T42 Transputer-in-FPGA


APB 1096 - 2:50pm-4:20pm
Speaker: Dipl.-Ing. Uwe Mielke
Preface

A project report or review from time to time is an excellent tool to sort results and organize (new) ideas.

My T42 design work started spring 2014, after one year of investigations and preparations. My Dad (now 88 year old), who is an electronic engineer as well, was interested in the design progress all the time and pushed me to go ahead and to follow my intensions. Thank you very much Dad!

Today I've reminded myself how much time the T42 design project has already taken: best guess more than 300 days (1.5MY “in parallel”) within 3.5 years of working in a normal full time job.

That gives a good reason to spend the leftover effort and time needed to make the T42 Transputer-in-FPGA finally a really success!


Uwe Mielke
Agenda

1. Motivation, Preparations
2. T425 Specification (Target)
3. Design Considerations
4. Micro Code, Assembler
5. Design Partitioning
   a. Data Path
   b. Control Path
   c. Links
   d. Memory Hierarchy
   e. System Control
6. Verification, Results
7. Summary, Conclusions, Prospects
1.0 Motivation

Why to reverse engineer a chip from the past?

The **Transputer** was an extraordinary chip ahead of its time…

Picture: AVM ISDN Controller B1 (1990) presented by Peter Faxel (CTO)

This worldwide most often sold ISDN card design was manufactured by AVM in different versions until 2015.
1.1 Motivation

Why to reverse engineer a chip from the past?

The Transputer was an extraordinary chip ahead of its time…

many of them could be connected to solve a problem together:

- he introduced the idea of a communicating computer
- the first one enabling easy parallel processing (occam)
- based on excellent mathematical foundations (CSP)
- language and processor architecture designed together
- the first one with fast plug’n play serial links (4x 20Mbd)
- the first one w/ large & fast on-chip memory (4kByte)
- micro code was used to integrate process scheduling (OS like)

Being the big hope of Europe’s industry during the ESPRIT Program 1990+ …
Here is the Beauty...

Technology: 1.2µm CMOS
Transistors: 200,000
MIPS@Clock: 15MIPS@20MHz
Chip Size: 8.5 x 8.5 mm²
Power Supply: +5V ±5%
Packaging: CPGA 84
Production: 1989-1999
Price (1990): ~300 DM
Sales: > 1 Mio pc p.a.
1.2 Applications

1996: the HETE-2 spacecraft used 4x T805 Transputers and 8x DSP56001 yielding about 100 MIPS of performance and … redundancy

1992: Rank 259 on Top-500 List of Super Computers: 1024 CPU’s providing 4,5 GFLOP/s


HETE = High Energy Transient Explorer (UV, X-ray, gamma-ray)
1.3 Preparations … Literature

**web statistics**
- Google about „Transputer Architecture“ → more than 300,000 hits!
- over 500 Inmos patents (1978-94) → ~50 about micro processor(s)
- Inmos T425 specification(s) → 7 releases until 1996

**my main literature** (for more links please see appendix):
- [www.transputer.net](http://www.transputer.net) (Michael Bruestle)
- „Transputer-Leitfaden“ (H. Reinecke, J. Schreiner)
- The-Simple-42 (David May, Homepage, released Jan.2013)
- 5 Inmos patents from 1984-1993
- Inmos IMS T425 Specification (42-1426-07) Feb.96
- original T425 development document from 1988, received from former Inmos designer (Roger Shepherd, Sep.2015)
1.4 The Simple 42

- a 16 bit micro processor test chip … processed by Inmos in 1982
- the whole basic functionality of the Transputer was implemented:
  - 3 register stack architecture
  - operand register
  - work space pointer
  - instruction pointer
- micro code with 7 bit address (128 uWords)
- 46 instructions *) … all primary and some secondary (+ comm‘s)

docu. from Transputer architect David May: http://www.cs.bris.ac.uk/~dave/transputer.html
investigation from Gavin Crate: https://sites.google.com/site/transputeremulator/Home/inmos-s42

*) e.g.: J, LDL, PFIX, NFIX, REV, GT, AND, OR, XOR, CLC, STAC, ADD, ADDC, SUB, SUBC, MUL, UMUL, DIV, UDIV, SEX, TLNG, LB, SL, SR, …
S42 (1982)

Note:
Inmos used the S42 basic architecture & instruction set principles for the T414 (1984) & T425 (1989) design(s).

