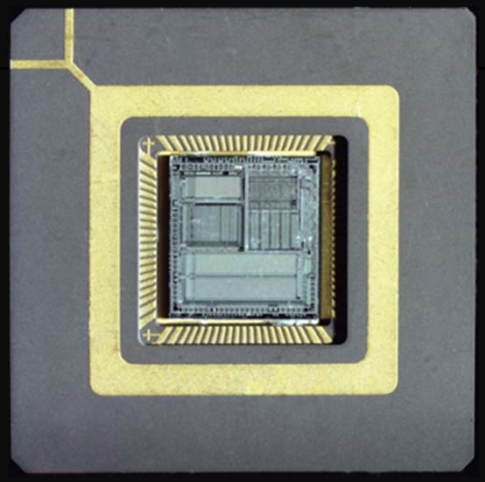
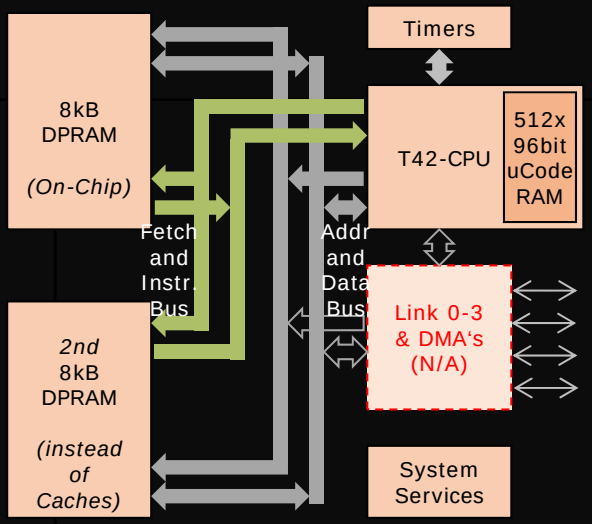
TUD Department of VLSI-Design, Diagnostic and Architecture

Department Seminar - Thu.20.Jul.2017

T42 Transputer-in-FPGA





Design Status Report (2014-2017) - Progress & Prospects

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!

Dresden, 19.Jul.2017 *Uwe Mielke*

Agenda

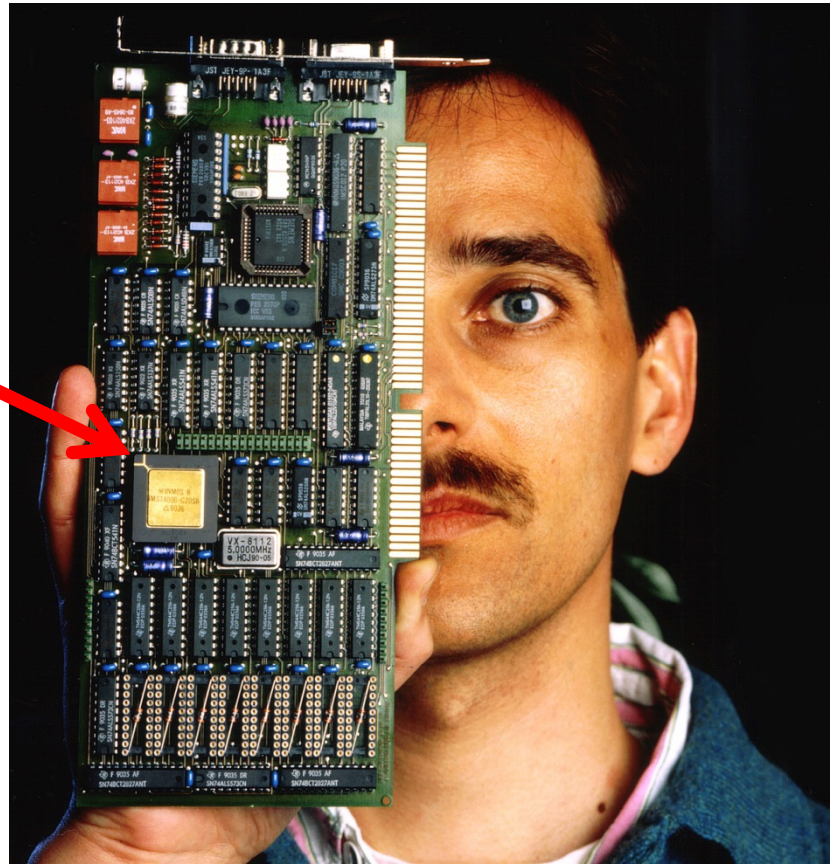
1. Motivation, Preparations
2. T425 Specification (Target)
3. Design Considerations
4. Micro Code, Assembler
5. Design Partitioning
   a. Data Path
   b. Control Path
   c. Links
   d. Memory Hierarchy
   e. System Control
6. Verification, Results
7. Summary, Conclusions, Prospects

1.0 Motivation

Why to reverse engineer
a chip from the past?

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

IMS T425



Picture:

AVM ISDN Controller B1 (1990)

presented by Peter Faxel (CTO)

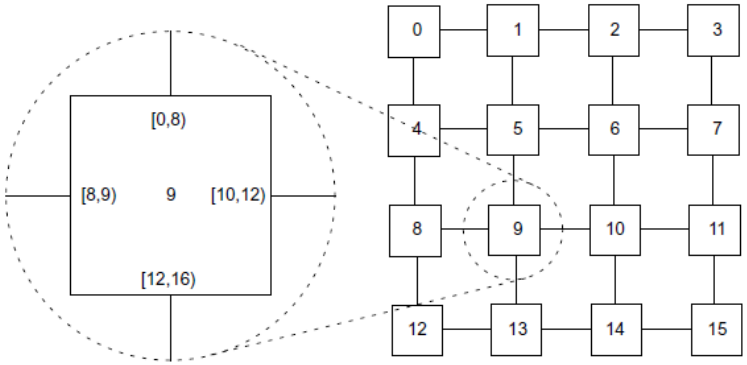
This worldwide most often sold ISDN card design was manufactured by AVM in different versions until 2015.

1.1 Motivation

Why to reverse engineer a chip from the past?

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

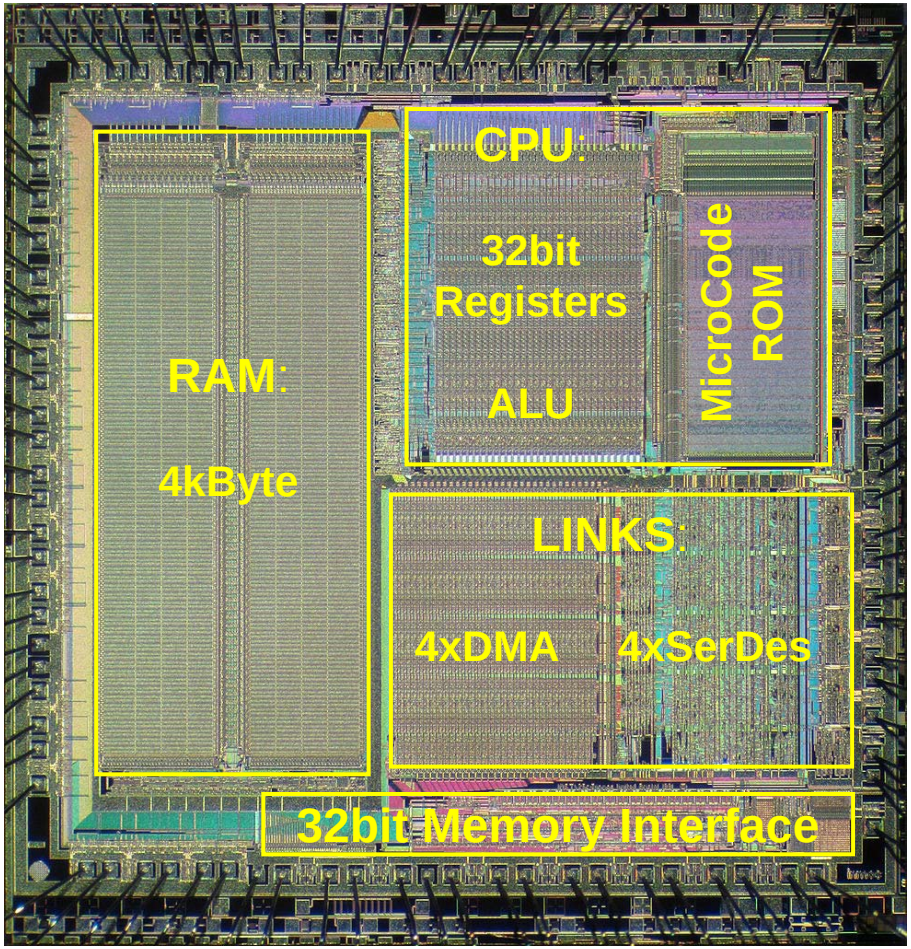
many of them could be connected to solve a problem together:



- 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...

