

Design of a transputer Core and Implementation in an FPGA

Yutaka Ooki / Naoya Fukuchi / Chikara Fukunaga / Makoto Tanaka
Tokyo Metropolitan University

Contents

- Motivation
- Architecture of TPCORE
 - CPU
 - Mechanism of MicroCode
 - Link
- Occam 2
- How to implement parallel processes
 - PAR
 - ALT
- Performance of TPCORE
 - Verification
 - Results

Motivation

- We study in detail the architecture of transputer(T425). Because CSP has been realized well in transputer while the architecture was simple and skillful.
- We'll understand it, then we'll get hints for a new parallel processor for own purpose. As the first step, we tried to construct transputer(T425) and implement in an FPGA, we call it **TPCORE**.
- We have been having some transputers and Occam Toolset. We use them to analyze. We also use the toolset for program development.

What is TPCORE

- TPCORE has been made based on the architecture of T425 that was made by Inmos(ST Micro Electronics)
- It has the same instruction set as T425.
- Each has four bi-directional links. We can construct various network topologies.
- Only one TPCORE can also perform parallel processes.

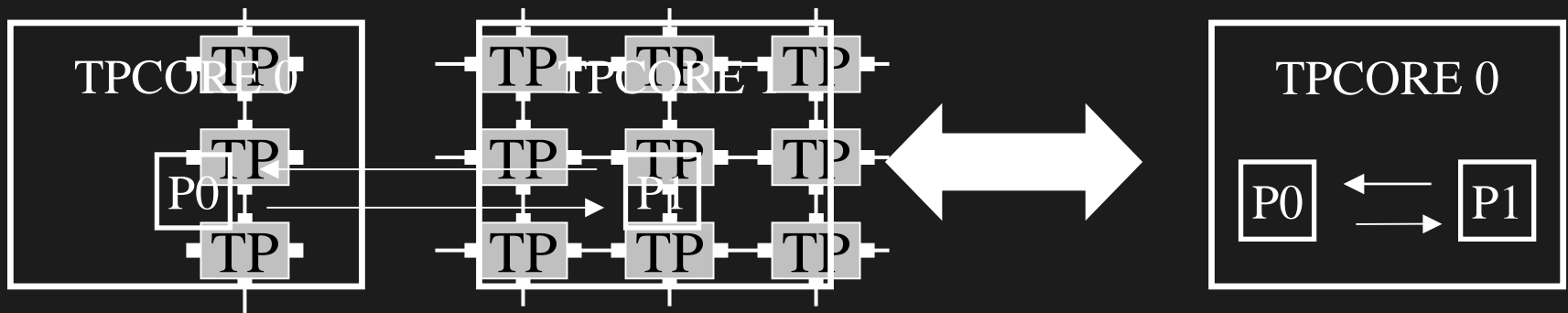
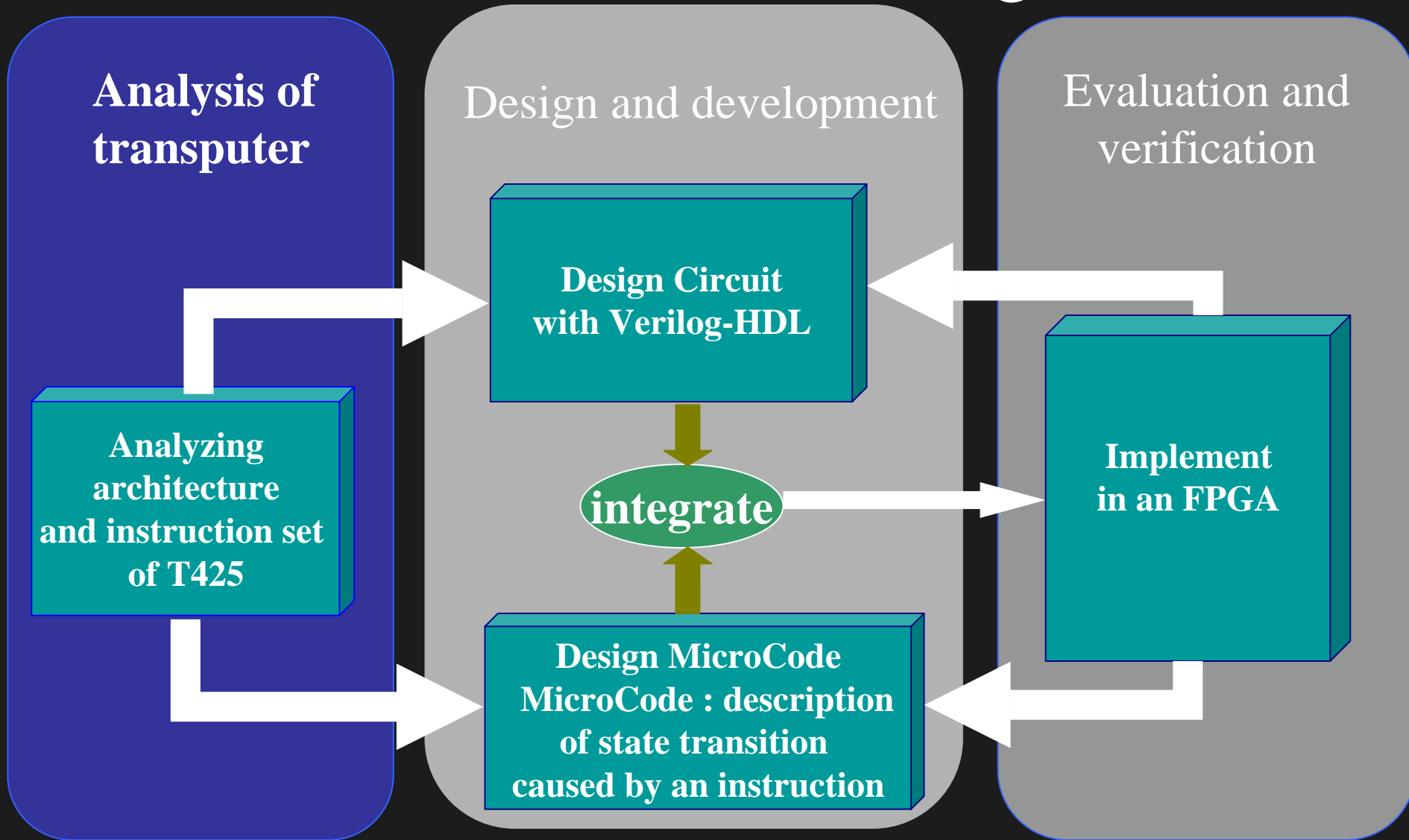


