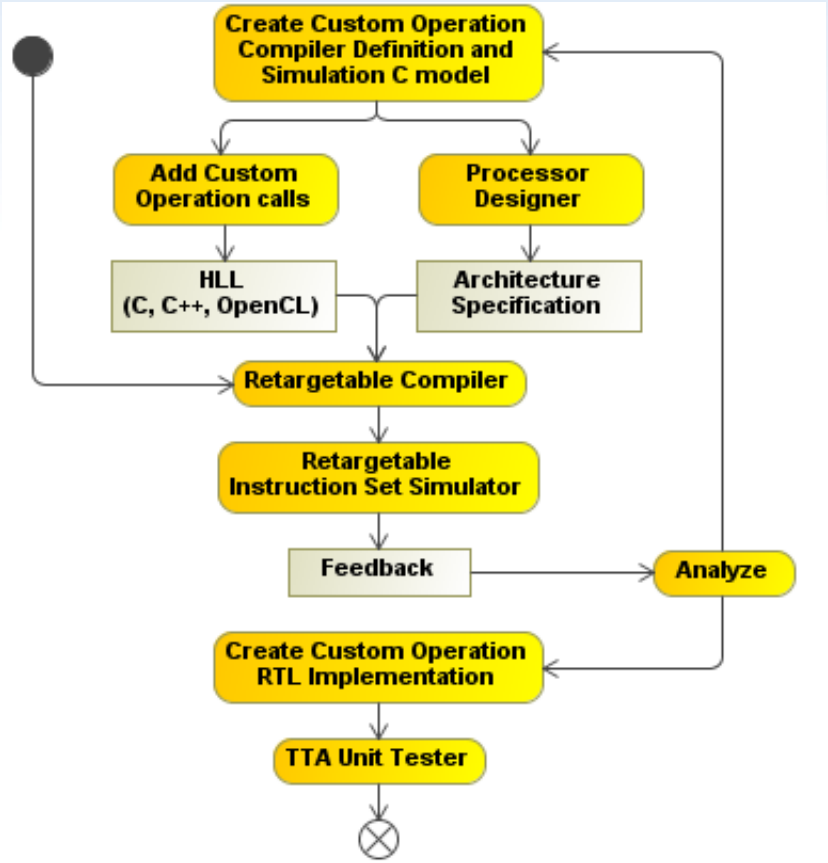
TCE Design Flow



Custom Operations

- TCE allows user defined MIMO custom operations
- Can speed up the application algorithm significantly
- Custom operations can be tested and evaluated without RTL implementation
  - Behavioral simulation model is needed (C/C++)

Custom Operation Design Flow



Example: Simulation C model

Original C code:

```
uint32
reflect(uint32 data, uint8 nBits) {
  uint32 refl = 0;
  uint8 bit = 0;
  for(bit = 0; bit < nBits;++bit){
    if(data & 0x01) {
      refl |= (1 << ((nBits-1)-bit));
    }
    data = (data >> 1);
  }
  return refl;
}
```

Behav. simulation model:

```
uint32 data = UINT(1);
uint8 nBits = UINT(2);
uint32 refl = 0;
uint8 bit = 0;
for(bit = 0; bit < nBits;++bit){
  if(data & 0x01) {
    refl |= (1 << ((nBits-1)-bit));
  }
  data = (data >> 1);
}
IO(3) = refl;
```