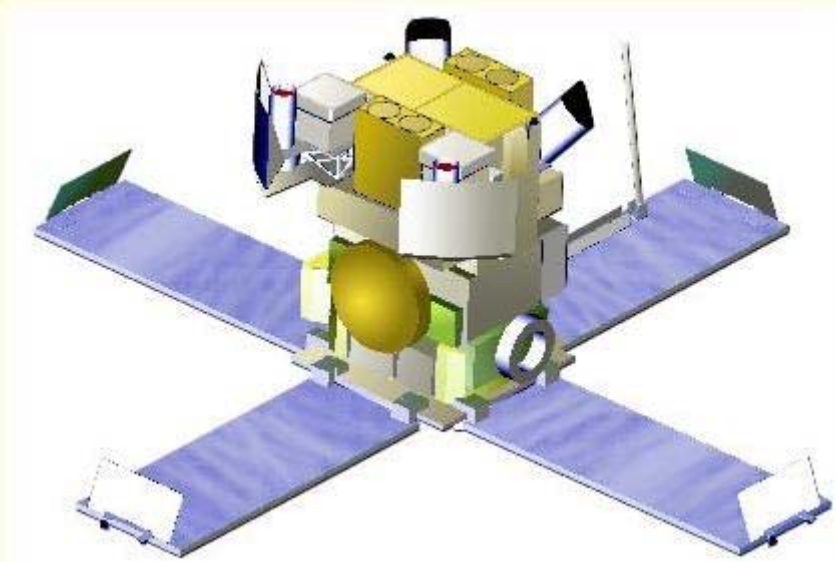


› IMS-T425-G20S chip 1024x981 (Pauli Rautakorpi) www.wikimedia.org

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



HETE = High Energy Transient Explorer (UV, X-ray, gamma-ray)

1992: Rank 259 on Top-500 List of Super Computers: 1024 CPU‘s providing

4,5 GFLOP/s

Now visible (since 2004) in the Heinz Nixdorf Computer Museums Forum in Paderborn.

parsytec *GigaCluster*

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 (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)

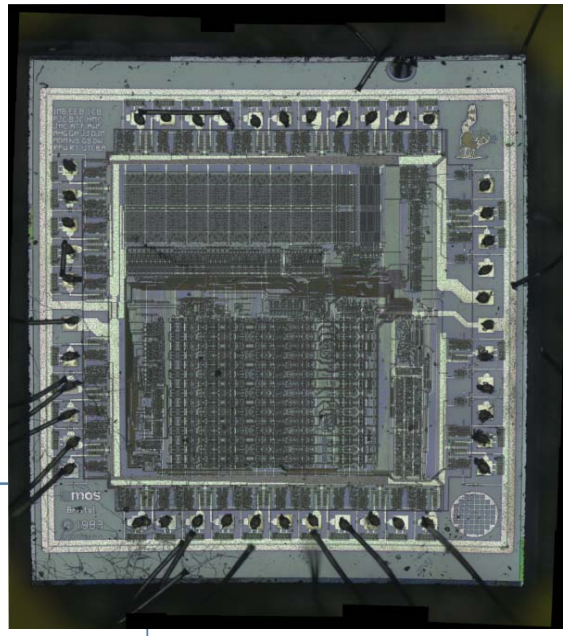
docu. from Transputer architect David May: http://www.cs.bris.ac.uk/~dave/transputer.html

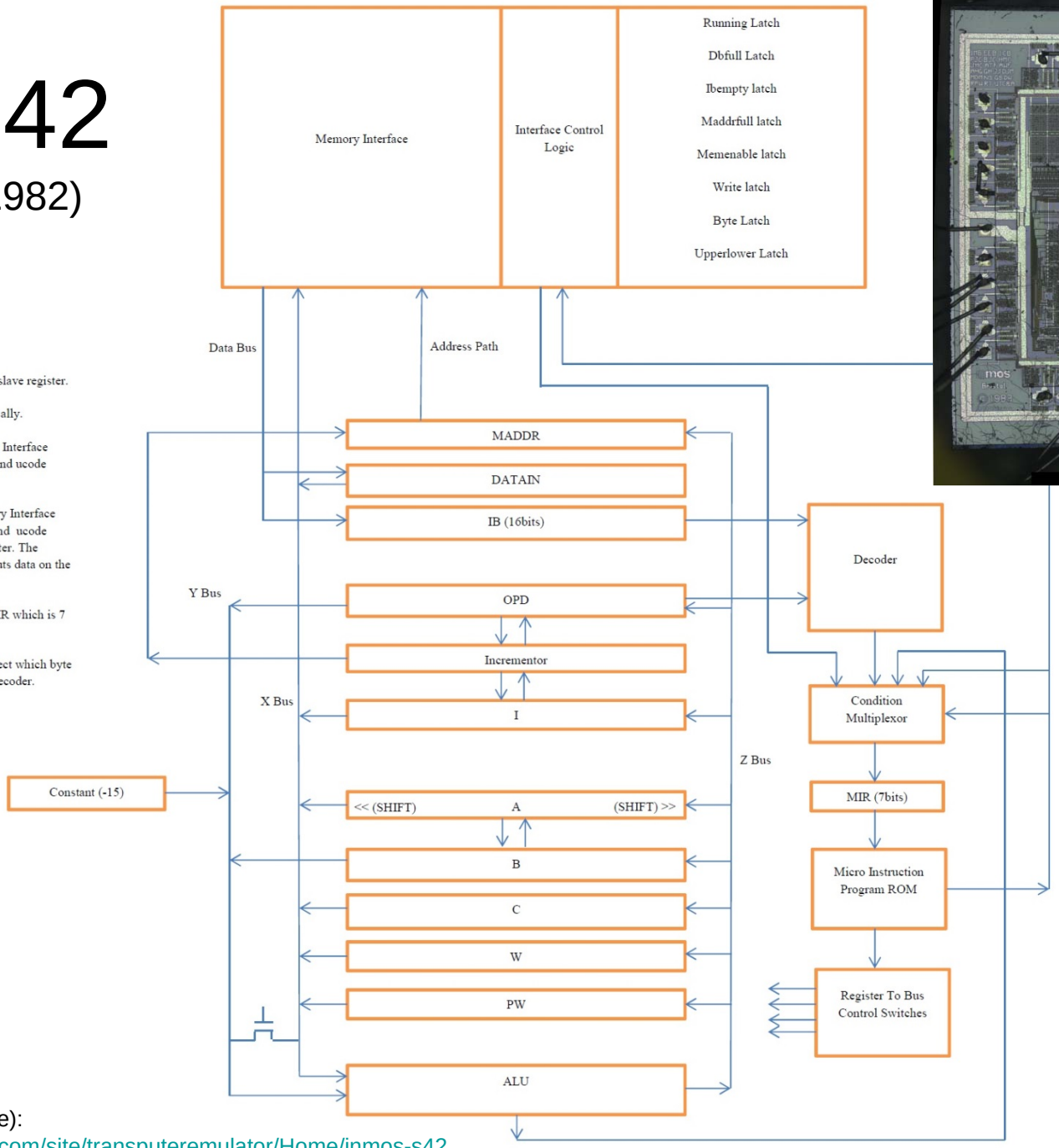
investigation from Gavin Crate: https://sites.google.com/site/transputeremulator/Home/inmos-s42

*) 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)





Notes:

A, B and C registers each have a slave register.

A register shifts using A slave locally.

During a ucode write to Memory Interface (MADDRfromZ Write), the second ucode supplies data on X Bus.

During a ucode read from Memory Interface (MADDRfromZ Read), the second ucode starts with data in DATAIN register. The (XfromDATAIN AND Dbfull) puts data on the X Bus.

All resisters are 16 bits except MIR which is 7 bits.

The lsb of I register is used to select which byte in the IB register is supplied to Decoder.

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

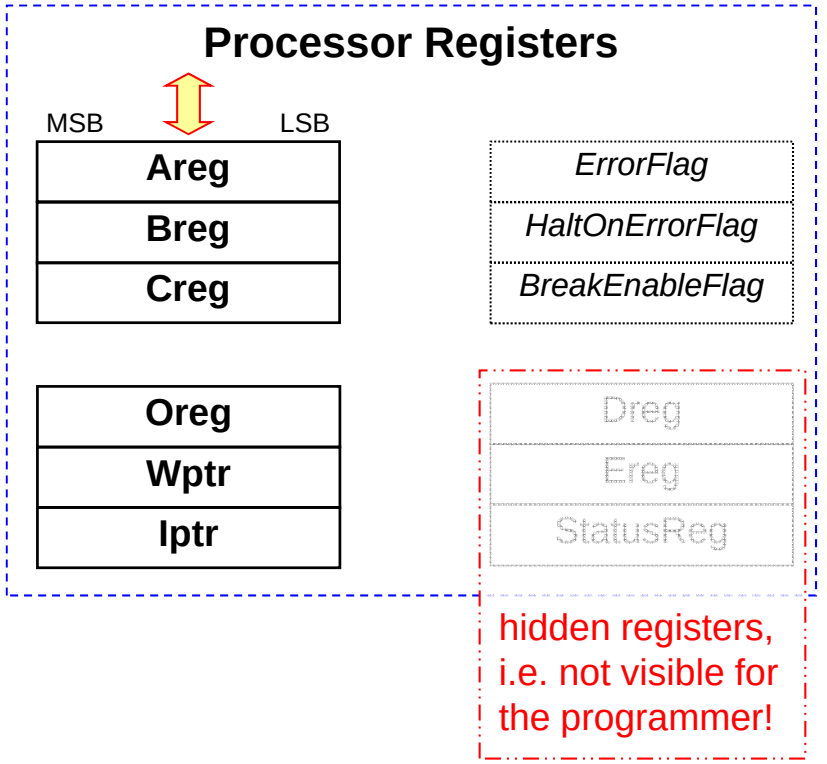
The **CPU** contains:

- sequential 32bit Integer **Processor**
- **Timers** & (micro-coded) Process **Scheduler**
- **Event Logic**