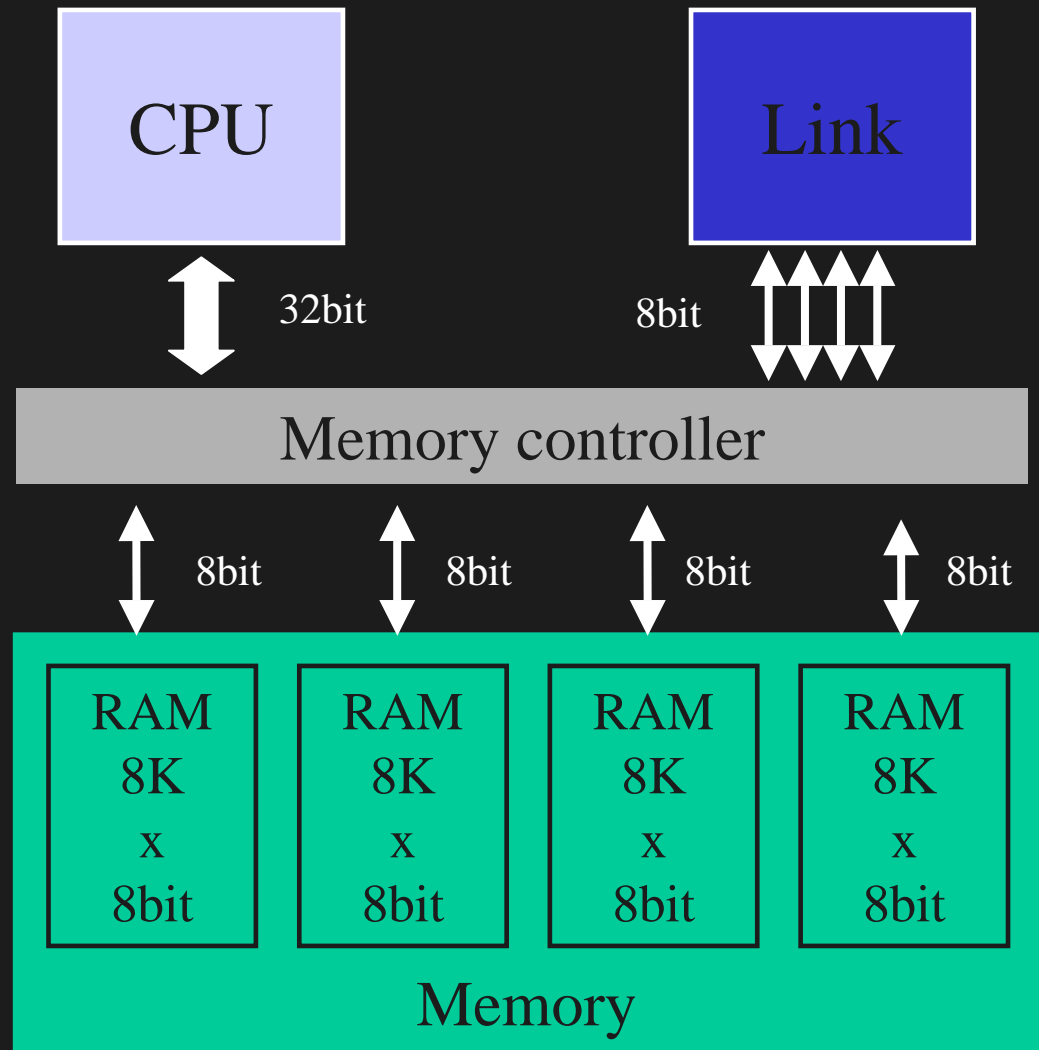
Figure : Network of transputer

Process of building

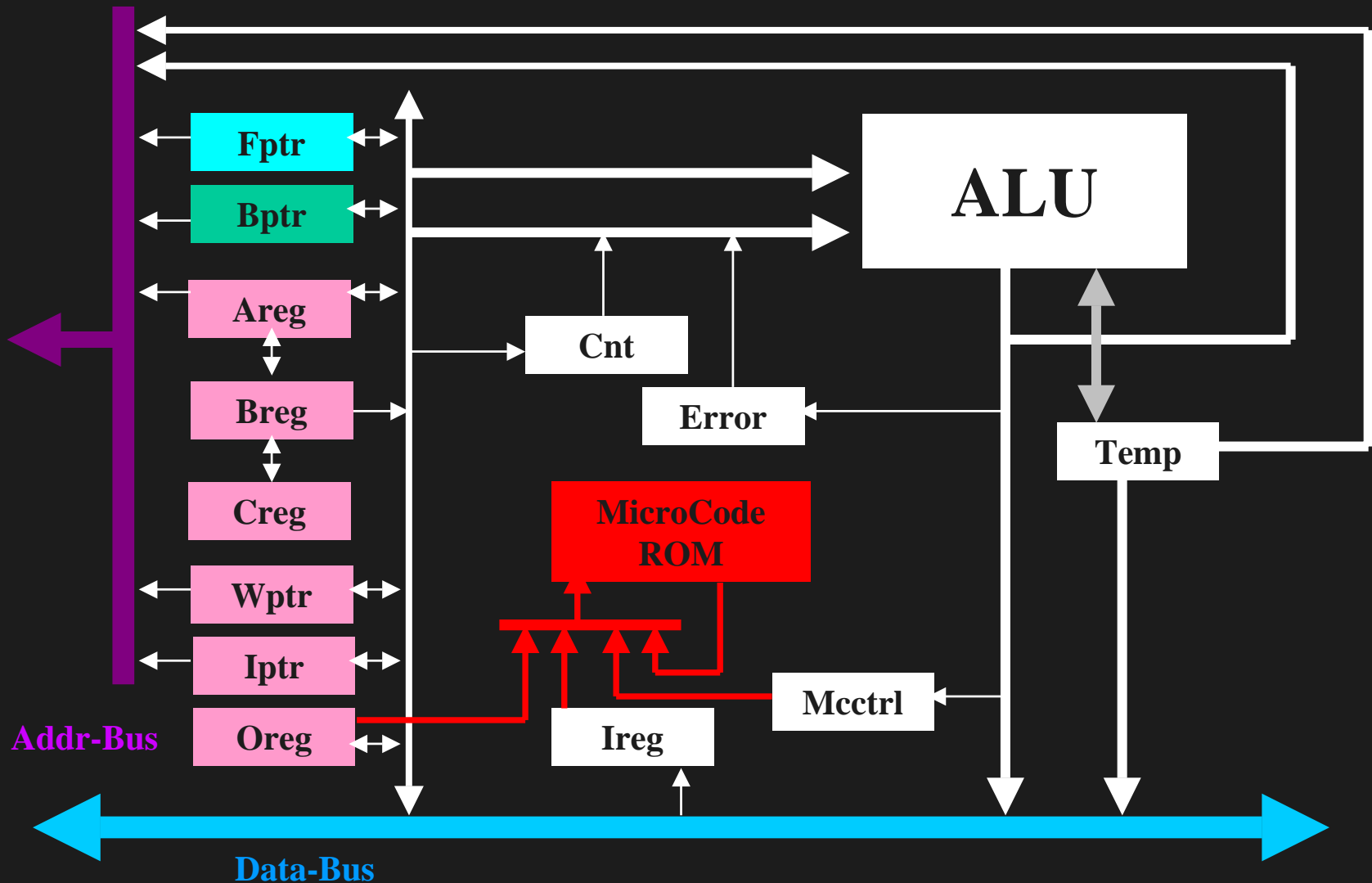


TPCORE Architecture

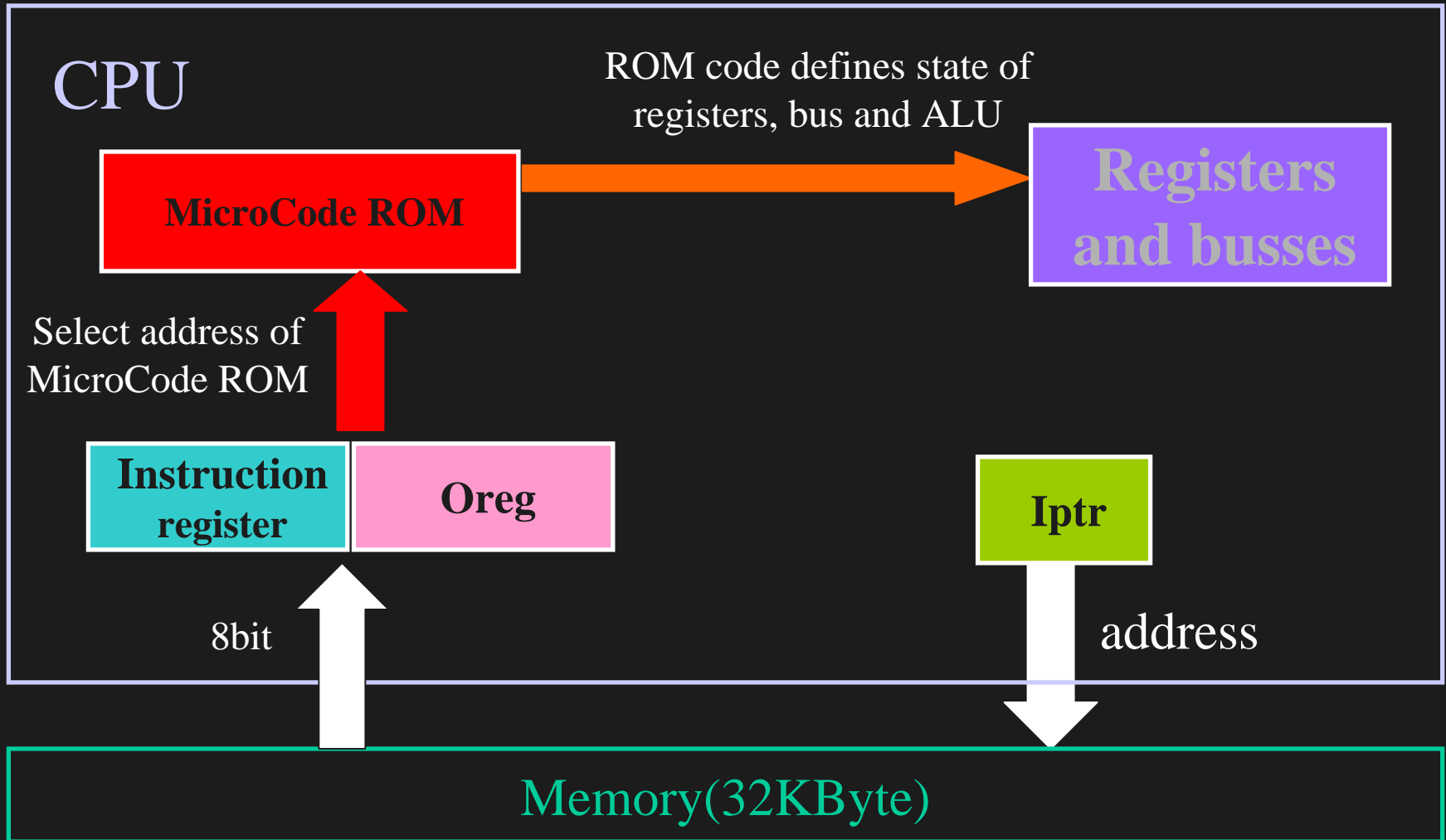
- CPU
- Link
- Memory controller
- Memory



The Schematic of CPU



Mechanism of MicroCode



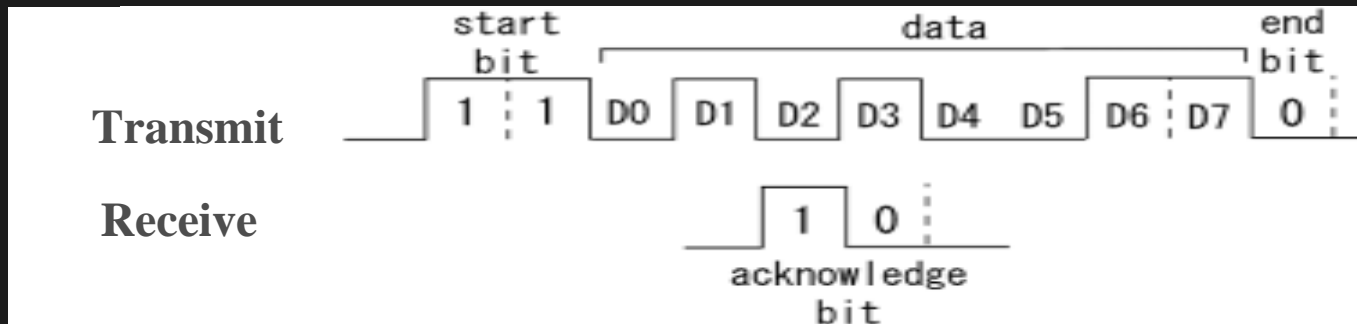
MicroCode ROM Table (64bitx512)

μ-add	OpCode	Abbreviation	3,3	3,2	3,1	3,0	2,7	2,6	2,5	2,4	2,3	2,2	2,1	2,0	1,7	1,6	1,5	1,4	1,3	1,2	1,1	1,0	0,7	0,6	0,5	0,4	0,3	0,2	0,1	0,0
			sel alub [3]	sel alub [2]	sel alub [1]	sel alub [0]	y tmp	sel ab [1]	sel ab [0]	sel db [3]	sel db [2]	sel db [1]	sel db [0]	sel pri [2]	sel pri [1]	sel pri [0]	op code [4]	op code [3]	op code [2]	op code [1]	op code [0]	next addr [8]	next addr [7]	next addr [6]	next addr [5]	next addr [4]	next addr [3]	next addr [2]	next addr [1]	next addr [0]
			27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		Inst_fetch	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
1		Inst_Decode	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2		i	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
3		ldlp	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
4		prefix	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
5		ldnl	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	
6		ldc	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	
7		ldnlp	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
8		nfix	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
9		ldl	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	
A		adc	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
B		call	0	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0
C		cj	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	
D		ajw	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
E		eqc	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	
F		stl	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
10		stnl	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1	0	
11		opr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	00	rev	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
13	01	lb	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1	1	
14	02	bsub	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
15	03	endp	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0	0	
16	04	diff	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	
17	05	add	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
18	06	gcall	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	
19	07	in	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	

*The 26 bits of the least significant bit of MicroCode

Link Interface

- TPCORE has 4 bi-directional serial Links
- Inmos Link(connect with other TPCORE)
- Transmission of identical data is kept continued until “ack” data is received.