Source (Gavin Crate):
https://sites.google.com/site/transputeremulator/Home/inmos-s42
2.0 T425 Specification

The CPU contains:
- sequential 32bit Integer Processor
- Timers & (micro-coded) Process Scheduler
- Event Logic

Processor Registers:
- Evaluation Stack (RPN): Areg, Breg, Creg
- Operand Register: Oreg
- Workspace Pointer: Wptr
- Instruction Pointer: Iptr
- Flags: Error, HaltOnError, BreakEnable
- Internal Registers: Dreg, Ereg, StatusReg

Reverse Polish Notation

P.S.: this page shows the programmers' view on a Transputer.
2.0 T425 Specification

The CPU contains:
- Sequential 32-bit Integer Processor
- Timers & (micro-coded) Process Scheduler
- Event Logic

Processor Registers:
- Evaluation Stack (RPN): Areg, Breg, Creg
- Operand Register: Oreg
- Workspace Pointer: Wptr
- Instruction Pointer: Iptr
- Flags: Error, HaltOnError, BreakEnable
- Internal Registers: Dreg, Ereg, StatusReg

Scheduler and Timer Registers
- Front- and Back-Pointers of high and low priority process queues: FptrX, BptrX (X = Prio 0,1)
- Timer Counter (actual) and Timer Next Event Registers for high and low priority process queues: ClockRegX, TNextX.
- Timer Queue Pointers: TPtrLocX (* in Memory)
2.1 T425 Architecture Details

- stack machine w/ 3 register stack (Areg, Breg, Creg)
- workspace = virtual register bank (4kB = 1000 reg's)
- on-chip RAM access in 1 clock = register quality
- 2 process queues (prio’s)
- 2 timer queues (prio’s)
- low prio processes run w/ time slicing (1ms)
- high prio processes always run to end (= „interrupts“)
- native support for (occam) process model, no shared variables, communicating processes instead (CSP)
- process communication external via 4 serial links
2.2 T425 Instruction Set

- ISA: RISK like but CISC too
- one byte = one instruction
  - i.e. zero…one address machine
- 31 primary instructions
  - there are 15 with an 4bit operand
  - operand is either address or data
- 103 secondary instructions
  - 1-3 operands are in register stack
- operand addressing
  - direct (LDC, ADC, ADC)
  - indirect (LDLP, LDNLP)
  - relative (LDL…STNL, J, CJ, LEND)
  - register (e.g. all arithmetic logic)

Overview:
- Primary Function Codes (16)
- Processor initialisation operation codes (10)
- Arithmetic/logical operation codes (17)
- Long arithmetic operation codes (9)
- General operation codes (8)
- Indexing/array operation codes (8)
- Timer handling operation codes (6)
- Input/output operation codes (12)
- Control operation codes (5)
- Scheduling operation codes (5)
- Error handling operation codes (8)
- Debugger support codes (9)
- 2D block move operation codes (4)
- CRC and bit operation codes (5)
- Floating point support operation codes (6)
- processor test instructions (7)
3.0 Design Considerations

FPGAs …
• …like pipelined architectures 😊
• dual ported RAMs are for free → Harvard Architecture
• LUT-RAMs can be utilized up to maximum
• ALU must be very (simple and) fast
• T42 critical path may be long … but will ease micro code
  (e.g.: register → bus-mux → ALU → Zbus → register = 6.794ns @ Spartan-6)

HW vs SW (micro code)
• MUL implementation by Booth‘s algorithm (hardware multipliers may be used in a later design)
• DIV implementation by SRT algorithm
3.1 Design Targets for T42

- **binary compatibility** with the original Inmos T425 CPU
- **main difficulties:**
  1. reverse engineering of the Transputer's processor architecture
  2. retrieval and understanding of the micro code
  3. connecting to modern (DDR) memory of today ...

- execution **performance** is not a (main) design goal
  ... but can be taken just „on the way“ whenever easy achievable ...

- a must: **pipelined architecture** → decision: 2-stage-pipe
  pre-fetch … is an autonomous FSM … not a pipeline stage
  
  1. **IF/ID** instruction fetch & decode
  2. **EX** execute (using a single or multiple clocks per instruction)
  memory read/write is part of execute … not a pipeline stage

- **many T42 cores** should run on any cheap student **FPGA** board!
3.2 Design Guidelines

A list of some of my design project experiences after 3.5 years:

- follow good VHDL style
  - procedural.vhd: here are all the glue logic and processes
  - structural.vhd: wiring only, no logic or process allowed
  - constants_package.vhd: all constants definition
  - functions_package.vhd: all functions for general support
- keep track of changes
  - put extensive comments within VHDL code as much as possible
  - maintain a detailed change log
  - make use of revision control … use a repository (SVN) … now!
- document whatever seems finished
  - report your status frequently … this helps to find (design) gaps!
- fix meta value issues (e.g. ‘X’) in VHDL, e.g. detected by simulation
4.0 Micro Code ROM

- principal schematic of a micro coded machine (CPU):