Reverse Polish Notation

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



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

2.0 T425 Specification

The **CPU** contains:

- sequential 32bit Integer **Processor**
- **Timers** & (micro-coded) Process **Scheduler**
- **Event Logic**

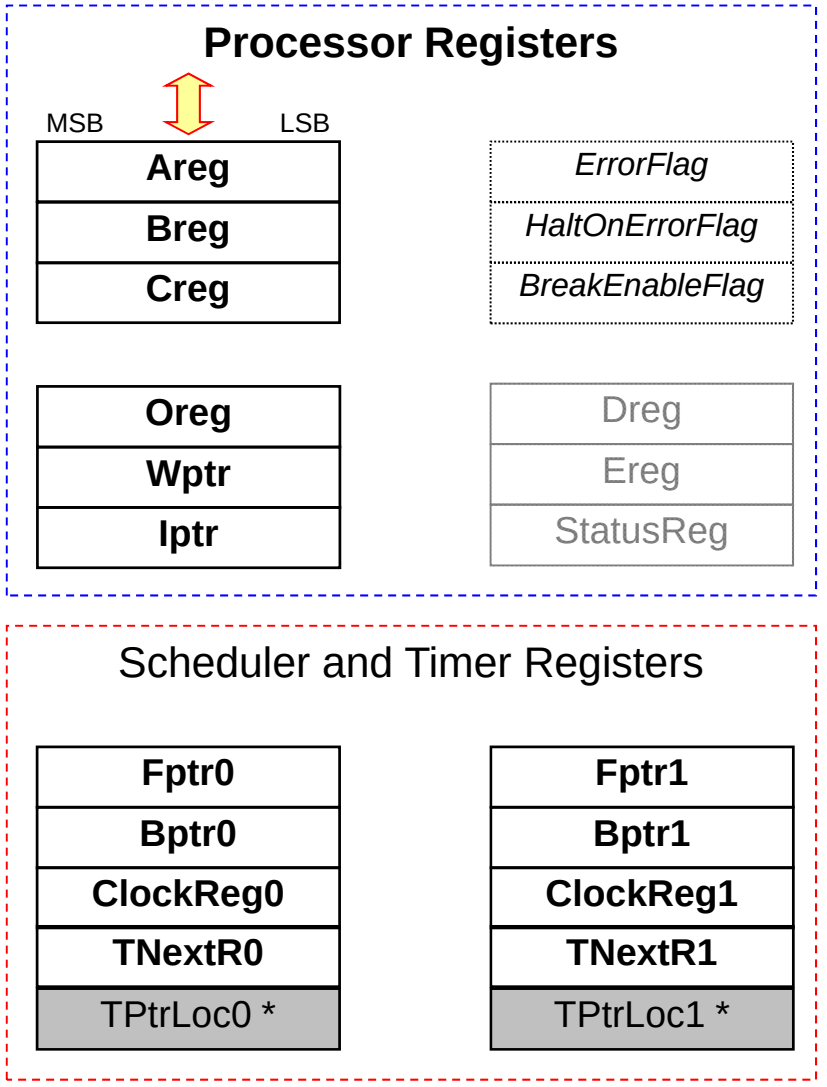
Reverse Polish Notation

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

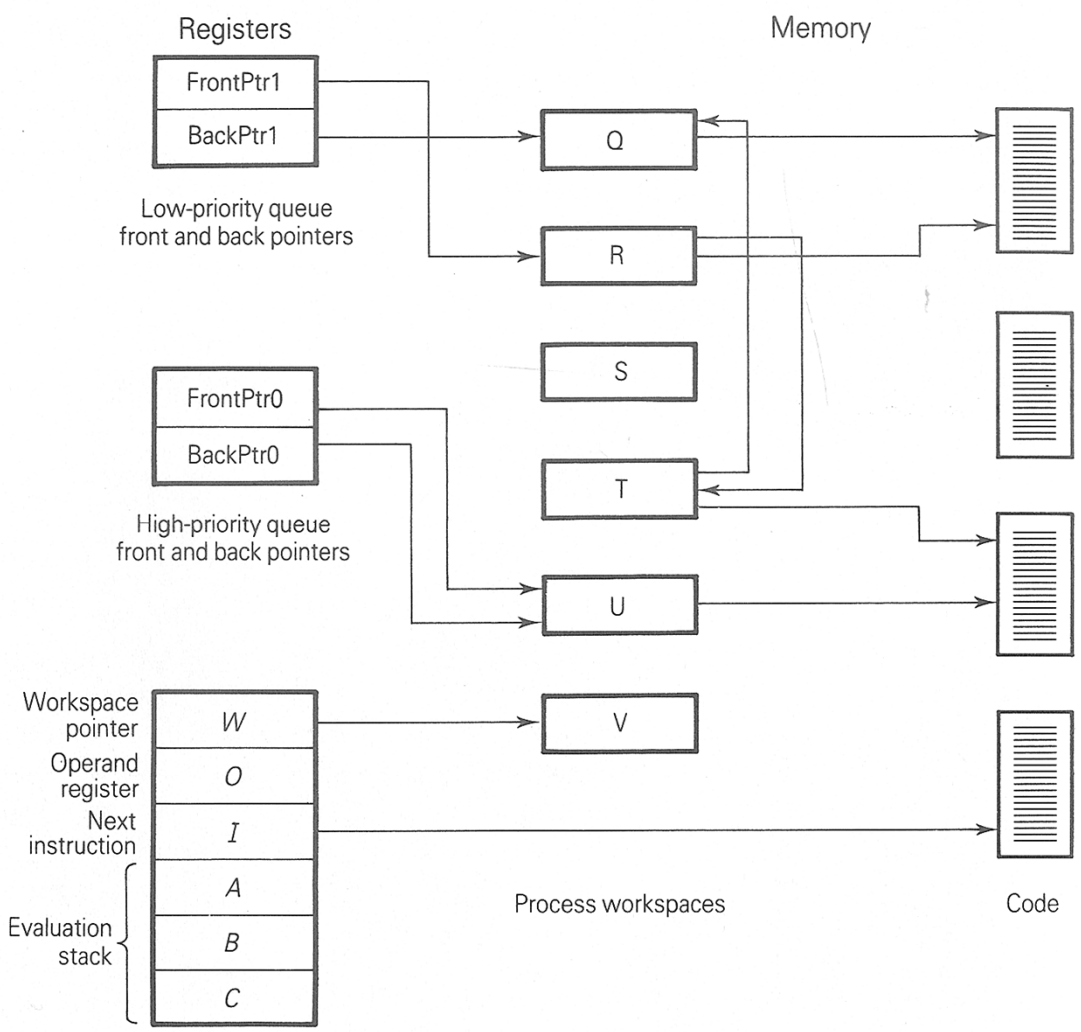
Scheduler and Timer Registers

- Front- and Back-Pointers of high and low priority process queues: **FptrX**, **BprtX** (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

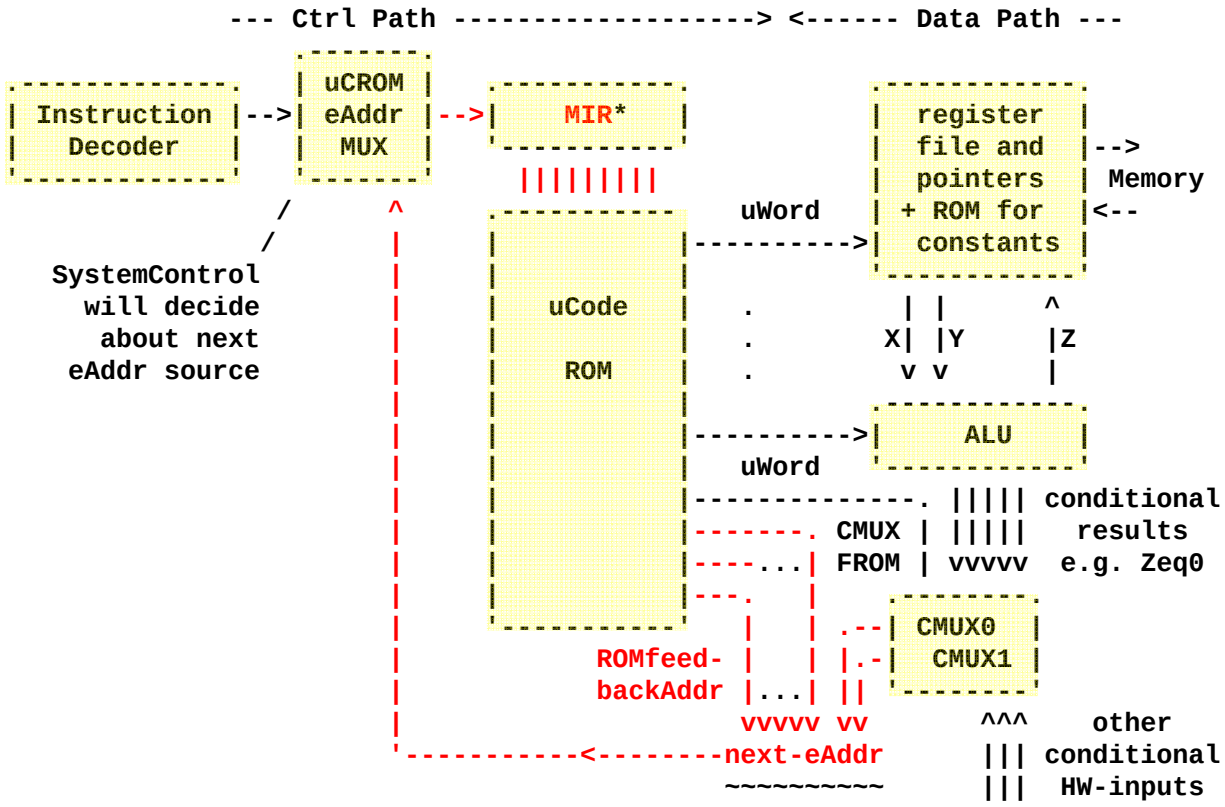
a list of some of my design project experiences after 3.5 years:

- 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):**

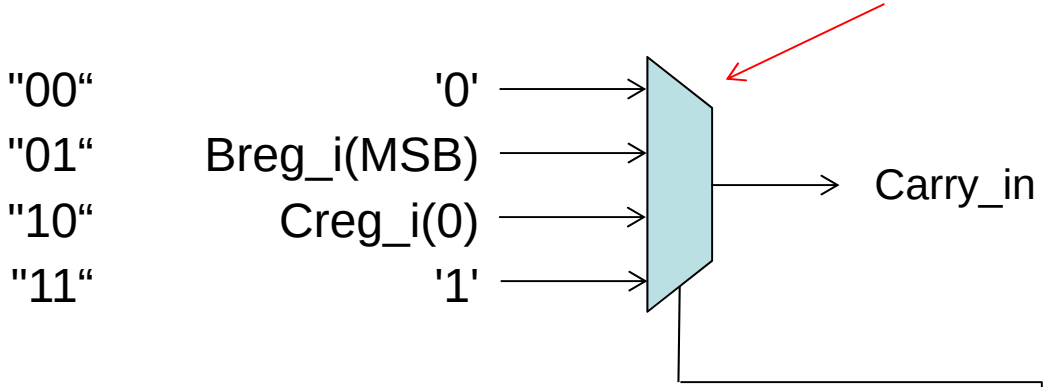
- T42 µCode ROM 1st size (2014) **512x96bit**
- T42 today (2017) **1024x128bit**
- T425 µCROM is **742x118bit** (~90kBit)
  investigation done in Nov.2016 by G. Crate



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

4.1 Micro Word

- micro word consist of micro operations
- µOp = bit group → HW control … e.g. MUX



- easy to be defined by only 2 VHDL files
  (many thanks to Martin Zabel) ☺
  1. t42_cpu_constpkg.vhd → bit coding of µOps
  2. t42cpu_ucode_rom.vhd → bit order in µWord

P.S.: shown µWord example is from 512x96bit →

```
ENTRYVALID;  95;  95;  -- 1 bit
NEXTACTION;  94;  94;  -- 1 bit
ERROR_MODE;  93;  93;  -- 1 bit
S_BIT_MODE;  89;  90;  -- 2 bit
S_BIT_FROM;  85;  88;  -- 4 bit
I_PTR_FROM;  83;  84;  -- 2 bit
W_PTR_FROM;  82;  82;  -- 1 bit
WPTR0_FROM;  81;  81;  -- 1 bit
X_BUS_FROM;  78;  80;  -- 3 bit
Y_BUS_FROM;  75;  77;  -- 3 bit
A_REG_FROM;  72;  74;  -- 3 bit
A_SHIFT_IN;  69;  71;  -- 3 bit
B_REG_FROM;  66;  68;  -- 3 bit
B_SHIFT_IN;  64;  65;  -- 2 bit
C_REG_FROM;  61;  63;  -- 3 bit
C_SHIFT_IN;  59;  60;  -- 2 bit
D_REG_FROM;  56;  58;  -- 3 bit
D_COUNT_IN;  54;  55;  -- 2 bit
E_REG_FROM;  51;  53;  -- 3 bit
E_SHIFT_IN;  49;  50;  -- 2 bit
CMUX1_FROM;  44;  46;  -- 3 bit
CMUX0_FROM;  40;  43;  -- 4 bit
MADDR_MODE;  38;  39;  -- 2 bit
MDATA_MODE;  36;  37;  -- 2 bit
MDATA_FROM;  33;  35;  -- 3 bit
XYSIGN_EXT;  32;  32;  -- 1 bit
Z_FROM_ALU;  26;  31;  -- 6 bit
CARRY_FROM;  24;  25;  -- 2 bit
CARRY_MODE;  23;  23;  -- 1 bit
POINT_MODE;  22;  22;  -- 1 bit
POINT_FROM;  17;  21;  -- 5 bit
TIMER_MODE;  15;  16;  -- 2 bit
PRESC_MODE;  14;  14;  -- 1 bit
CONST_FROM;   9;  13;  -- 5 bit
ROMFEEDBAK;   0;   8;  -- 9 bit
```

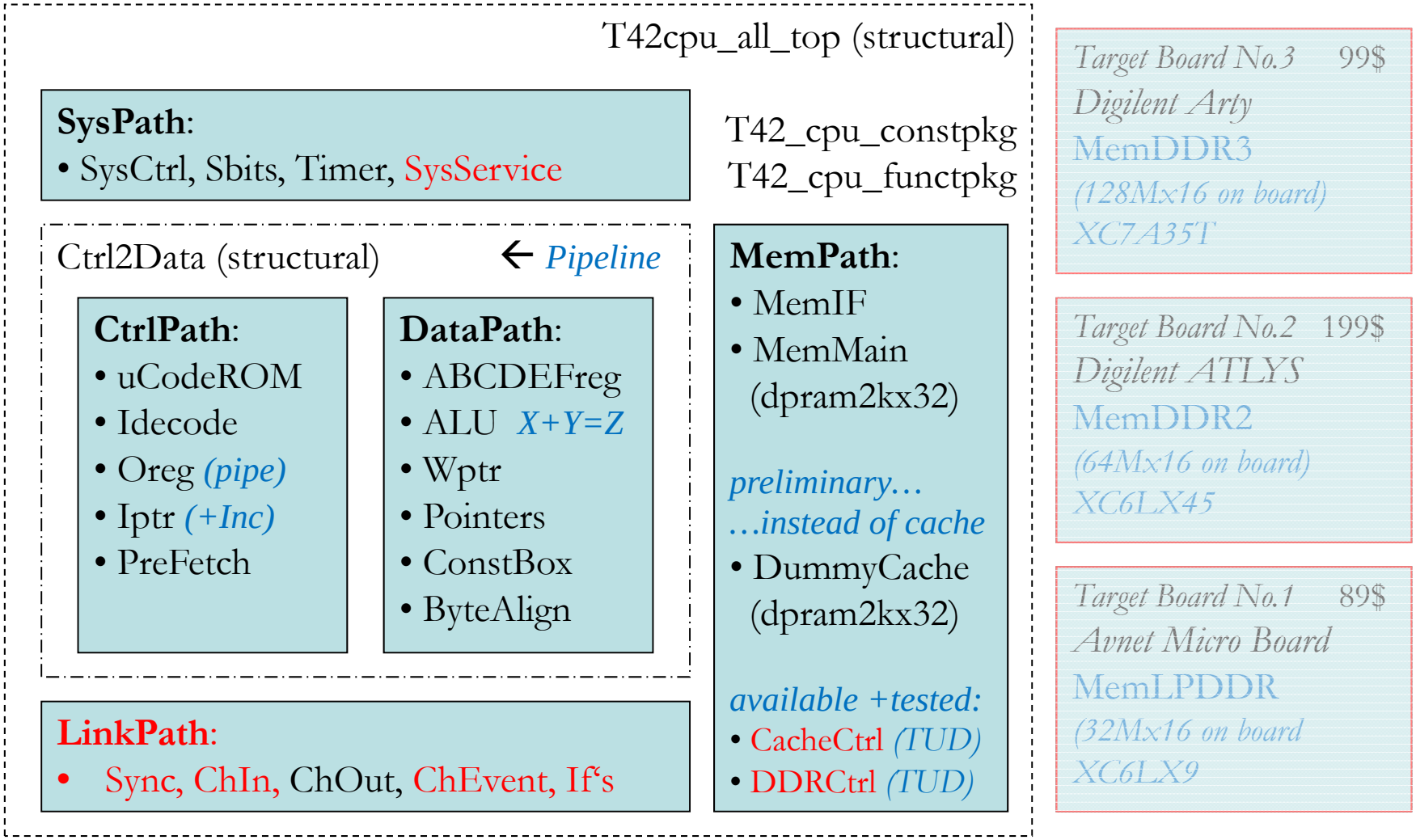
4.2 Micro Code Assembler

- 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

Remark: Blocks in red still N/A.