Example: Custom operation usage

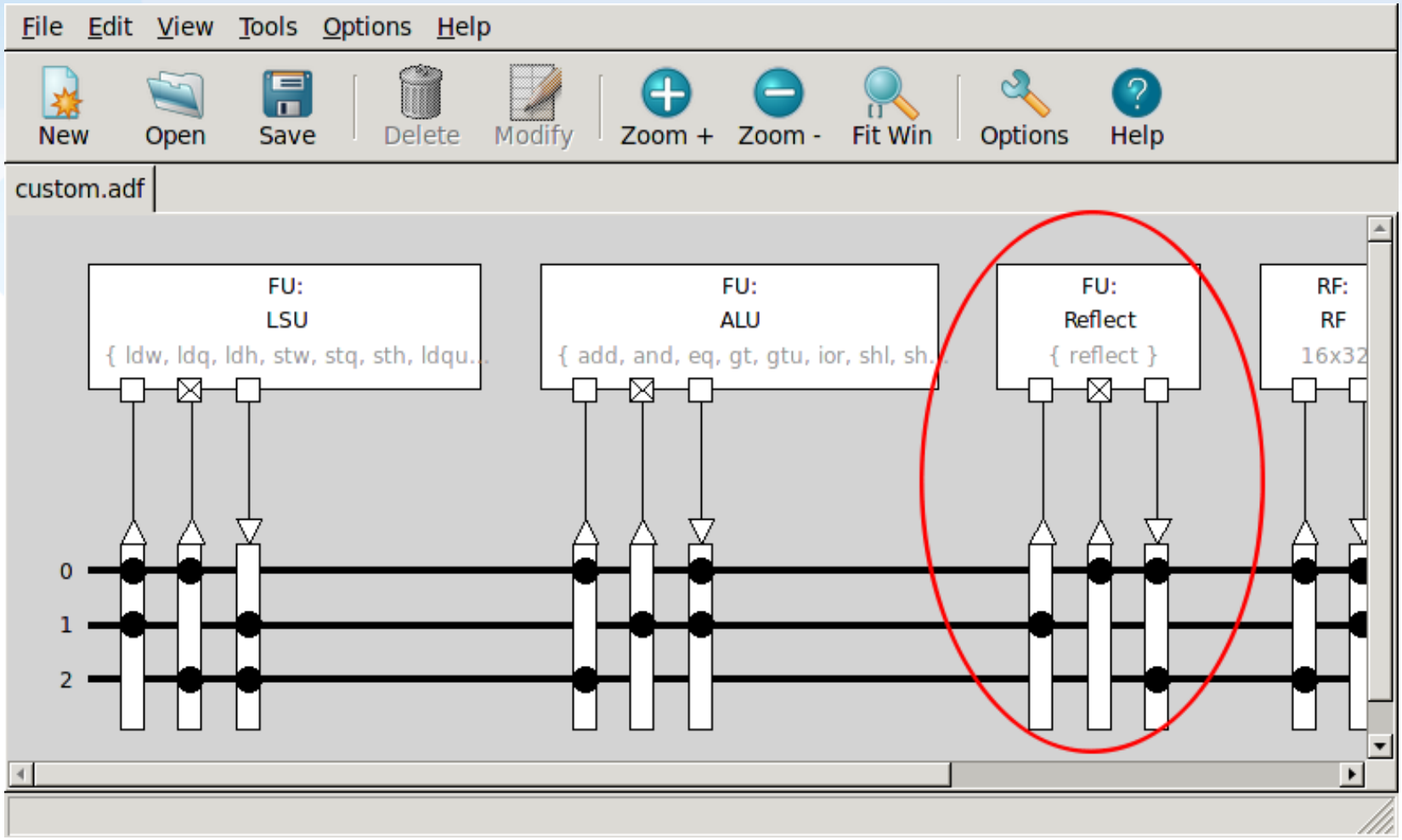
Original C code:

```
uint32 result = 0;
result = reflect(data, 8);
```

Custom operation call:

```
uint32 result = 0;
_TCE_REFLECT(
        data, 8, result);
```

Example: Architecture description



CHStone benchmark suite

- CHStone is a benchmark suite targeted for evaluating High Level Synthesis tools
- Benchmark programs are written in C
- Consists of
  - Arithmetic programs
  - Media applications
  - Cryptography programs
  - Processor emulation
- Seven test programs were used in evaluation

TTA used in evaluation

- One TTA architecture was created through design iterations
  - Various different architectures were tested
  - Processor resources were scaled
  - Interconnection was optimized
- Tradeoffs between
  - Cycle count – FPGA resource usage – Max clock frequency
- Design work was done at architecture level
  - All resources were taken from standard hardware library
  - Custom operations were not used
    - No RTL coding was needed
- Final processor architecture had
  - 3 ALUs, 1 Multiplier and 1 Load Store Unit
  - 3 register files (14 registers in each)
  - 1 fully connected and 17 partially connected transport buses