- T42 µCode ROM
  1st size (2014) 512x96bit
- T42 today (2017) 1024x128bit
- T425 µCROM is 742x118bit (~90kBit)
  investigation done in Nov.2016 by G. Crate

*MIR = Micro Instruction Register, holds uCROM entry address
CMUX = Condition Multiplexor, 2 of them allow a decision of 1 out of 4 cases

--- Ctrl Path -------------------> <------ Data Path ---

SystemControl will decide about next eAddr source

* MIR = Micro Instruction Register, holds uCROM entry address
CMUX = Condition Multiplexor, 2 of them allow a decision of 1 out of 4 cases
4.1 Micro Word

- micro word consist of micro operations
- \( \mu \text{Op} = \text{bit group} \) \( \rightarrow \) HW control \( \ldots \) e.g. MUX

```
00 0'
01 Breg_i(MSB)
10 Creg_i(0)
11 '1'
```

- easy to be defined by only 2 VHDL files
(many thanks to Martin Zabel) 😊

1. t42_cpu_constpkg.vhd → bit coding of \( \mu \text{Ops} \)
2. t42_cpu_ucode_rom.vhd → bit order in \( \mu \text{Word} \)

P.S.: shown \( \mu \text{Word} \) example is from 512x96bit →
4.2 Micro Code Assembler

- μCAsm Input is *.csv; generated out of an Excel Sheet (µOp table)
- micro code design: based on mnemonics, defined in const_pkg.vhd
- micro code assembler: a simple BAT + AWK script system …

1.1 PRE PROCESSING - READ BIT POSITIONS OF MICRO-OP's
1.2 PRE PROCESSING - READ MICRO-OP IDs AND CODING
2.1 PRIMARY PROCESSING - ASSEMBLE MICROWORDs
2.2 SECONDARY PROCESSING - SORT MICROWORD COLUMNs
2.3 SECONDARY PROCESSING - SORT MICROWORD ROWs
2.4 SECONDARY PROCESSING - ALLOCATE FIXED ADDRESSES
2.5 SECONDARY PROCESSING - TABULATE BRANCH CAPABILITIES
2.6 SECONDARY PROCESSING - ALLOCATE JUMP+BRANCH LABELs
2.5 SECONDARY PROCESSING - CALCULATE ROMFEEDBAK ADDR
3.1 POST PROCESSING - BUILD ROM (BINARY FORMAT)
3.2 POST PROCESSING - WRITE HEX ROM
3.3 POST PROCESSING - BUILD uCodeROM … CALL XILINX DATA2MEM
## 5.0 Design Partitioning

<table>
<thead>
<tr>
<th>Ctrl2Data (structural)</th>
<th>LinkPath:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• A B C D E F r e g•A L U</td>
<td>• Sync, ChIn, ChOut, ChEvent, If’s</td>
</tr>
<tr>
<td>• W p t r</td>
<td>• Pointers</td>
</tr>
<tr>
<td>• Pointers</td>
<td>• ConstBox</td>
</tr>
<tr>
<td>• ByteAlign</td>
<td>• MemPath:</td>
</tr>
<tr>
<td></td>
<td>• MemIF</td>
</tr>
<tr>
<td></td>
<td>• MemMain</td>
</tr>
<tr>
<td></td>
<td>(dpram2kx32)</td>
</tr>
<tr>
<td></td>
<td>preliminary...</td>
</tr>
<tr>
<td></td>
<td>...instead of cache</td>
</tr>
<tr>
<td></td>
<td>• DummyCache</td>
</tr>
<tr>
<td></td>
<td>(dpram2kx32)</td>
</tr>
<tr>
<td></td>
<td>available +tested:</td>
</tr>
<tr>
<td></td>
<td>• CacheCtrl (TUD)</td>
</tr>
<tr>
<td></td>
<td>• DDRCtrl (TUD)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>MemPath:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• MemIF</td>
</tr>
<tr>
<td></td>
<td>• MemMain</td>
</tr>
<tr>
<td></td>
<td>(dpram2kx32)</td>
</tr>
<tr>
<td></td>
<td>preliminary...</td>
</tr>
<tr>
<td></td>
<td>...instead of cache</td>
</tr>
<tr>
<td></td>
<td>• DummyCache</td>
</tr>
<tr>
<td></td>
<td>(dpram2kx32)</td>
</tr>
<tr>
<td></td>
<td>available +tested:</td>
</tr>
<tr>
<td></td>
<td>• CacheCtrl (TUD)</td>
</tr>
<tr>
<td></td>
<td>• DDRCtrl (TUD)</td>
</tr>
</tbody>
</table>