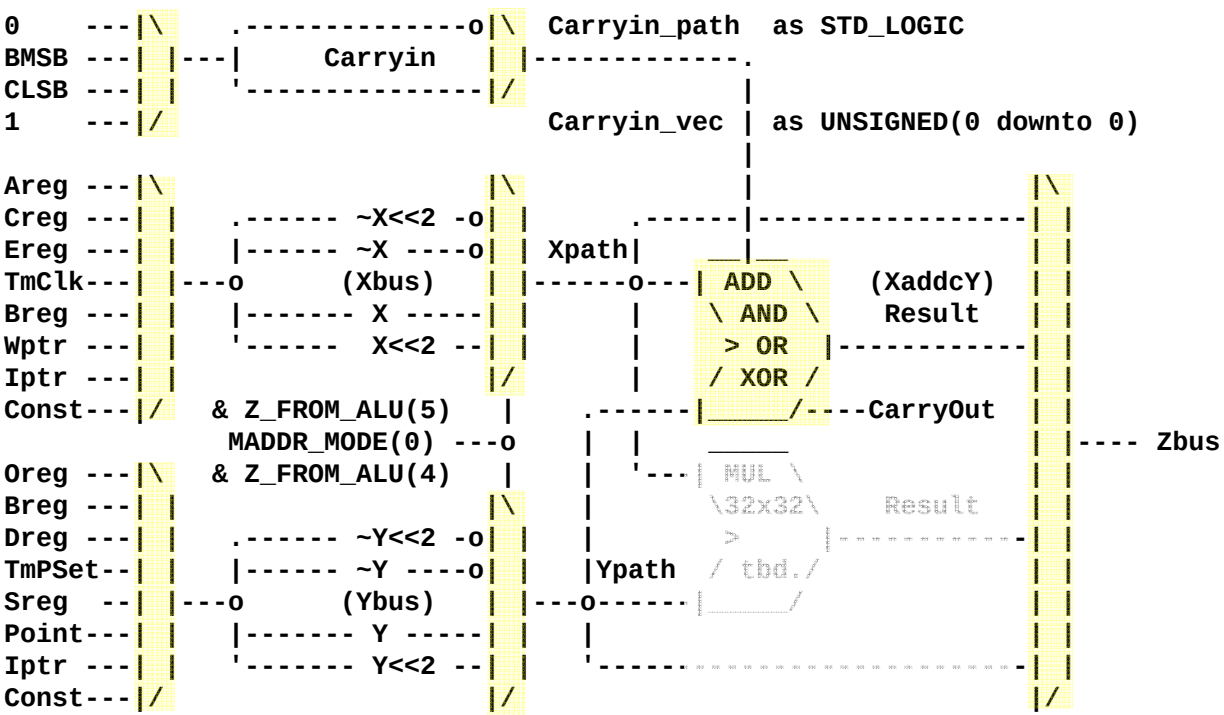
5.0 Design Partitioning



5.a Data Path

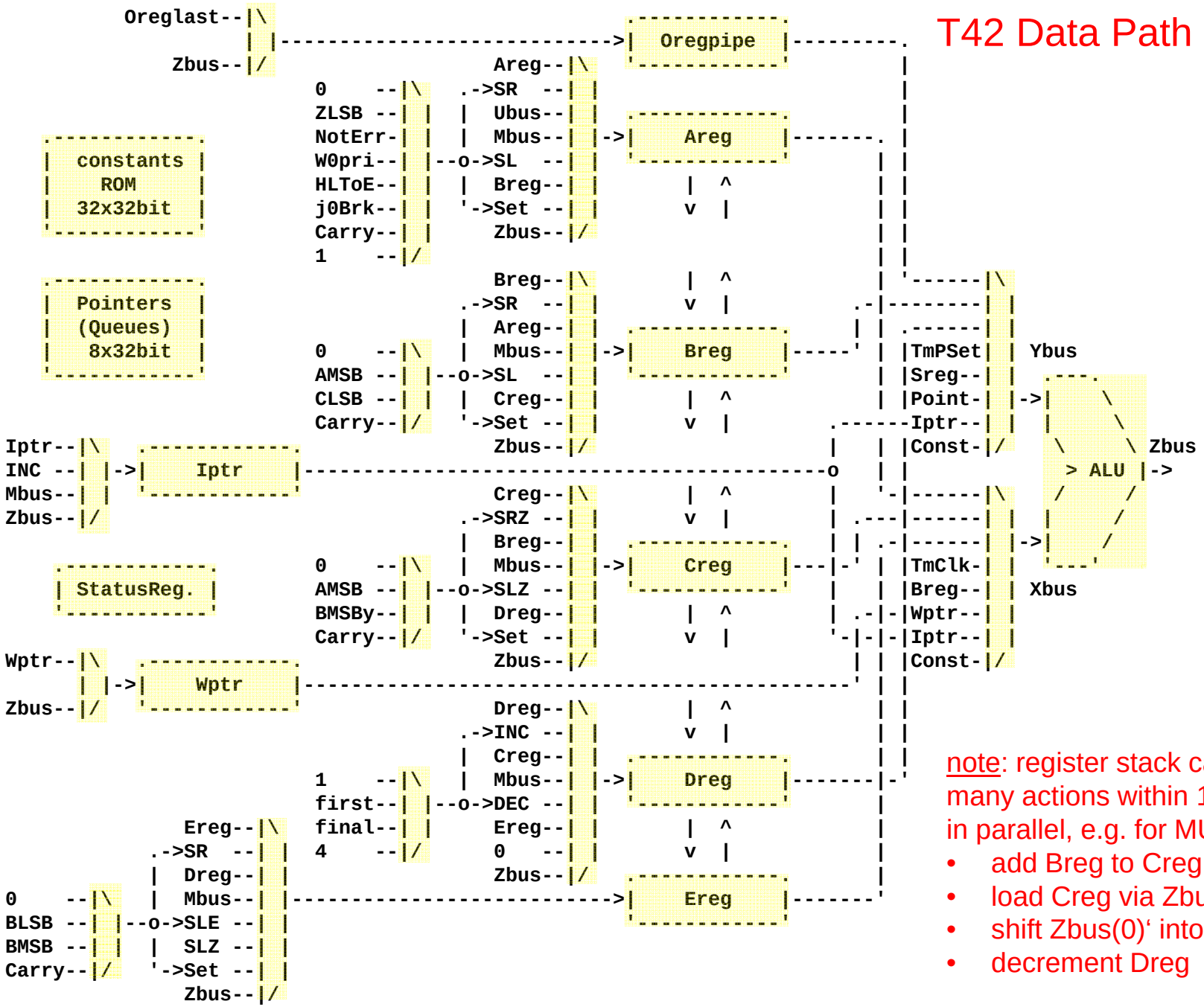
ALU:

- 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



MADDR_MODE(0)		path	out	Result	out	Zalu	out
BYTE	00 | X or Y	0.	~X	00	AND	00	ARITHM.LOG.
XWORD	01 | Y<<2	1.	X	01	XOR	01	~Y / Y
SIXT	10 | X or Y	.0	~Y	10	ADD	10	~X / X
YWORD	11 | X<<2	.1	Y	11	OR	11	MULTIPLY

T42 Data Path



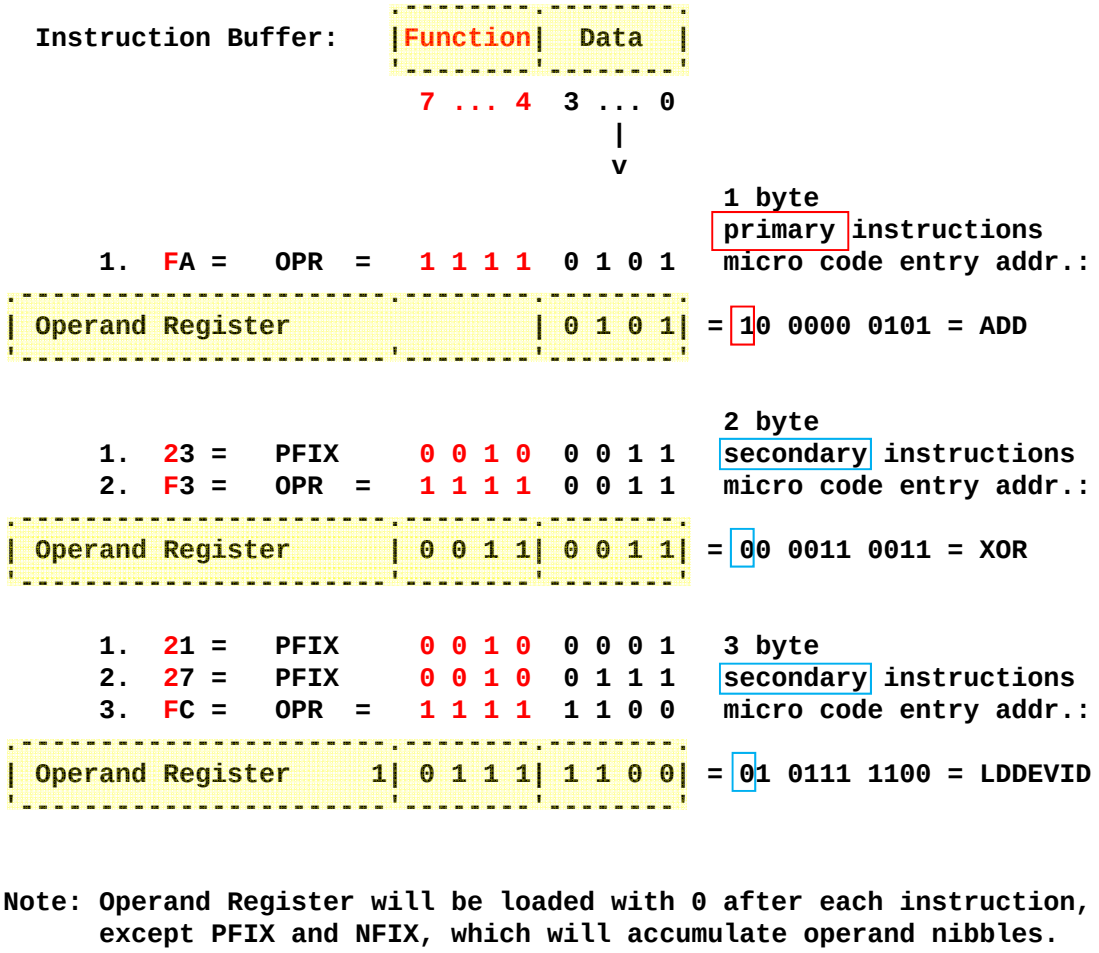
<u>note</u>: register stack can do many actions within 1 clock in parallel, e.g. for MUL :