Occam 2

- Occam is programming language based on the concept of CSP
- It enables us to write a sequential process(SEQ), parallel processes(PAR), alternative processes(ALT)
- A communication between processes is described using a channel variable. It also synchronizes between processes.

```
CHAN OF INT ch1:  
PAR  
  SEQ  
    -----  
    a := b+c  
    ch1 ! a  
    -----  
  SEQ  
    -----  
    ch1 ? x  
    x := y-z  
    -----
```

Occam program

How to execute parallel processes

- If we perform parallel processes in **single TPCORE**, we need special hardware that manages process changing

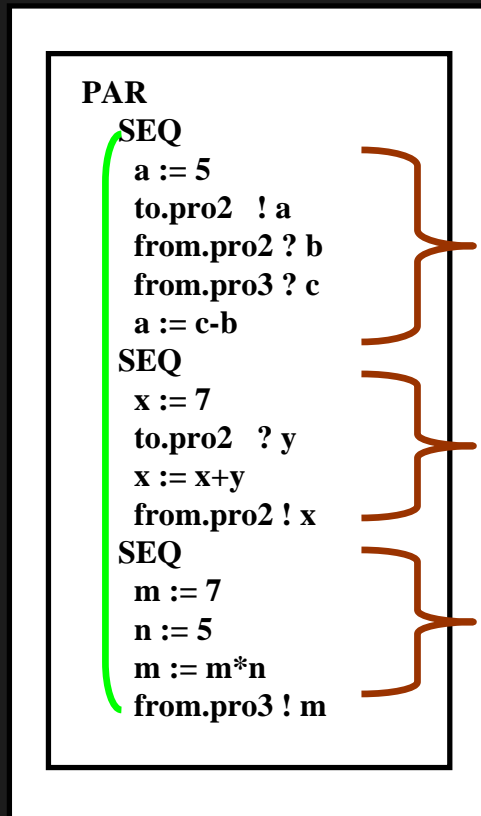


Queue(link list structure, FIFO)

- In case of **multi-TPCOREs**, an appropriate process is distributed to the specified TPCORE with occonf configuration file.

Implement PAR(single TPCORE)

