Invasive Computing

Paralleles Betriebssystem einer SFB/TRR-Projektinitiative

Invasive Computing

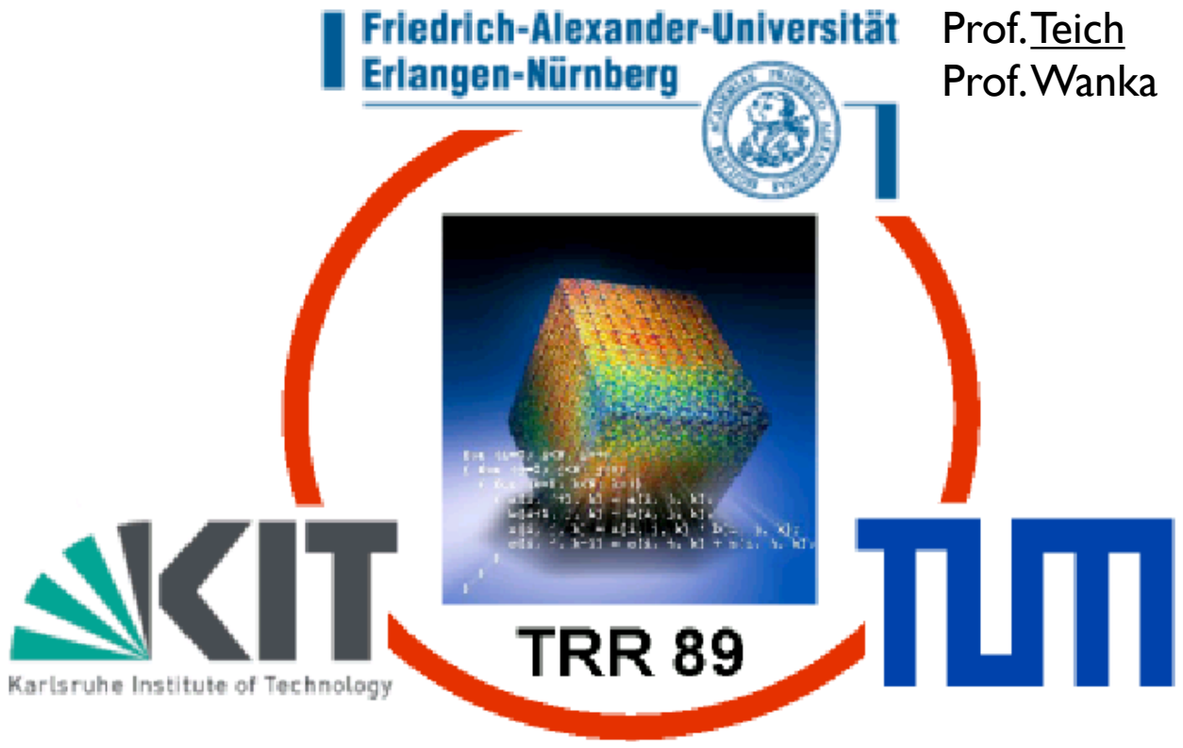
Paralleles Betriebssystem einer SFB/TRR-Projektinitiative

Wolfgang Schröder-Preikschat
FAU Erlangen-Nürnberg

SFB Transregio

SFB Transregio

Dr. Hannig
Dr. Lohmann
Prof. Schröder-Preikschat
Prof. Stamminger
Prof. Teich
Prof. Wanka