- add Breg to Creg,
- load Creg via Zbus_sh_r
- shift Zbus(0)‘ into Areg
- decrement Dreg

5.b Control Path

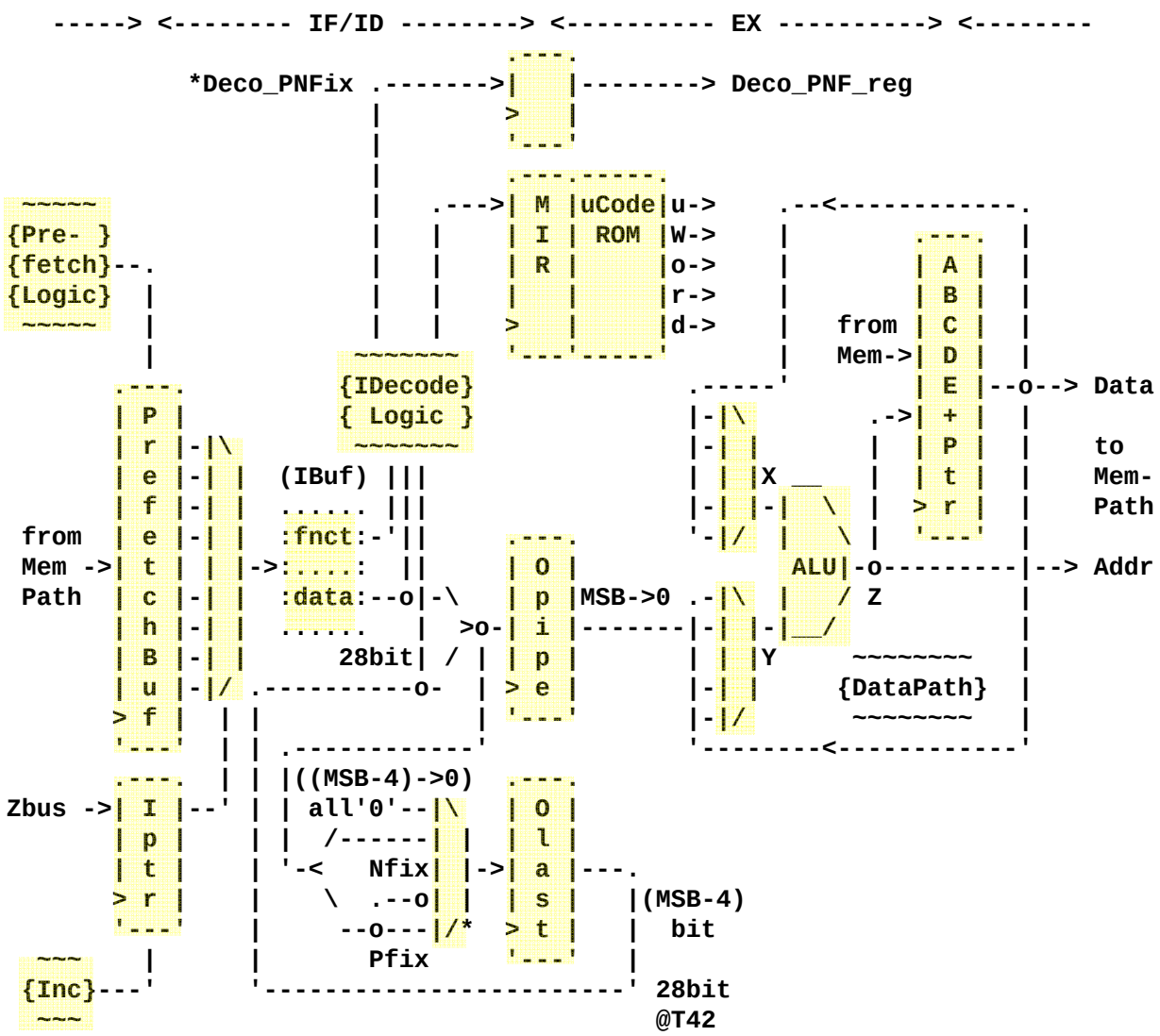
- autonomous pre-fetch (FSM) has buffer for 2 words (8 instructions)
- each instruction consists of a single byte divided into two four bit parts.
- instruction fetch is from (virtual) instruction buffer, i.e. via 1 of 8 multiplexor located after pre-fetch buffer

- **instruction decode …** (10bit addr. examples for 1024x128bit µCROM)



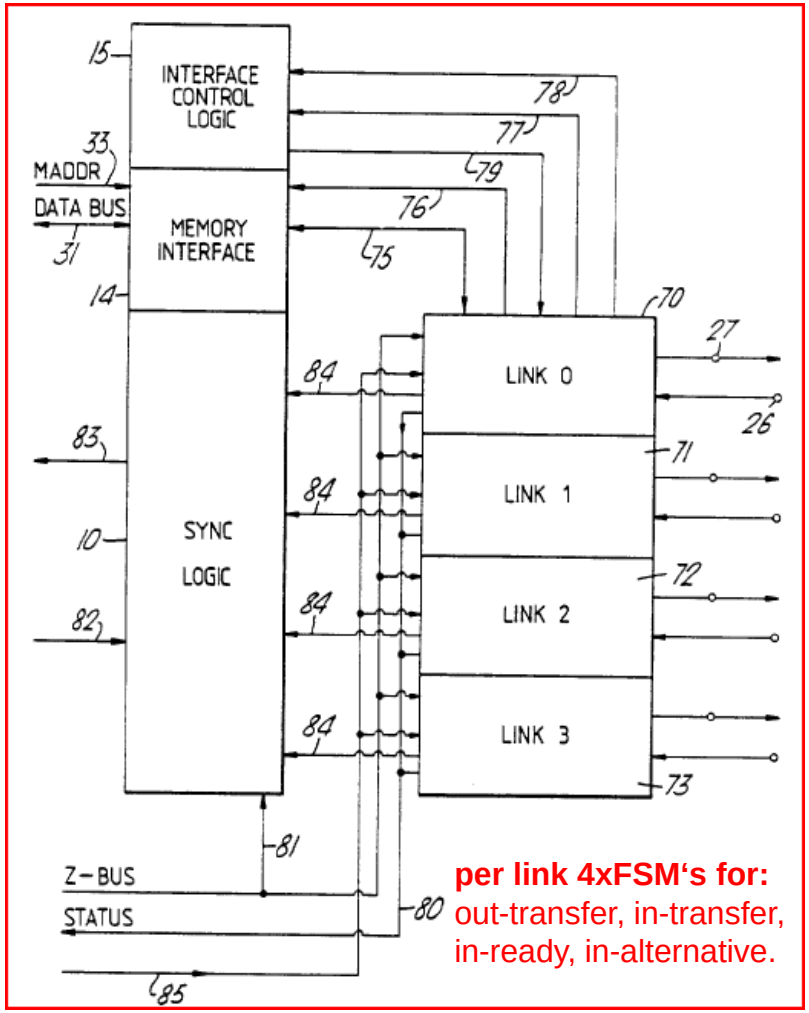
Note: Operand Register will be loaded with 0 after each instruction, except PFIX and NFIX, which will accumulate operand nibbles.