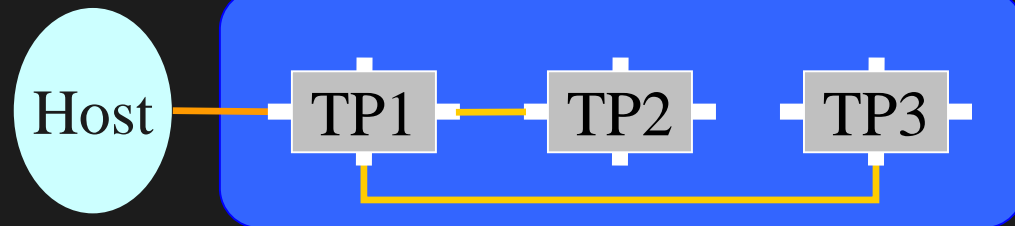
MULTI TPCORE



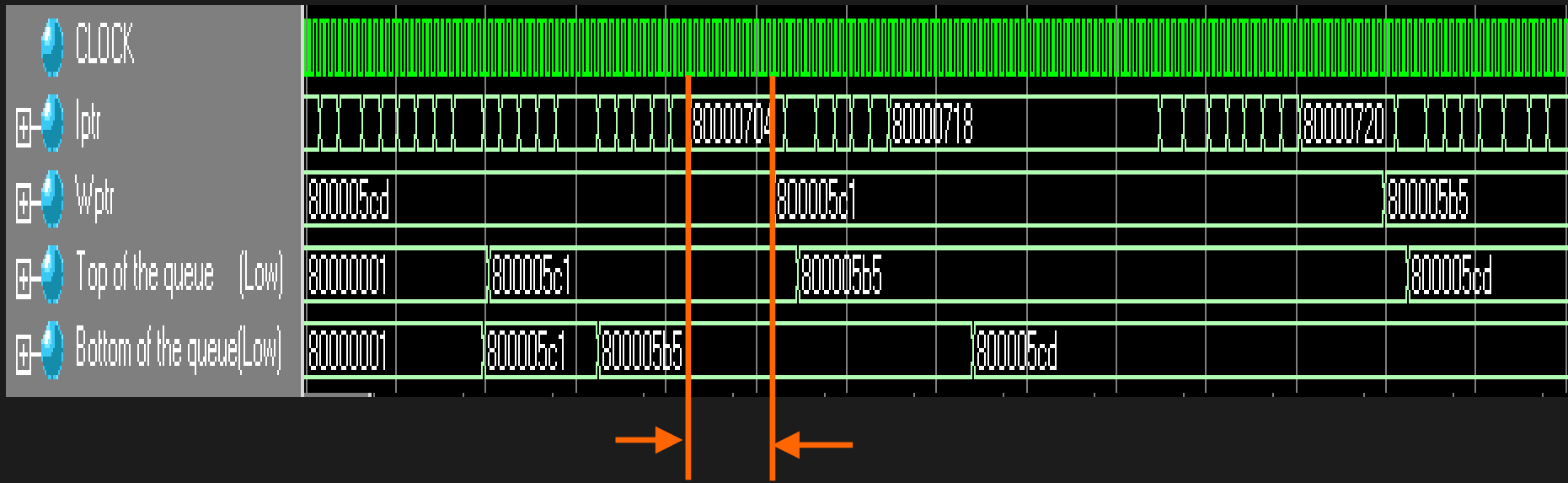
TPCORE1

TPCORE2

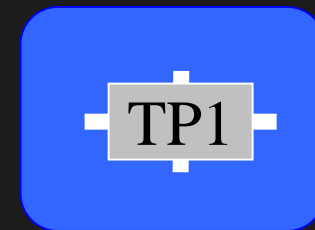
TPCORE3



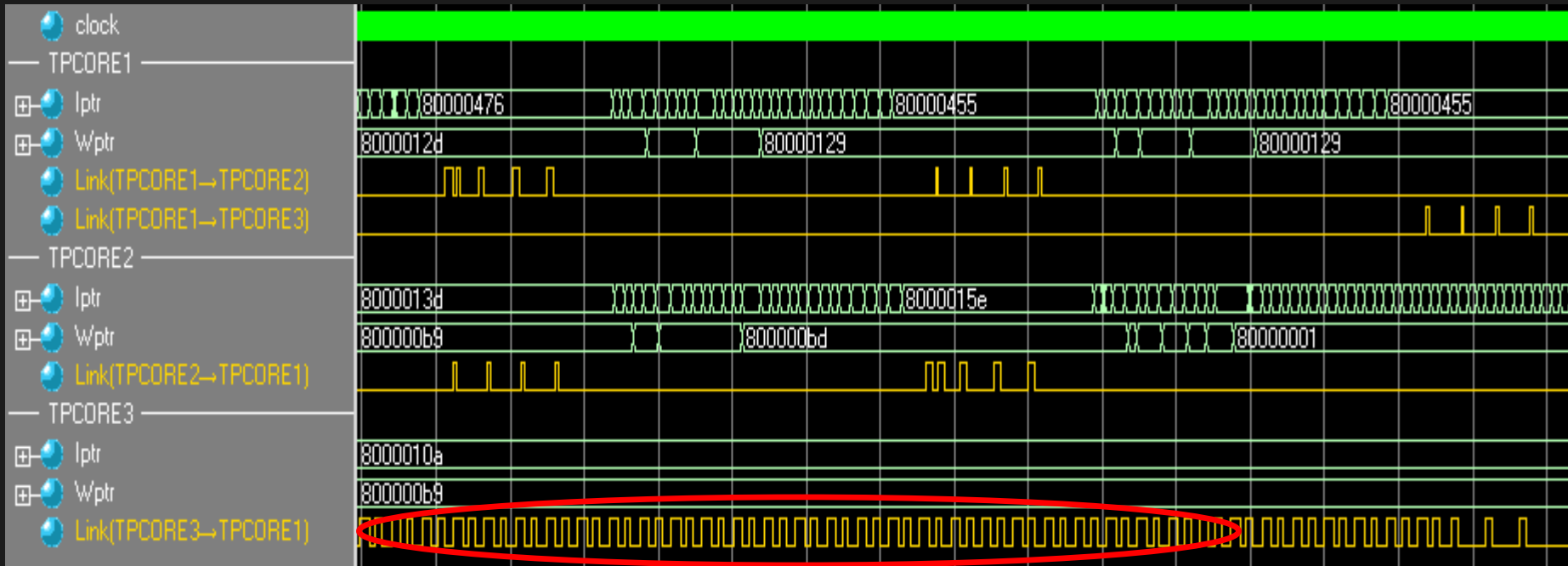
Perform PAR(single TPCORE)



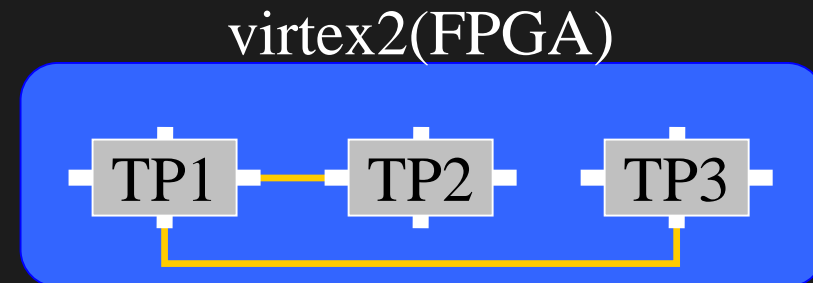
virtex2(FPGA)



Perform PAR (Multi-TPCOREs)



Sending data until "ack" come



Performance of TPCORE

	TPCORE(FPGA)	Transputer T425(ASIC)
Frequency	27MHz(31.5MHz ^{*1})	25MHz
Size of circuit	3,757 / 10,752 Slices ^{*2} (twin 7,153 : triple 9,703)	
Size of memory	32KByte/TPCORE (extendable)	4KByte(Internal RAM) 4GByte(External RAM)
Memory access	27MByte/sec	100MByte/sec (Internal RAM) 33MByte/sec (External RAM)
Link Speed	27Mbit/sec	5/10/20 Mbit/sec
Instruction set	96	103