SFB Transregio

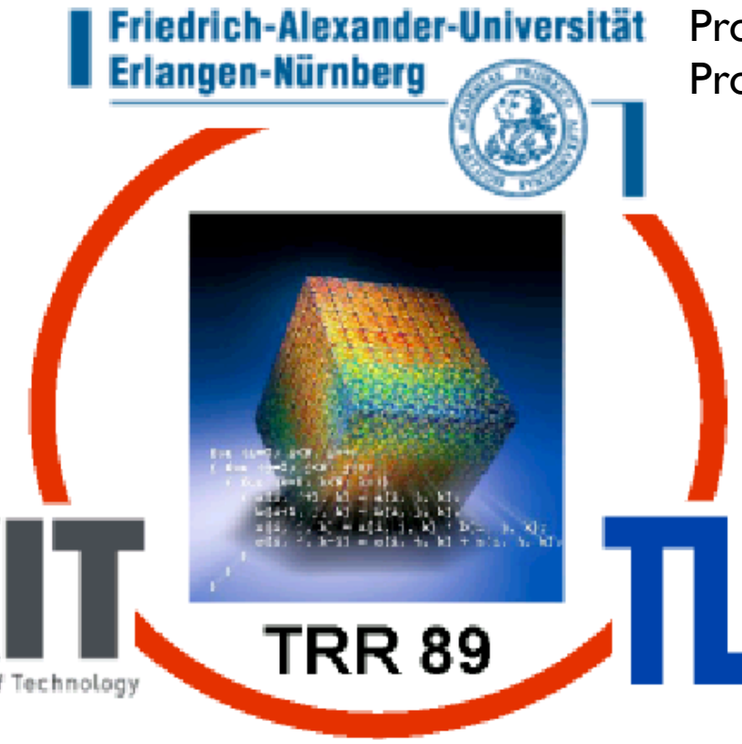
Dr. Hannig
Dr. Lohmann
Prof. Schröder-Preikschat
Prof. Stamminger
Prof. Teich
Prof. Wanka

Dr. Bader
Prof. Bungartz
Prof. Gerndt
Prof. Herkersdorf
Prof. Schlichtmann
Prof. Schmitt-Landsiedel
Prof. Stechele

SFB Transregio

Dr. Hannig
Dr. Lohmann
Prof. Schröder-Preikschat
Prof. Stamminger
Prof. Teich
Prof. Wanka

Dr. Bauer
Prof. Becker
Prof. Dillmann
Prof. Henkel
Dr. Hübner
Prof. Sanders
Prof. Snelting



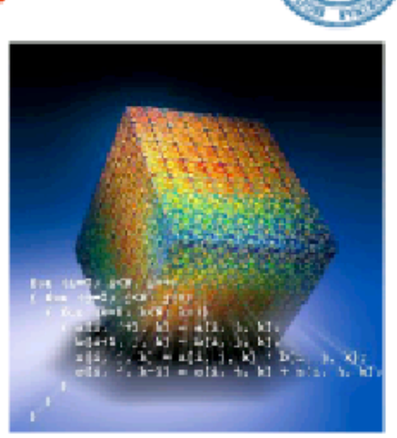
Dr. Bader
Prof. Bungartz
Prof. Gerndt
Prof. Herkersdorf
Prof. Schlichtmann
Prof. Schmitt-Landsiedel
Prof. Stechele

SFB Transregio

Dr. Hannig
Dr. Lohmann
Prof. Schröder-Preikschat
Prof. Stamminger
Prof. Teich
Prof. Wanka

Dr. Bauer
Prof. Becker
Prof. Dillmann
Prof. Henkel
Dr. Hübner
Prof. Sanders
Prof. Snelting

Friedrich-Alexander-Universität Erlangen-Nürnberg



TRR 89

Karlsruhe Institute of Technology

Dr. Bader
Prof. Bungartz
Prof. Gerndt
Prof. Herkersdorf
Prof. Schlichtmann
Prof. Schmitt-Landsiedel
Prof. Stechele

Acronym: InvasIC

Acronym: InvasIC

Investigation of a new paradigm of parallel computing

Acronym: InvasIC

Investigation of a new paradigm of parallel computing

- *through introduction of resource-aware language, programming and operating system support*

Acronym: InvasIC

Investigation of a new paradigm of parallel computing

- *through introduction of resource-aware language, programming and operating system support*
- *and through dynamic and distributed allocation and reconfiguration of processor, interconnect, and memory resources*