### SysPath:
- SysCtrl, Sbits, Timer, **SysService**

### CtrlPath (structural):
- uCodeROM
- Idecode
- Oreg *(pipe)*
- Iptr *(+Inc)*
- Prefetch

### DataPath:
- ABCDEFreg
- ALU *X+Y=Z*
- Wptr
- Pointers
- ConstBox
- ByteAlign

### LinkPath:
- Sync, ChIn, ChOut, ChEvent, If’s

### Target Board No.1
- 89$
- Avnet Micro Board
- MemLPDDR *(32Mx16 on board)*
- XC6LX9

### Target Board No.2
- 199$
- Digilent ATLYS
- MemDDR2 *(64Mx16 on board)*
- XC6LX45

### Target Board No.3
- 99$
- Digilent Arty
- MemDDR3 *(128Mx16 on board)*
- XC7A35T

Remark: Blocks in red still N/A.
5.a Data Path

**ALU:**
- simple+fast!
- data manipulation
- address calculations
- note: 32bit address displacement has to be multiplied with number of “bytes-per-word” (i.e. `<<2`) before addition to base
note: register stack can do many actions within 1 clock in parallel, e.g. for MUL:
- add Breg to Creg,
- load Creg via Zbus_sh_r
- shift Zbus(0) into Areg
- decrement Dreg
5.b Control Path

- autonomous pre-fetch (FSM) has buffer for 2 words (8 instructions)
- each instruction consists of a single byte divided into two four bit parts.
- instruction fetch is from (virtual) instruction buffer, i.e. via 1 of 8 multiplexor located after pre-fetch buffer

Instruction Buffer:

<table>
<thead>
<tr>
<th>Function</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 ... 4 3 ... 0</td>
<td></td>
</tr>
</tbody>
</table>

1 byte instructions

1. \( FA = OPR = 1 1 1 1 0 1 0 1 \) micro code entry addr.:

<table>
<thead>
<tr>
<th>Operand Register</th>
<th>0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10 0000 0101 = ADD]</td>
<td></td>
</tr>
</tbody>
</table>

2 byte instructions

1. \( 23 = PFIX \) \( 0 0 1 0 0 0 1 1 \)
2. \( F3 = OPR = 1 1 1 1 0 0 1 1 \) micro code entry addr.:

<table>
<thead>
<tr>
<th>Operand Register</th>
<th>0 1 1 1</th>
<th>0 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[00 0011 0011 = XOR]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 byte instructions

1. \( 21 = PFIX \) \( 0 0 1 0 0 0 0 1 \)
2. \( 27 = PFIX \) \( 0 0 1 0 0 1 1 1 \) secondary instructions
3. \( FC = OPR = 1 1 1 1 1 1 0 0 \) micro code entry addr.:

<table>
<thead>
<tr>
<th>Operand Register</th>
<th>1</th>
<th>0 1 1 1</th>
<th>1 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>[01 0111 1100 = LDDEVID]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Operand Register will be loaded with 0 after each instruction, except PFIX and NFIX, which will accumulate operand nibbles.
T42 Pipeline

--- <------- IF/ID -------> <------- EX -------> ---

*Deco_PNFix* <------- > <----- Deco_PNF_reg

---

Pre-

(fetch) --

(Logic) |

-----

(IDecode)

{Logic}

---

P

r

e
f
from e |

Mem | path |

t

----------

fnct: '-' |

:fnct: '-' |

:fnct: '-' |

:fnct: '-' |

28bit / 

| | | |

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit

| | | |

| | | |

| | | |

28bit
5.c Links tbd.

source: US-Pat-4783734

- Link-Stack: Breg (Zbus) → CountReg → PtrReg → DBuffReg → (Ubus) Areg
- Idea: use (additional) Sbits to give start & stop pulses to the link state machines.
memory hierarchy:
• on-chip memory (2 clock access) e.g. for process workspace
• cache memory (3 clock access due to tag RAM) … still t.b.d.
• external memory (10+ clock access DDR-RAM) … still t.b.d.

defining address and data bus structures and protocols (arbiter) for 6 consumers (rd) and 5 producers (wr):
• CPU data path (rd/wr)
• CPU pre-fetch (rd)
• 4x Link-in DMA (wr)
• 4x Link-out DMA (rd)
T42 Schematic

- 8kB DPRAM (On-Chip)
- 2nd 8kB DPRAM (preliminary instead of Caches)
- T42-CPU
- 512x 96bit uCode ROM
- Link 0-3 & DMA’s (N/A)
- System Services
- Timers
- IMS-T425
- Link 0-3
- 4k5 RAM
- CPU
- System Services & Event Channel
- T42-in-FPGA
- 8kB RAM
- CPU
- System Services & Event Channel
- DDR-RAM Controller
- Instruction & Data Cache
### CPU:Cache:DDRCtrl = 1:2:4