Performance ratio of the TPCORE and the transputer

^{*1} estimative value

^{*2} Full resource of the Virtex2 that we use

Clock cycle of instructions

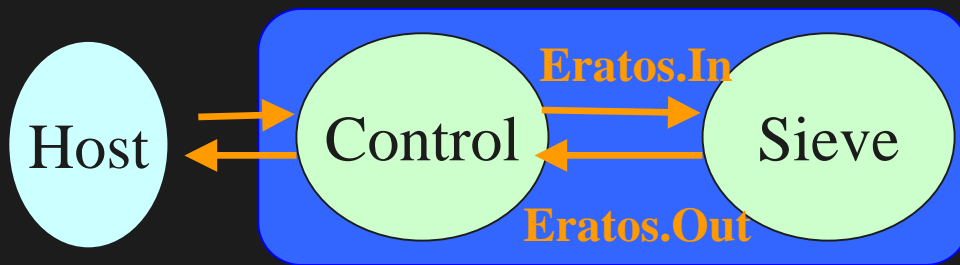
Instruction	Description	T425	TPCORE
ldc	load constant	1	1
ldnl	load non local	2	2
eqc	equal constant	2	1
pfix	prefix	1	1
gcall	gcall	4	2
wcnt	word count	5	6
in(internal)	Input mssg	16	16+ ^{*1} 7B+ ^{*2} PS
cut(internal)	Output mssg	16	16+7B+PS
altwt	Alt wait		7+PS
enbc	Enable Channel	7	8

*1 number of the top bit which contained "1" of areg

*2 Process switch

Verification of single TPCORE

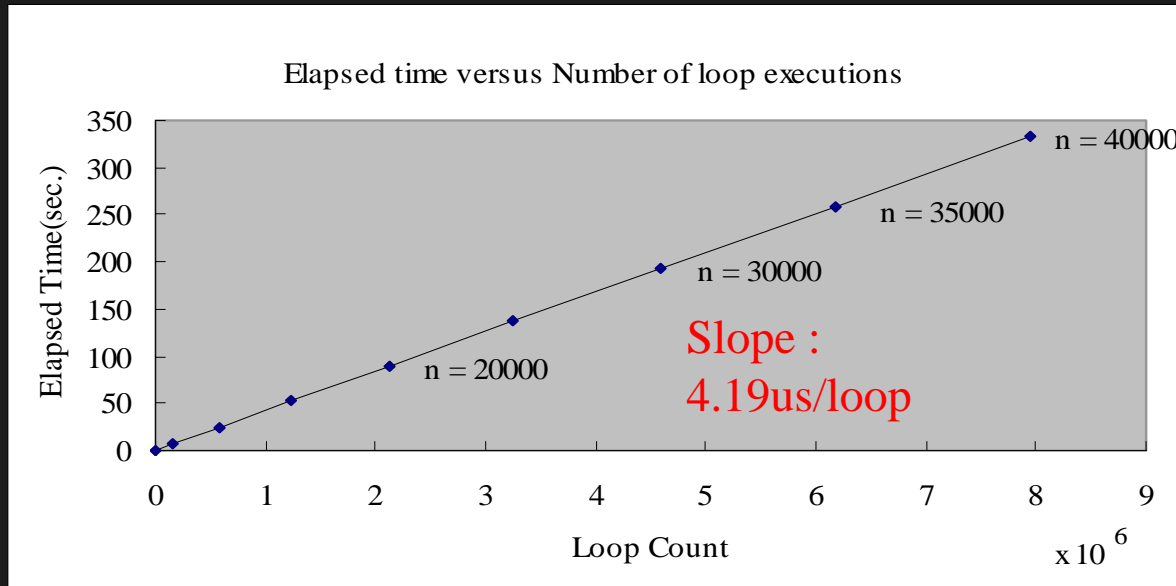
- Demonstrate examples of the occam program execution
 - The sieve of Eratosthenes
- Measure execution time