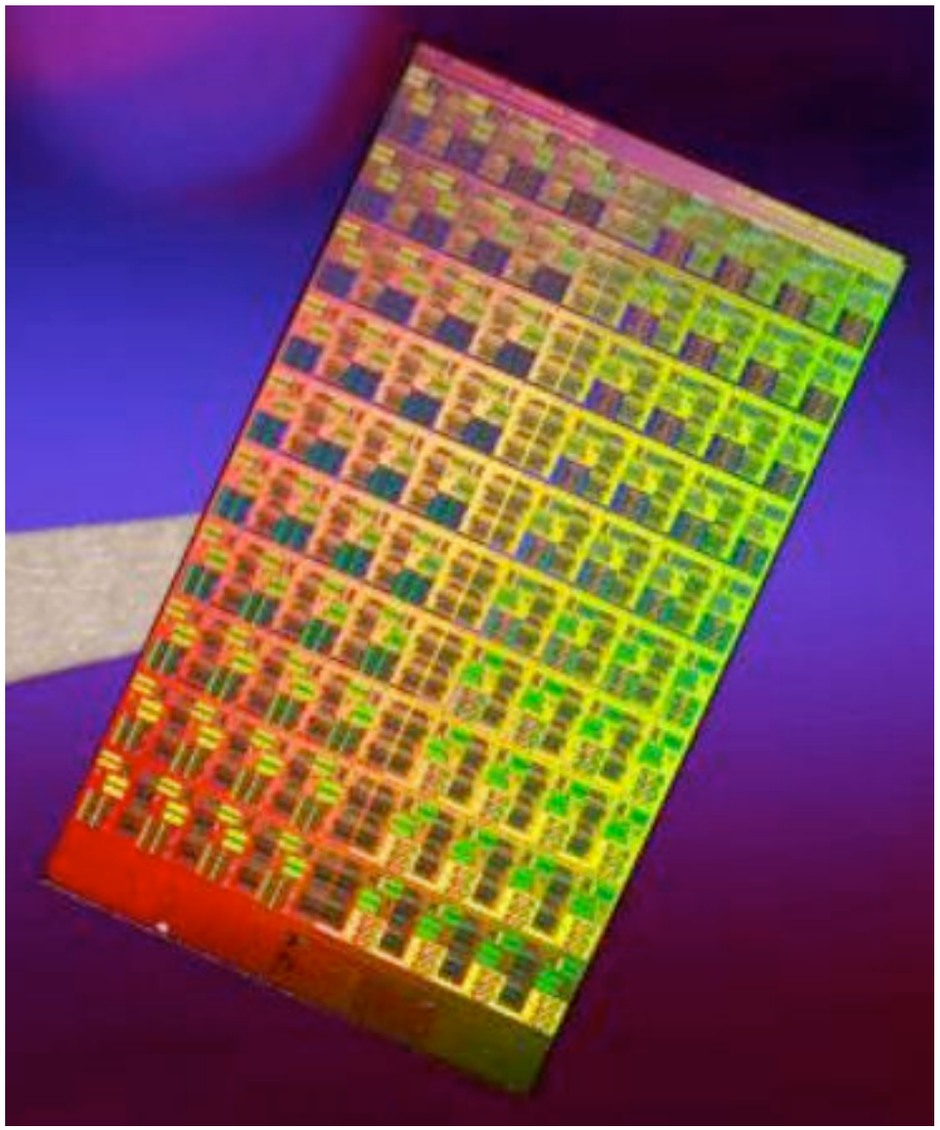
Acronym: InvasIC

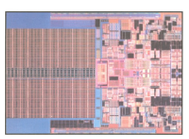
Investigation of a new paradigm of parallel computing

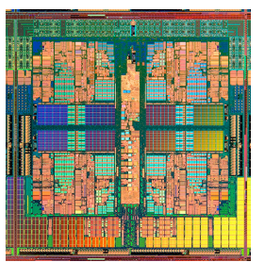
- *through introduction of resource-aware language, programming and operating system support*
- *and through dynamic and distributed allocation and reconfiguration of processor, interconnect, and memory resources*
- *with particular focus on Multiprocessor System-on-Chip (SoC) systems for the years 2020 and beyond.*