Thanks to **Martin Zabel** for **Estimations** (01-Jul-2016 ; upd. 19-Jul-2017)

<table>
<thead>
<tr>
<th>T42 &amp; DDR-RAM</th>
<th>Spartan 6 LUTs / BRAM</th>
<th>Artix 7 LUTs / BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>T42 core (16-May-2016 w/o Links)</td>
<td>1800 / 8+4</td>
<td><del>same expected</del></td>
</tr>
<tr>
<td>T42 links (estimation)</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>8kB Cache (16 Byte = 128bit per Line) Controller (4x associative) + Tag RAM</td>
<td>4000 / 4</td>
<td><del>same</del></td>
</tr>
<tr>
<td>8kB Cache (16 Byte = 128bit per Line) Controller (16x associative) + Tag RAM</td>
<td>5100 / 4</td>
<td><del>same</del></td>
</tr>
<tr>
<td>DDR/2/3 Controller (multi bank capable)</td>
<td>Xilinx Hw. MCB + 700</td>
<td>7000 *</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(no MCB avail.)</td>
</tr>
</tbody>
</table>

FPGA utilization of a minimal configuration (3000+4000+700 = 7700 LUTs):

- **XC6LX9** ( 5720 LUTs / 32 BRAMs) LUTs > 100% / BRAMs ~ 50%
- **XC6LX45** (27228 LUTs / 116 BRAMs) LUTs ~ 28% / BRAMs ~ 14%
- **XC7AT35** (20568 LUTs / 65 BRAMs) * LUTs > 71% / BRAMs ~ 25%

* - no MCB available.
5.e System Control

- Takes decision which micro word is executed as next
- Deals w/ events from: timer, links, extern

→ HW: 2 cascaded priority encoders
  1. Micro code entry address control → select next address
  2. Event order control → generate event start addresses

- Maintain processor status register:

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 16/15</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>j0Break</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mov2Dnonzero</td>
<td>Mov2Dall</td>
</tr>
<tr>
<td>H</td>
<td>T</td>
<td>I</td>
</tr>
<tr>
<td>D</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(6) DIST and INS (not used)
(7) HALT on Error
(8-15) MOVE2Dall
(16-22) MOVE2Dnon
(23-30) j0Break
(31) Error

(0) '0'
(1) Goto Start Next Process
(2) IO Run (used after IN, OUT)
(3) MOVE = COPY
(4) DELETE
(5) INSERT
GoSNP is depending PROC_pri

* signal required for middle step (entry!) @hi-prio to guarantee run-to-end!
  input is always ANDed with NOT Wptr0pri for hi-prio signals

** Note Event order: in US-Pat-4989133 MOV_1 bit is lowest (after ChnReq1)
(different - vs - 20140904_T4-Transputer-Spec_19880927.pdf)
6.0 Verification: TVS-1

Regression Test Bench!

- **TVS-1** uses golden reference(s) based on real **T425** output(s).
- **54 IUT** (instructions under test) with 1, 2 or 3 operands
- **adaption** for VHDL simulation was required: reduction of sample set count from $2^{17}$ (128k) to $2^{12}$ (4k) to meet suitable run time(s)
- assembler code and sample set will be loaded in on-chip memories before each simulation run … for comparison w/ golden reference
- **basic test set** contains **32** selected 32bit data values (corner cases, single moving ‘1’ and ‘0’, small and large integers)

info: 3 different simulators can be used (GHDL, Modelsim, Xilinx iSim)
info: **TVS-1** was written by Michael Bruestle in 2010 to support software development & verification of Transputer emulator project (Gavin Crate)
6.1 Verification: TVS-1

**TVS-1** covers 54 instructions:

- primary (3/16) ldc, adc, eqc
- arithm. logic (16/17) add, gt, xor, ...
- long arithmetic (9/9) ladd, lsum, ...
- indexing (5/8) bsub, wcnt, ...
- error handling (2/8) ccnt1, csub0, ...
- general (7/8) csngl, xword, ...
- CRC and bits (5/5) bitcnt, ...
- floating point (5/6) unpack, ...
- ALT (2/12) alt, talt

**TVS-1** has 7 different input files, depending on IUT requirements.

```plaintext
; load test
ldl  CREG
ldl  BREG
ldl  AREG

__IUT__
stl  AREG
stl  BREG
stl  CREG
testerr
stl  ERROR

; send result
```
6.2 Verification: TVS-1