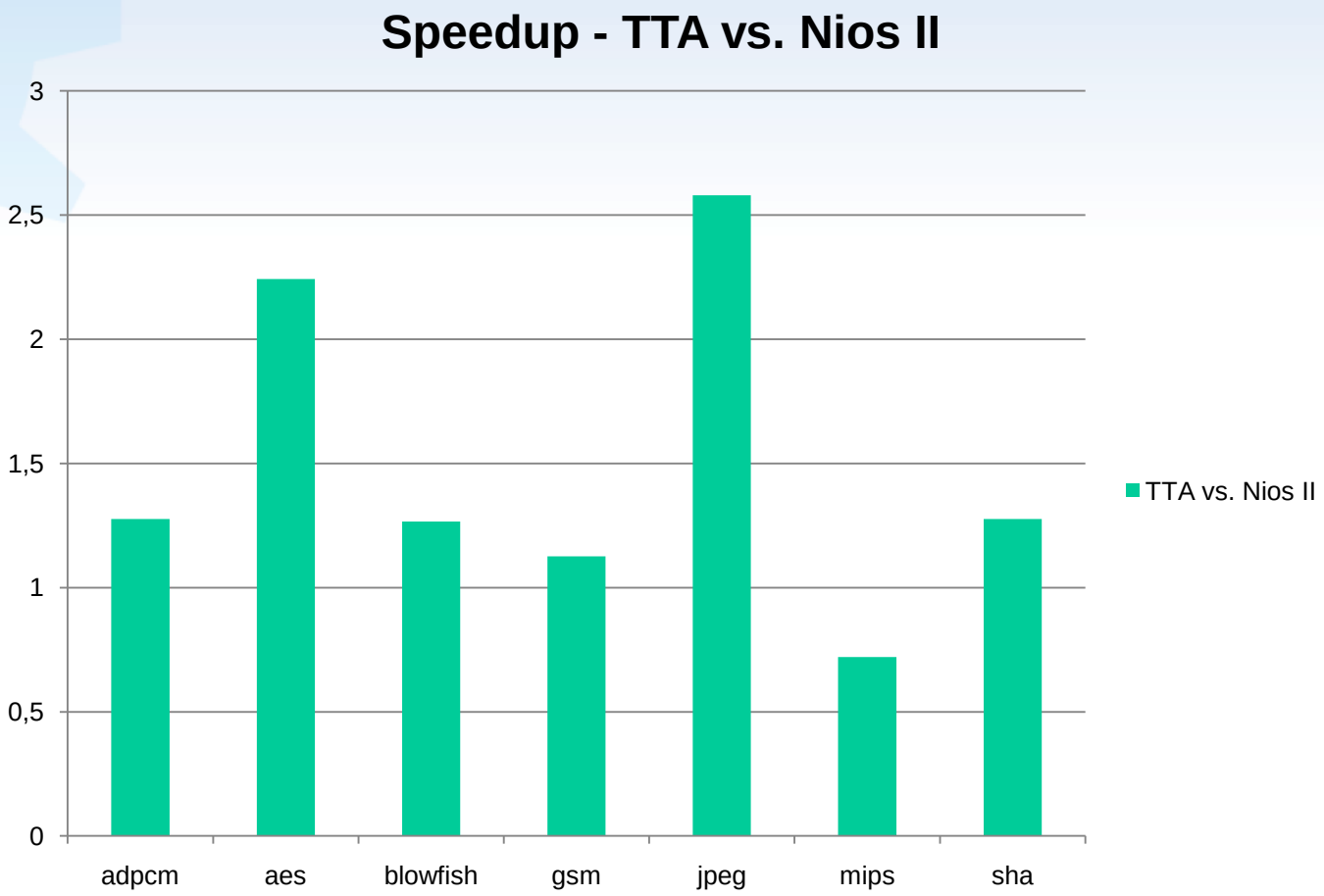
Tested architectures

- Hardware multipliers were used on all targets
- FPGAs' internal memory was used on all targets
- TTA
  - Synthesized on Altera Stratix II and Xilinx Virtex 5
- Nios II
  - Fastest Nios II/f configuration was used
  - Synthesized on Altera Startix II
- MicroBlaze
  - Both 3- and 5-stage pipeline version were used
  - Synthesized on Xilinx Virtex 5