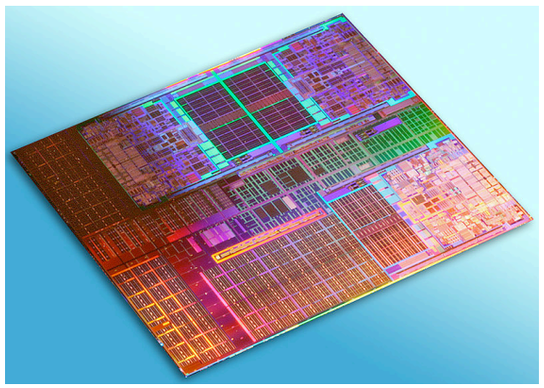
Parallele Systeme

Parallele Systeme

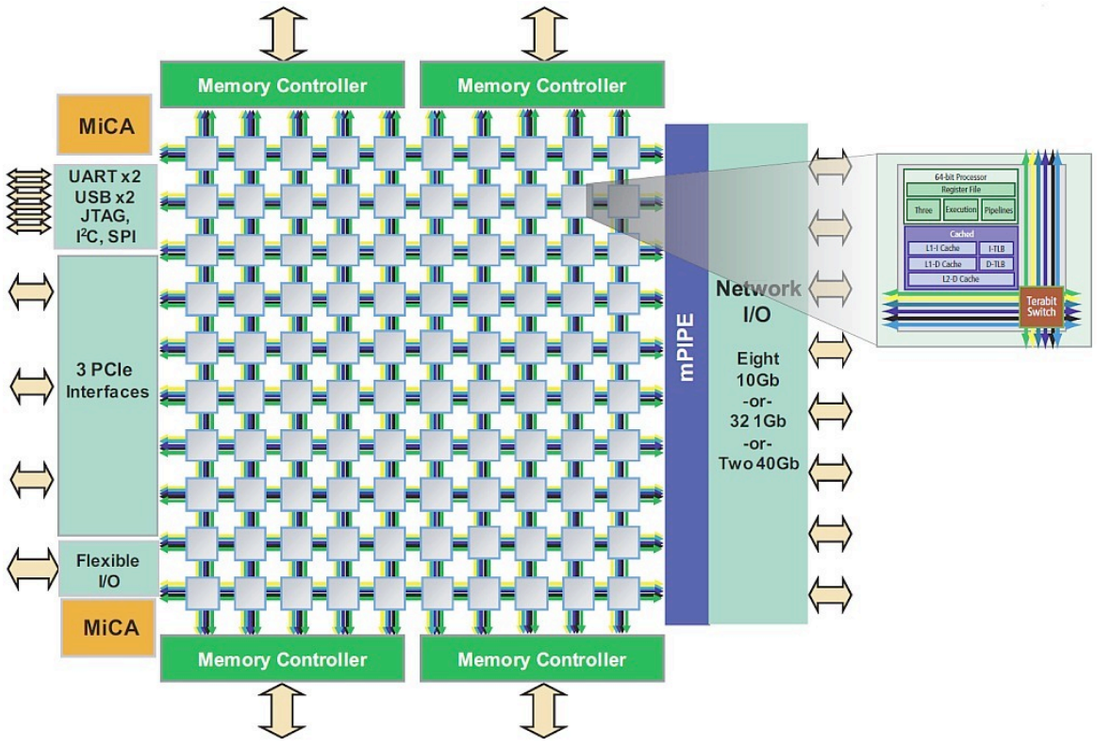


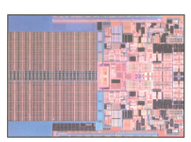