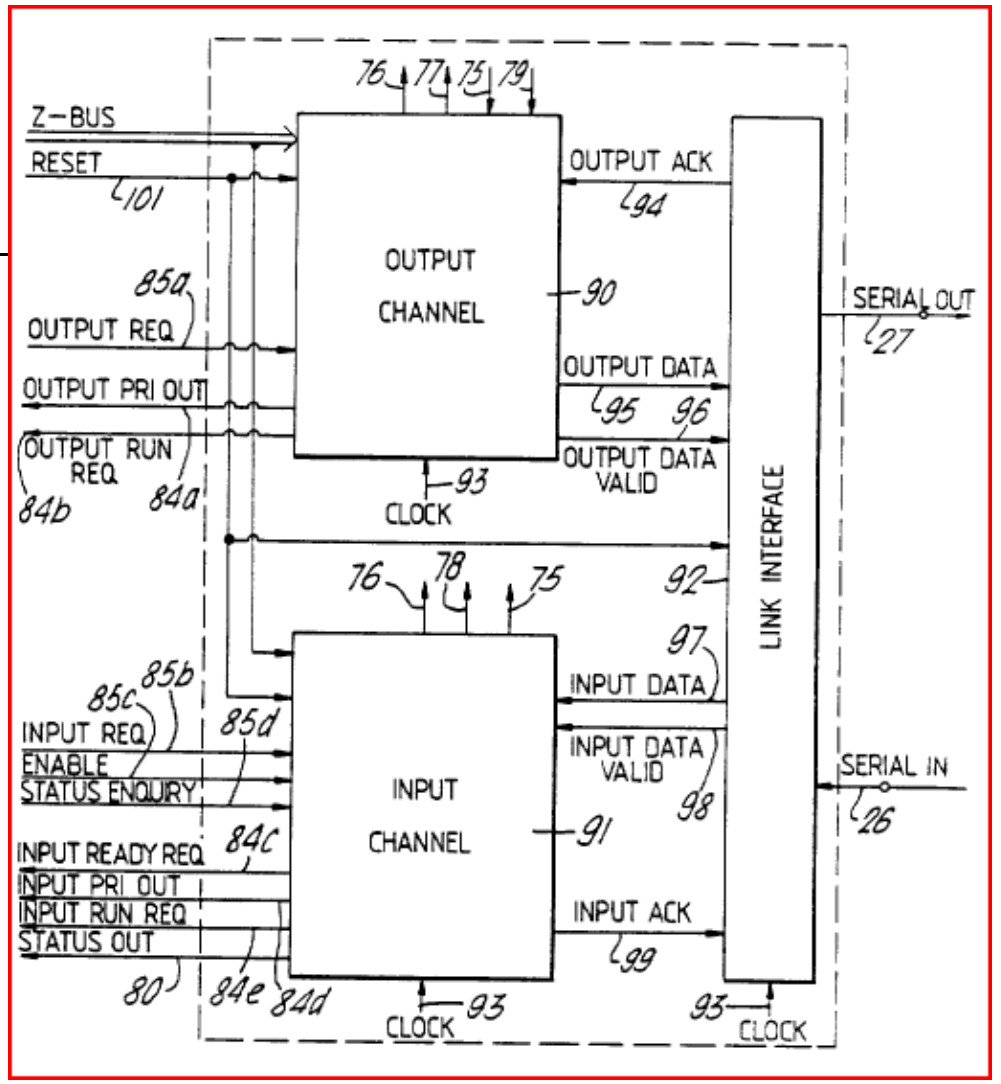
T42 Pipeline



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.

5.d Memories & Interface(s)

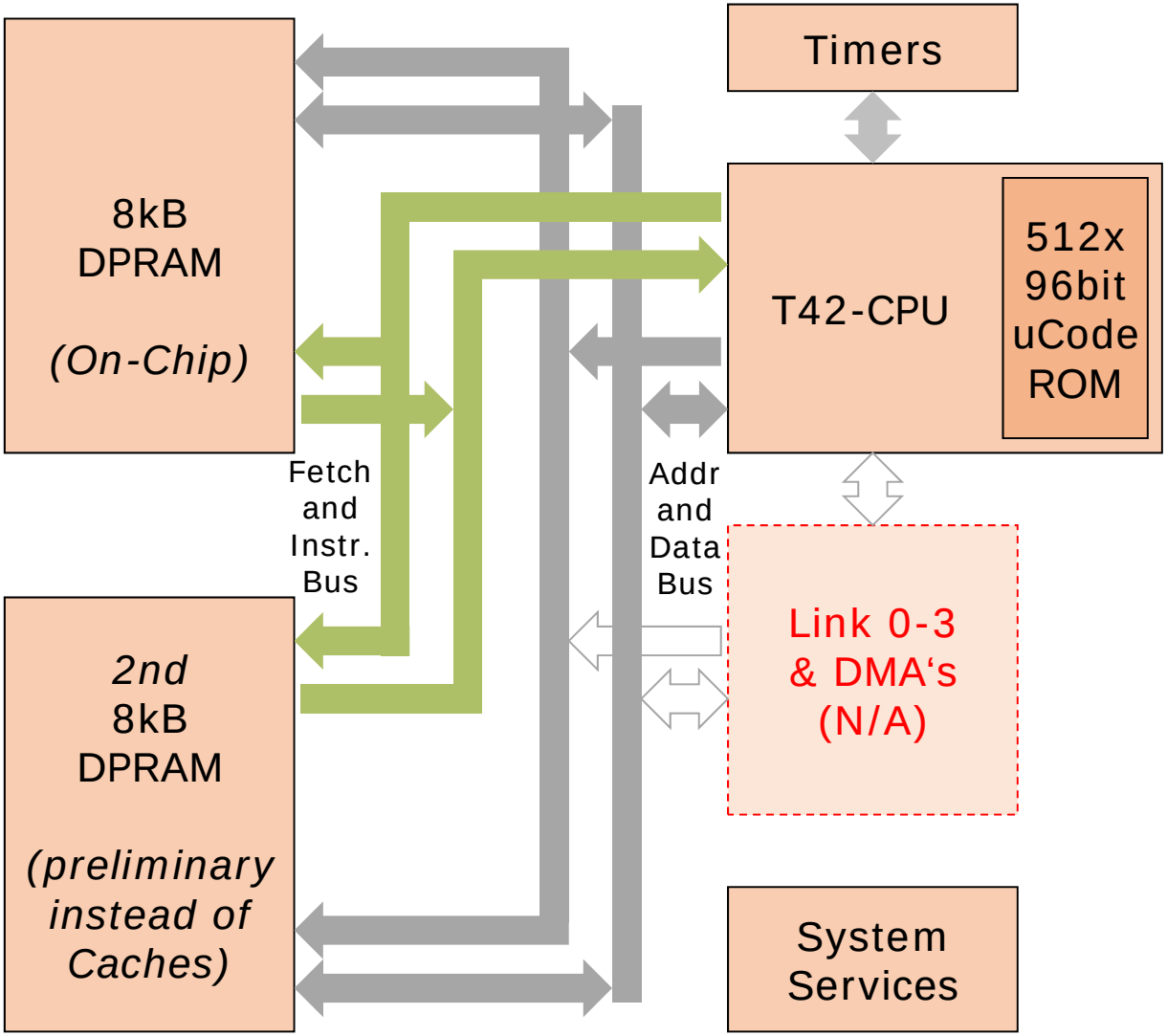
memory hierarchy:

- 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.

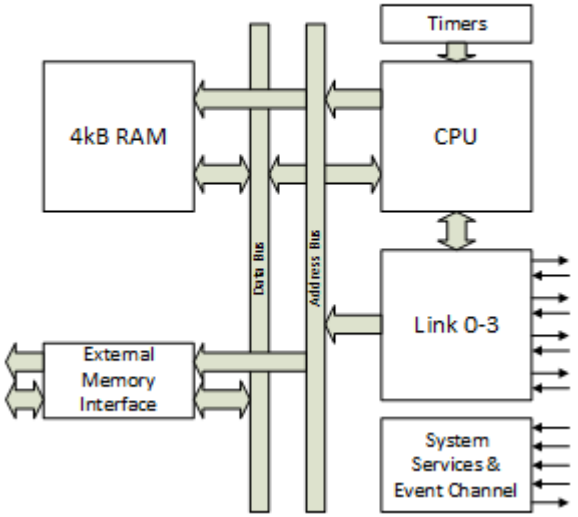
defining address and data bus structures and protocols (arbiter) for 6 consumers (rd) and 5 producers (wr):

- CPU data path (rd/wr)
- CPU pre-fetch (rd)
- 4x Link-in DMA (wr)
- 4x Link-out DMA (rd)