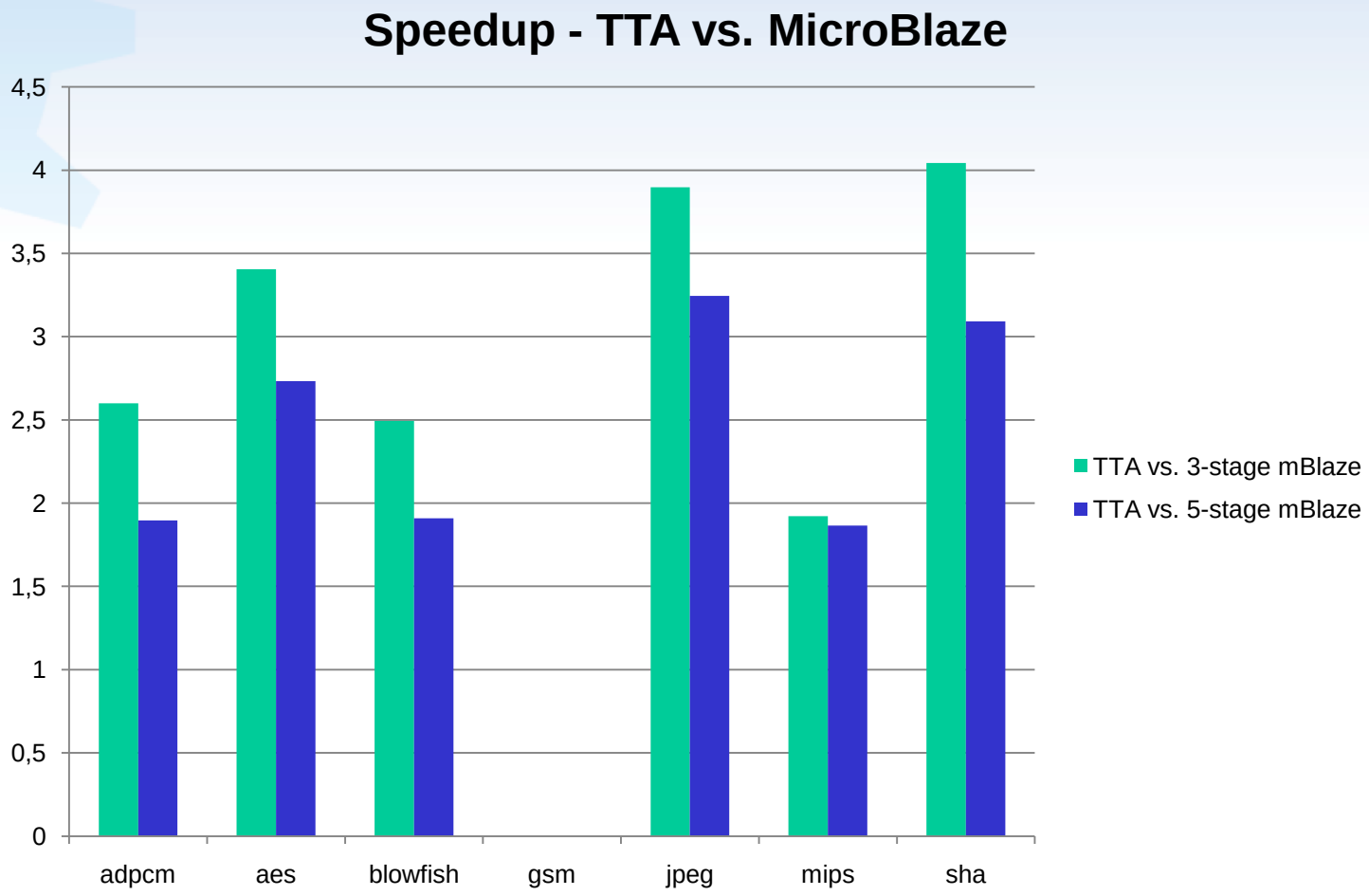
Results – Max clock frequency

	Stratix II		Virtex 5		
	TTA	Nios II	TTA	mBlaze 3-stage	mBlaze 5-stage
f_{max} / MHz	149	175	191	169	195