<table>
<thead>
<tr>
<th>TVS-1</th>
<th>input set</th>
<th>Extra Constants</th>
<th>Areg Values</th>
<th>Breg Values</th>
<th>Creg Values</th>
<th>NO. of TESTs</th>
<th>WORDs per SET</th>
<th>IN-WORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST.1</td>
<td>i32_1.bin</td>
<td></td>
<td>128</td>
<td>BBBBBBBB</td>
<td>CCCCCCCC</td>
<td>128</td>
<td>3</td>
<td>384</td>
</tr>
<tr>
<td>TEST.1.4</td>
<td>i32_1.bin</td>
<td>8</td>
<td>128</td>
<td>BBBBBBBB</td>
<td>CCCCCCCC</td>
<td>1.024</td>
<td>3</td>
<td>3072</td>
</tr>
<tr>
<td>TEST.2</td>
<td>i32_2.bin</td>
<td></td>
<td>128</td>
<td>128</td>
<td>CCCCCCCC</td>
<td>16.384</td>
<td>3</td>
<td>49152</td>
</tr>
<tr>
<td>TEST.3</td>
<td>i32_3.bin</td>
<td></td>
<td>128</td>
<td>128</td>
<td>8</td>
<td>131.072</td>
<td>3</td>
<td>393216</td>
</tr>
<tr>
<td>TEST.B</td>
<td>i32_B.bin</td>
<td></td>
<td>32</td>
<td>128</td>
<td>8</td>
<td>32.768</td>
<td>3</td>
<td>98304</td>
</tr>
<tr>
<td>TEST.F</td>
<td>i32_F.bin</td>
<td></td>
<td>64</td>
<td>BBBBBBBB</td>
<td>CCCCCCCC</td>
<td>64</td>
<td>3</td>
<td>192</td>
</tr>
<tr>
<td>TEST.P</td>
<td>i32_P.bin</td>
<td>14</td>
<td>8</td>
<td>72</td>
<td>14</td>
<td>112.896</td>
<td>4</td>
<td>451584</td>
</tr>
<tr>
<td>TEST.S</td>
<td>i32_S.bin</td>
<td></td>
<td>66</td>
<td>128</td>
<td>8</td>
<td>67.584</td>
<td>3</td>
<td>202752</td>
</tr>
</tbody>
</table>

- Original TVS-1 has ~350,000 tests (can be used over link only)
- T42 TVS-1 will have ~25,000 tests (may be increased if necessary)

**TVS-1 Benefit:**
- a TVS-1 run after (some) VHDL modifications will verify if the design still meets the spec ... or if there is any (bad) impact from changes.
6.3 Verification: TVS-1

<table>
<thead>
<tr>
<th>TVS-1 (T-42)</th>
<th>input set</th>
<th>Extra Constants</th>
<th>Areg Values</th>
<th>Breg Values</th>
<th>Creg Values</th>
<th>NO. of TESTs</th>
<th>WORDs per SET</th>
<th>IN-WORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST.1</td>
<td>i32_1.bin</td>
<td></td>
<td>128</td>
<td>BBBB</td>
<td>CCCCC</td>
<td>128</td>
<td>3</td>
<td>384</td>
</tr>
<tr>
<td>TEST.1.4</td>
<td>i32_1.bin</td>
<td>8</td>
<td>128</td>
<td>BBBB</td>
<td>CCCCC</td>
<td>1.024</td>
<td>3</td>
<td>3072</td>
</tr>
<tr>
<td>TEST.2</td>
<td>i32_2.bin</td>
<td></td>
<td>32</td>
<td>32</td>
<td>CCCCC</td>
<td>1.024</td>
<td>3</td>
<td>3072</td>
</tr>
<tr>
<td>TEST.3</td>
<td>i32_3.bin</td>
<td></td>
<td>32</td>
<td>32</td>
<td>8</td>
<td>8.192</td>
<td>3</td>
<td>24576</td>
</tr>
<tr>
<td>TEST.B</td>
<td>i32_B.bin</td>
<td></td>
<td>32</td>
<td>32</td>
<td>2</td>
<td>2.048</td>
<td>3</td>
<td>6144</td>
</tr>
<tr>
<td>TEST.F</td>
<td>i32_F.bin</td>
<td></td>
<td>64</td>
<td>BBBB</td>
<td>CCCCC</td>
<td>64</td>
<td>3</td>
<td>192</td>
</tr>
<tr>
<td>TEST.P</td>
<td>i32_P.bin</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>8.000</td>
<td>4</td>
<td>32000</td>
</tr>
<tr>
<td>TEST.S</td>
<td>i32_S.bin</td>
<td></td>
<td>32</td>
<td>32</td>
<td>4</td>
<td>4.096</td>
<td>3</td>
<td>12288</td>
</tr>
</tbody>
</table>

report example:

run_tvs1_tb_07_ghdl_sim.BAT Start Time is ... 04.07.2017_17:21:02,76
---------------------------------------------------------------------
  prep_iut.BAT: IUT is ADC  prepared @ 04.07.2017 17:21:02,89
tb_07_tvs1.vhd: simulation started...
tb_07_tvs1.vhd: simulation Ok. - end of ..\..\..\..\simulation\tb_07\golden_reference.mem reached.
  ghdl_sim.BAT: IUT is ADC  finished @ 04.07.2017 17:21:26,64
---------------------------------------------------------------------
  prep_iut.BAT: IUT is ADD  prepared @ 04.07.2017 17:21:26,76
tb_07_tvs1.vhd: simulation started...
tb_07_tvs1.vhd: simulation Ok. - end of ..\..\..\..\simulation\tb_07\golden_reference.mem reached ...