TPCORE1

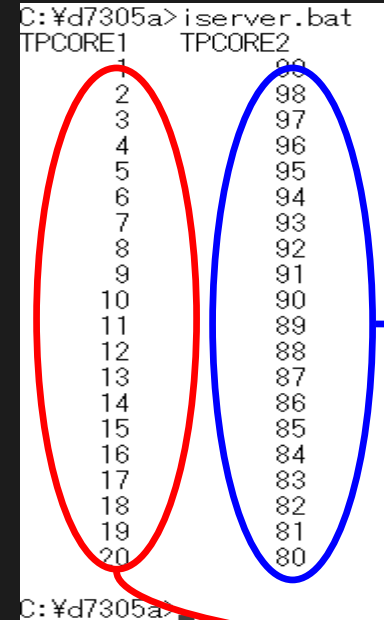
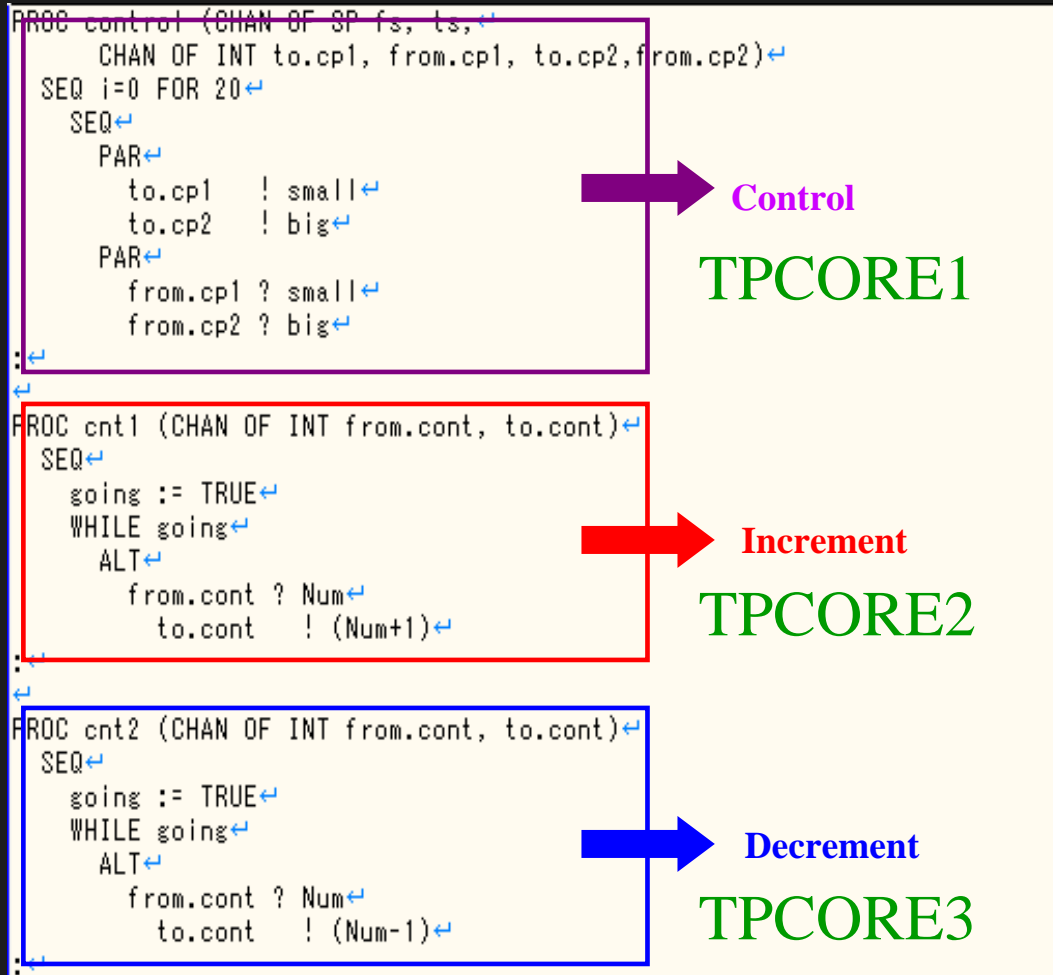
```
C:¥d7305a>iserver.bat
Please Type Number :1000
1000:
  2  3  5  7 11 13 17 19 23 29
 31 37 41 43 47 53 59 61 67 71
 73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601
607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733
739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863
877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997
C:¥d7305a>
```

Result of Sieve(single TPCORE)



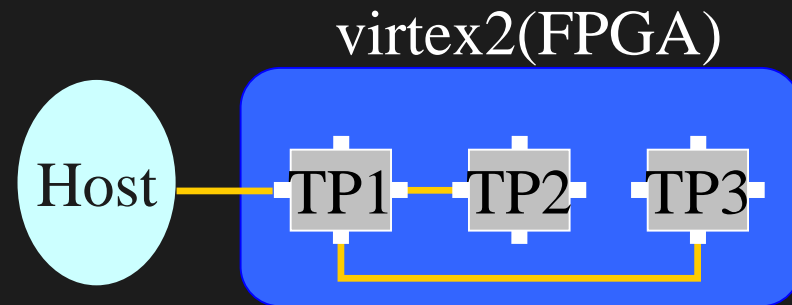
- This slope means execution time of one loop
- One loop takes 111 clock cycles, which is the result of analyzing the assembler code for this part
- Estimate : performance speed = $4.19\text{us} / 111 = 37.7\text{ns}$
clock cycle = 37.5ns

Verification of Multi-TPCORE



Output from
TPCORE2

Output from
TPCORE1



Summary

- We have made an IP core of Transputer T425, we call it TPCORE
- TPCORE can execute a program written in occam (compile and link with INMOS toolset)
- It can communicate with host computer using occam tool, that is “iserver”.
- Almost all the instructions prepared for T425 have been successfully implemented but some instructions haven't, for example “ldtimer”, “talt”.

Outlook

- We have implemented multi- TPCOREs in one FPGA. In the future we would like to configure a large TPCORE network using multi FPGA chips, and evaluate the performance.
- So far we have not implemented some instructions concerning Timer. We have to implement them.

ALT

```
CHAN OF INT ch1, ch2;  
PAR
```

```
  ALT
```

```
    ch1 ? a
```

```
      a := a+1
```

```
    ch2 ? a
```

```
      a := a-1
```

```
  SEQ
```

```
    -----
```

```
    ch1 ! b
```

```
    -----
```

```
  SEQ
```

```
    -----
```

```
    ch2 ! c
```

```
    -----
```

Occam program

Perform ALT(Multi-TPCOREs)