Speedup: TTA vs. Nios II/f



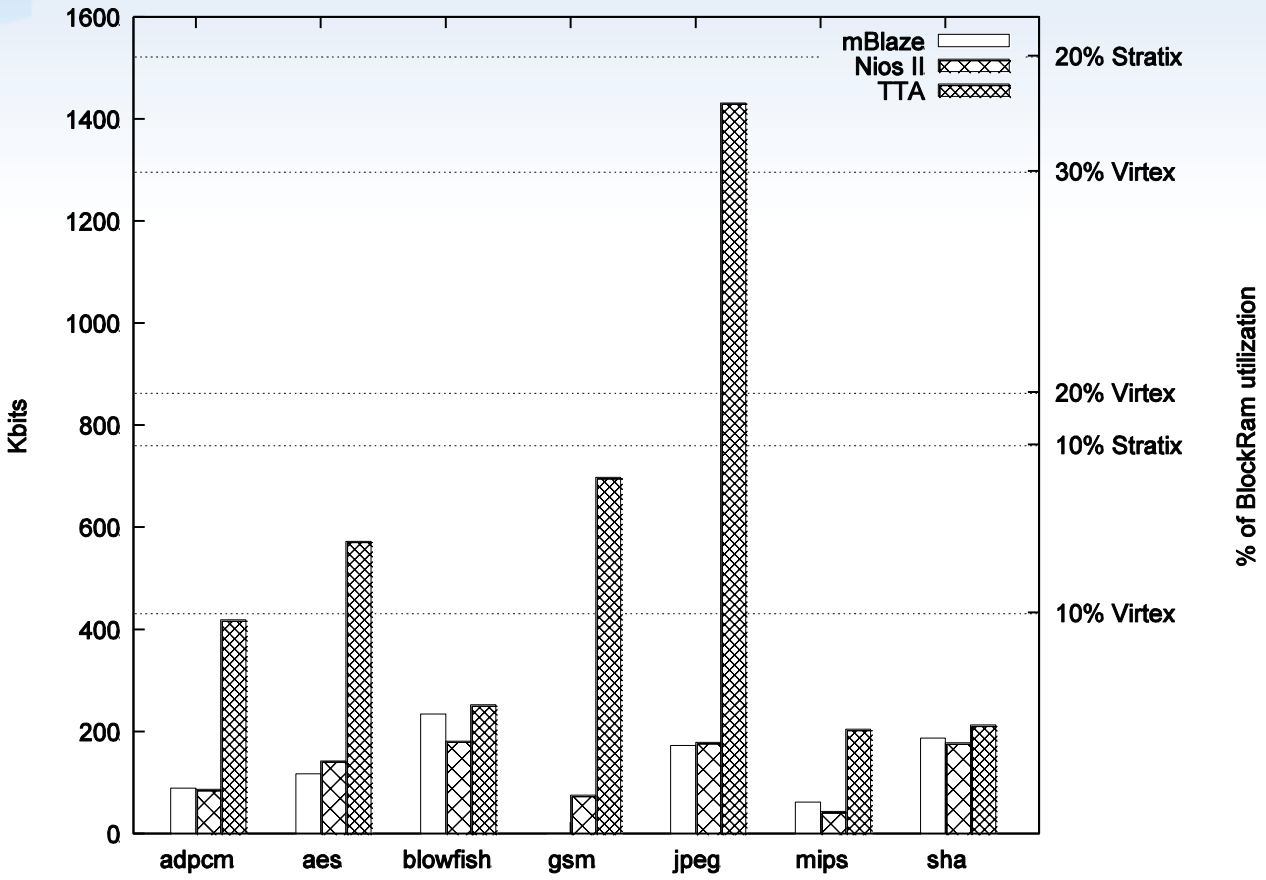
Speedup: TTA vs. MicroBlaze



Results - Area

		Stratix II		Virtex 5		
		TTA	Nios II	TTA	mBlaze 3-stage	mBlaze 5-stage
LUTs	#	5 218	2 322	5 024	1 537	1 899
	% of all	3.6 %	1.6 %	7.3 %	2.2 %	2.7 %
Registers	#	2 785	1 890	3 485	1 318	1 841
	% of all	1.9 %	1.3 %	5.0 %	1.9 %	2.7 %

Results – Instruction memory usage



Conclusion

- TTA template provides high level of customization
  - Processor resources
  - Datapath connectivity
- TCE provides tools for the proposed customizable soft-core design flow
  - Runtime retargetability
  - Released as open source
    - Available at http://tce.cs.tut.fi
- TTA can be used to scale performance
  - when ILP is available
  - at the expense of FPGA resource and memory usage
- Custom operations could be used to increase performance even further
  - Requires manual RTL coding