...
7.0 Summary & Conclusions

what has been achieved (2014-2017):

1. partial Transputer implementation available:
   – integer CPU
   – simple memory path („on-chip“)
   – timer

2. semi automated micro code generation possible.
   – one update run takes less than 1 minute

3. already 100 (of 134) instructions in micro code written
   – ~50 tested & used (e.g. primary instructions)
     • ~30 instructions proven correct by TVS-1 regression test bench

→ best conditions to continue and finish the whole design!
7.1 Work! Work! Work!

still a lot of work to do:  (partially pre-prepared already)
• IUTs to debug: IN, OUT, ALTs, PARs, CRCs, BITs
• IUTs still to implement: MULs, DIVs, FP support
• scheduler code to debug: start next process, time slice
• micro code ROM increase: 1024x128bit (pre-prepared)
• scheduler code completion: HW event interaction w/ timer, links, boot, peek, poke, Halt-on-Error, analyse
• reverse engineering: links + control logic (in prep.)

by help of TUD & POC library:  (pre-prepared)
• connect external memory: cache & DDR-RAM controller
7.2 Prospects

T42-in-FPGA …

- is an open source design (GNU Public License v.3) … enhance it!
- it’s the “multi-many“ core toy for students to play with in FPGA!
- is a nice HW+SW example for teaching concurrency (CSP)!
- e.g. many cores (e.g. 8++) may fit into a FPGA on an ATLYS board:

Benefits in Future may be:

- HW accelerators can be easily and transparent implemented as (Occam) channels in FPGA!
- T42 performance can be enhanced step by step … following a proven roadmap!
- T42 may help to overcome the absence of CSP in teaching, engineering and public perception.
- T42 can be a promising example vs CPU mono culture in FPGA ;-(
7.3 Transputer & FPGA

- **ideal partners!** cheap FPGA boards for the lab are 100~200EUR
- memory on board: e.g. 128MByte LPDDR, DDR2/3 up to 1.6GB/s
- fast RxTx (Link) support up to 3.2Gbit/s (LVDS) possible
- **examples** of available RISC soft cores (commercial & open cores):

<table>
<thead>
<tr>
<th>Softcore</th>
<th>Provider</th>
<th>Pipeline</th>
<th>FPGA</th>
<th>Clock</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microblaze</td>
<td>Xilinx</td>
<td>3-stage</td>
<td>Virtex-II</td>
<td>80MHz</td>
<td>2441</td>
</tr>
<tr>
<td>Microblaze</td>
<td>Xilinx</td>
<td>5-stage</td>
<td>Spartan-6</td>
<td>195MHz</td>
<td>1350</td>
</tr>
<tr>
<td>Microblaze</td>
<td>Xilinx</td>
<td>5-stage</td>
<td>Virtex-7</td>
<td>330MHz</td>
<td>1350</td>
</tr>
<tr>
<td>Open-RISC</td>
<td>OpenCores</td>
<td>5-stage</td>
<td>Virtex-5</td>
<td>185MHz</td>
<td>3802</td>
</tr>
<tr>
<td>MIPS-MP32</td>
<td>Altera</td>
<td>5,6-stage</td>
<td>Cyclone-3</td>
<td>115MHz</td>
<td>5500</td>
</tr>
<tr>
<td>MIPS-MP32</td>
<td>Altera</td>
<td>5,6-stage</td>
<td>Stratix-4</td>
<td>300MHz</td>
<td>5500</td>
</tr>
<tr>
<td>NIOS-II</td>
<td>Altera</td>
<td>5-stage</td>
<td>Stratix-V</td>
<td>305MHz</td>
<td>3200</td>
</tr>
<tr>
<td>T42</td>
<td>TUD</td>
<td>2-stage</td>
<td>Spartan-6</td>
<td>100MHz</td>
<td>4000</td>
</tr>
</tbody>
</table>
7.4 Benchmark Preview

- 8-stage Transputer* can beat any 6-stage MIPS!