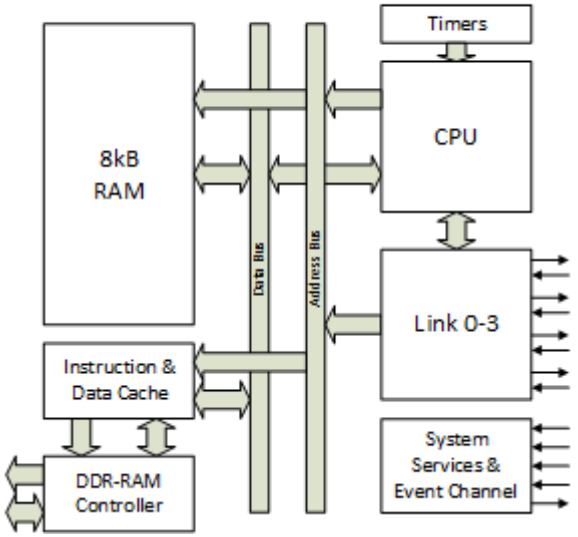
T42 Schematic



IMS-T425



T42-in-FPGA



CPU:Cache:DDRCtrl = 1:2:4

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

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

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%

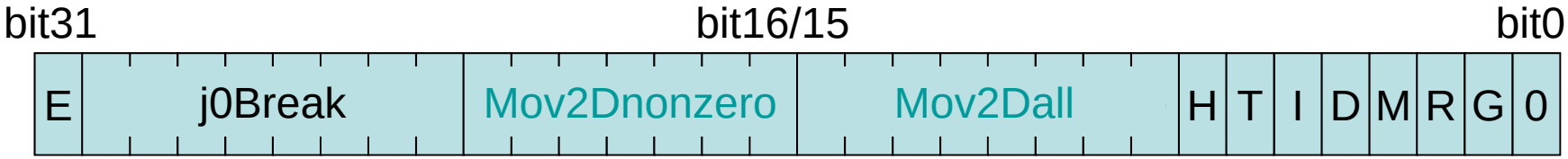
5.e System Control

- 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

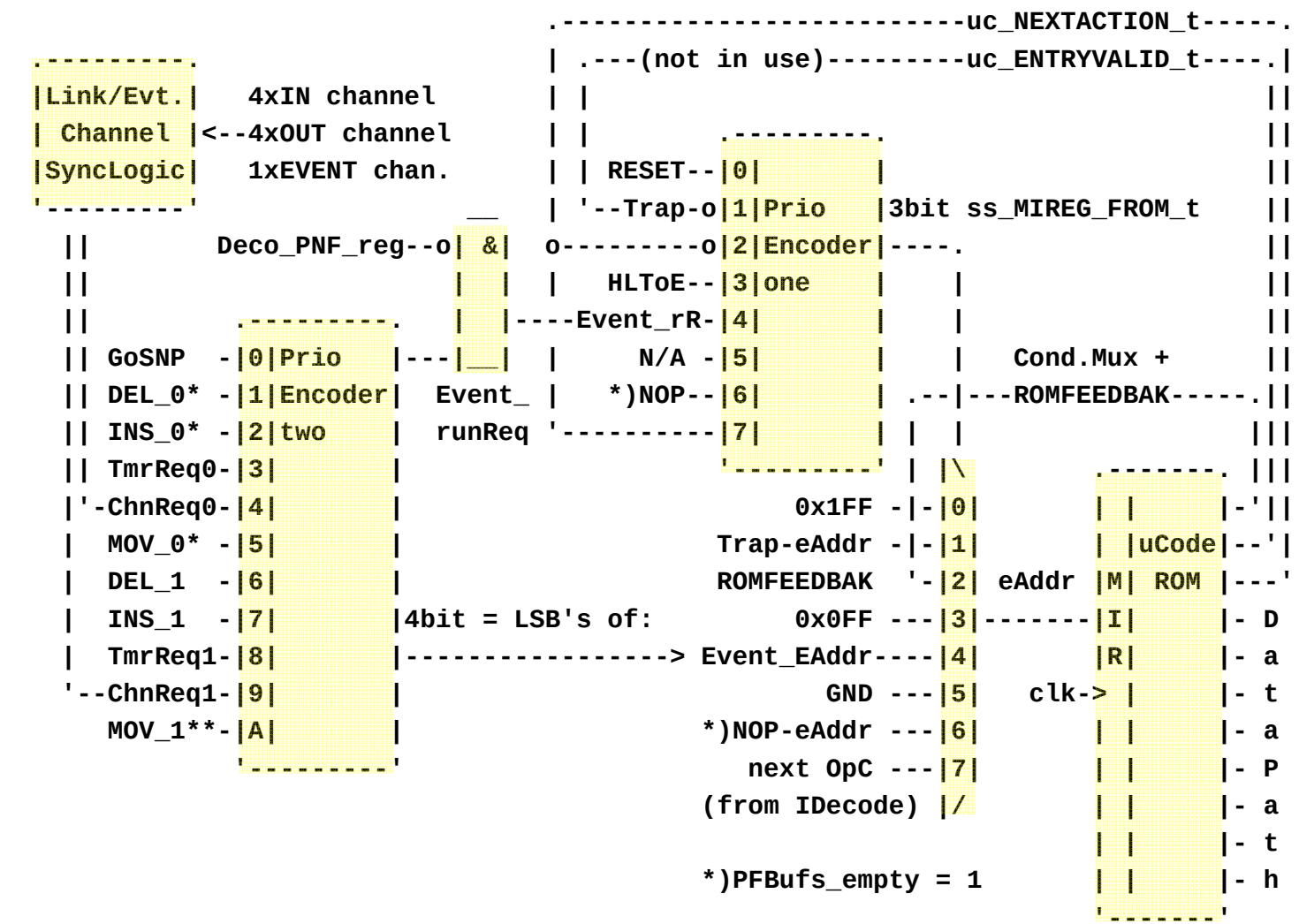
- maintain processor status register:



(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

VERSION from 08-May-2016:



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 IUT requirements.

```
; 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

TVS-1 original	input set	Extra Constants	Areg Values	Breg Values	Creg Values	NO. of TESTs	WORDs per SET	IN-WORDs
TEST.1	i32_1.bin		128	BBBBBBBB	CCCCCCCC	128	3	384
TEST.1.4	i32_1.bin	8	128	BBBBBBBB	CCCCCCCC	1.024	3	3072
TEST.2	i32_2.bin		128	128	CCCCCCCC	16.384	3	49152
TEST.3	i32_3.bin		128	128	8	131.072	3	393216
TEST.B	i32_B.bin		32	128	8	32.768	3	98304
TEST.F	i32_F.bin		64	BBBBBBBB	CCCCCCCC	64	3	192
TEST.P	i32_P.bin	14	8	72	14	112.896	**4**	451584
TEST.S	i32_S.bin		66	128	8	67.584	3	202752
						361.920		

- 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

TVS-1 (T-42)	input set	Extra Constants	Areg Values	Breg Values	Creg Values	NO. of TESTs	WORDs per SET	IN-WORDs
TEST.1	i32_1.bin		128	BBBBBBBB	CCCCCCCC	128	3	384
TEST.1.4	i32_1.bin	8	128	BBBBBBBB	CCCCCCCC	1.024	3	3072
TEST.2	i32_2.bin		32	32	CCCCCCCC	1.024	3	3072
TEST.3	i32_3.bin		32	32	8	8.192	3	24576
TEST.B	i32_B.bin		32	32	2	2.048	3	6144
TEST.F	i32_F.bin		64	BBBBBBBB	CCCCCCCC	64	3	192
TEST.P	i32_P.bin	*10*	*8*	*10*	*10*	*8.000*	**4**	32000
TEST.S	i32_S.bin		32	32	4	4.096	3	12288
						24.576		

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.)*

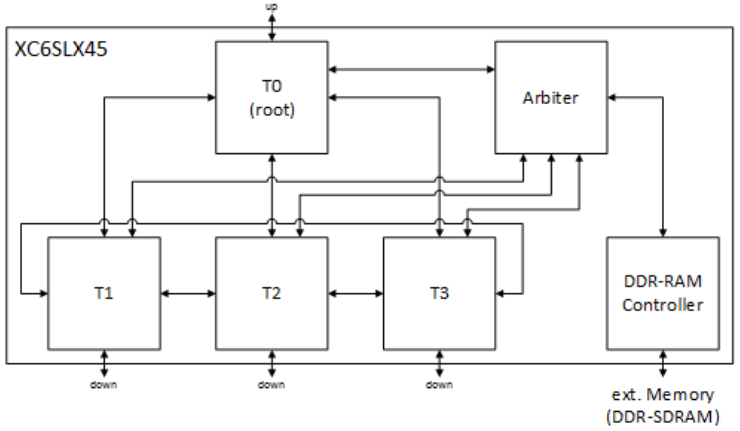
by help of TUD & POC library: *(pre-prepared)*

- 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!** cheep 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):

Softcore	Provider	Pipeline	FPGA	Clock	LUTs
Microblaze	Xilinx	3-stage	Virtex-II	80MHz	2441
Microblaze	Xilinx	5-stage	Spartan-6	195MHz	1350
Microblaze	Xilinx	5-stage	Virtex-7	330MHz	1350
Open-RISC	OpenCores	5-stage	Virtex-5	185MHz	3802
MIPS-MP32	Altera	5,6-stage	Cyclone-3	115MHz	5500
MIPS-MP32	Altera	5,6-stage	Stratix-4	300MHz	5500
NIOS-II	Altera	5-stage	Stratix-V	305MHz	3200
T42	**TUD**	**2-stage**	**Spartan-6**	**100MHz**	**4000**

7.4 Benchmark Preview

- 8-stage Transputer* can beet 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

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

T. performance road map

**) fictive designs (estimated)

Appendix

This page is intended left blank.

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 + Iptr incrementer 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 → INMOS Datasheets & Technical Notes
- About Parallel: http://www.classiccmp.org/transputer → Boards, Hardware, Software
- WoTUG Archive: http://www.wotug.org/parallel → Software, Documents, Papers
- H. Reinecke, J. Schreiner, Transputer-Leitfaden – Eine Einführung und umfassende Beschreibung, C. Hanser München 1991, ISBN 3-446-16063-9
- D.A.P. Mitchell, J.A. Thompson, G.A. Manson, G.R. Brookes, Inside the Transputer, Blackwell Scientific Publications 1990, ISBN 0-632-01689-2
- Transputer Instruction Set: A compiler writer's guide (72-TRN-11905), Prentice Hall 1988, ISBN 0-139-29100-8
- John Roberts, „Transputer Assembly Programming" 1992, ISBN 0-442-00872-4, free download: http://www.transputer.net/iset/pdf/transbook.pdf
- M.D. May, P.W. Thompson, P.H. Welch, Networks, Routers and Transputers, IOS Press 1993, ISBN 9-051-99129-0, free download: http://wotug.ukc.ac.uk/docs/nrat/book.psz.tar
- C. A. R. Hoare „Communicating Sequential Processes", 21jun2004, ISBN 01-31-53289-8, free download: http://www.usingcsp.com
- Homepage of Transputer architect David May: http://www.cs.bris.ac.uk/~dave/index.html
- Homepage of Gavin Crate's Transputer-Emulator: https://sites.google.com/site/transputeremulator

Appendix: Patents

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

- US-Pat-4704678 - INMOS 26Nov1982 https://patents.google.com/patent/US4704678A/en
  Function set for a microcomputer
- US-Pat-4724517 - INMOS 26Nov1982 https://patents.google.com/patent/US4724517A/en
  Microcomputer with prefixing functions
- US-Pat-4758948 - INMOS 19Jul1988 https://patents.google.com/patent/US4758948A/en
  Microcomputer
- US-Pat-4989133 – INMOS 29Jan1991 https://patents.google.com/patent/US4989133A/en
  System for executing time dependent processes
- US-Pat-4783734 – INMOS 08Nov1988 https://patents.google.com/patent/US4783734A/en
  Computer with variable length process communication
- US-Pat-4794526 – INMOS 27Dec1988 https://patents.google.com/patent/US4794526A/en
  Microcomputer with priority scheduling

End