- our *virtual* Benchmark: affine transformation in 3D (3x3 matrix, 9x MUL, 6xADD)
- conditions: same clock frequency, all variables already in cache (2clk latency)
- Example: MIPS ALU /w 4:1-forwarding mux in XC6S-LX16-3 runs at 186MHz

<table>
<thead>
<tr>
<th>Processor:</th>
<th>MIPS 5-stage</th>
<th>MIPS 6-stage</th>
<th>T425 2-stage</th>
<th>T44** 4-stage?</th>
<th>ST20-C2 8-stage</th>
<th>ST20-iC 8-stage</th>
<th>ST20-iE** 8-stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Cycles</td>
<td>91</td>
<td>78</td>
<td>205</td>
<td>134</td>
<td>109</td>
<td>71</td>
<td>62</td>
</tr>
<tr>
<td>Relative Perf.</td>
<td>0.85</td>
<td>1.00</td>
<td>0.38</td>
<td>0.58</td>
<td>0.72</td>
<td>1.10</td>
<td>1.26</td>
</tr>
<tr>
<td>Exec. Time</td>
<td>116%</td>
<td>100%</td>
<td>263%</td>
<td>172%</td>
<td>140%</td>
<td>91%</td>
<td>79%</td>
</tr>
<tr>
<td>Relative Perf.</td>
<td>2.25</td>
<td>2.63</td>
<td>1.00</td>
<td>1.53</td>
<td>1.88</td>
<td>2.89</td>
<td>3.31</td>
</tr>
</tbody>
</table>

***) fictive designs (estimated)

T. performance road map
Appendix: Achievements (1)

- T42 project started May’2013 – VHDL design started ... Jan‘2014
- data path and control path (1st concept) working ... Apr‘2014
- micro code assembler (12 AWK scripts) completed ... Jan‘2015
- ~50 simple instr. implemented, data path extended ... Apr‘2015
- pipeline is running (from 8 byte pre-fetch buffer) ... May‘2015
- on-chip memory added (load ... store) and verified ... Jun‘2015
- pre-fetch state machine + lptr incremener verified ... Jul‘2015
- system control unit, status bits, more flags added * ... Aug‘2015

mid.2015: core infrastructure is almost complete, but still * t. b. verified
Appendix: Achievements (2)

- system control unit, status bits to Sreg connected … Aug’2015
- timer VHDL (not fully tested yet, uCode missing!) … Sep’2015
- pipelined Oreg within IDecode (hardware pfix, nfix) … Nov’2015
- move + move2D: ByteAlign + uCode + move-bit ok. … Feb’2016
- MemIF w/ dual port arbiter completed (8kB + 8kB) … Apr’2016
- uCode for long arithmetics, error mode tested ok. … May’2016
- uCode for In, Out, ALT’s (no timer! still ongoing) … Jun’2016
- scheduler uCode (some 1st routines, still ongoing) … Jul’2016
- 1st trial VHDL of (the most simple) output link … Aug’2016

mid.2016: >460 lines uCode written (of 512, uCodeROM is almost full)
intension: understand the uCode for instructions and system control
Appendix: Achievements (3)

- memory renovation(s): 1024x128bit_ucrom prepared ... Nov.2016
- TVS-1 regression test bench preparations started ... Nov.2016
- clean up: fix simulation warnings about “X” values ... Jan.2017
- 1st instruction tests running (ADC, tvs1_i32-1.inp) ... Feb.2017
- all 54 instructions prepared and half of them running ... Apr.2017
- data path + uCrom renovation (for more instructions) ... May.2017
- full design environment reorganization (for SVN) ... Jul.2017
- switch to 1024x128bit_ucrom, MULs, DIVs, FPs ... ... Aug.2017

mid.2017: regression test bench for design verification in place
partial verified instruction set (i.e. whole design vs spec verified)
reorganized design environment, SVN repository
Appendix: Links & Literature

- **Documentation**: [www.tranputer.net](http://www.tranputer.net) ➔ INMOS Datasheets & Technical Notes
- **About Parallel**: [http://www.classiccmp.org/transputer](http://www.classiccmp.org/transputer) ➔ Boards, Hardware, Software
- **WoTUG Archive**: [http://www.wotug.org/parallel](http://www.wotug.org/parallel) ➔ Software, Documents, Papers
- Homepage of Gavin Crate’s Transputer-Emulator: [https://sites.google.com/site/transputeremulator](https://sites.google.com/site/transputeremulator)
Appendix: Patents

- Inmos Innovation and Patents: [http://www.petritzfoundation.org/?mdocs-file=968](http://www.petritzfoundation.org/?mdocs-file=968)
  The “In” in Inmos Stands for Innovation; James R. Adams, PhD

  Function set for a microcomputer

  Microcomputer with prefixing functions

  Microcomputer

  System for executing time dependent processes

  Computer with variable length process communication

  Microcomputer with priority scheduling
End