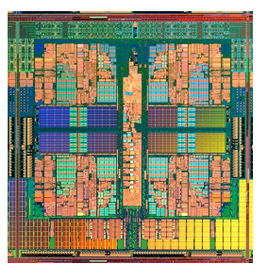


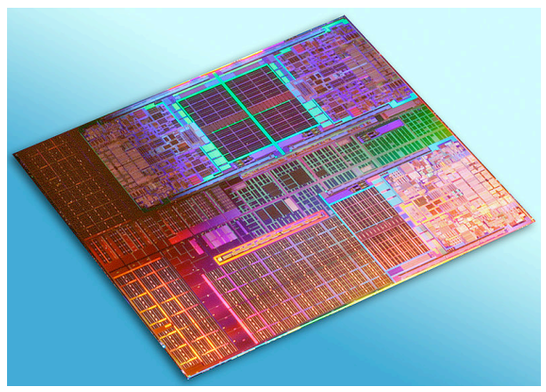


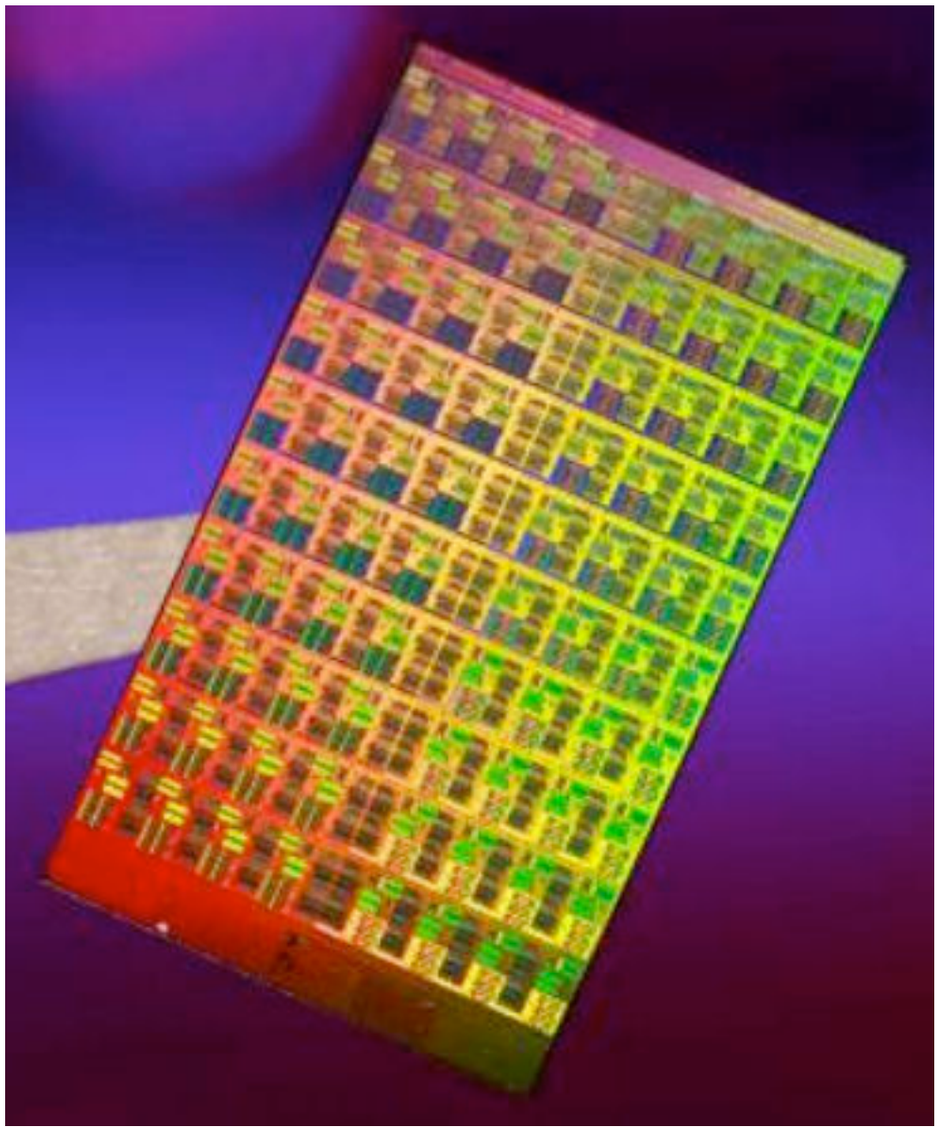
Parallele Systeme



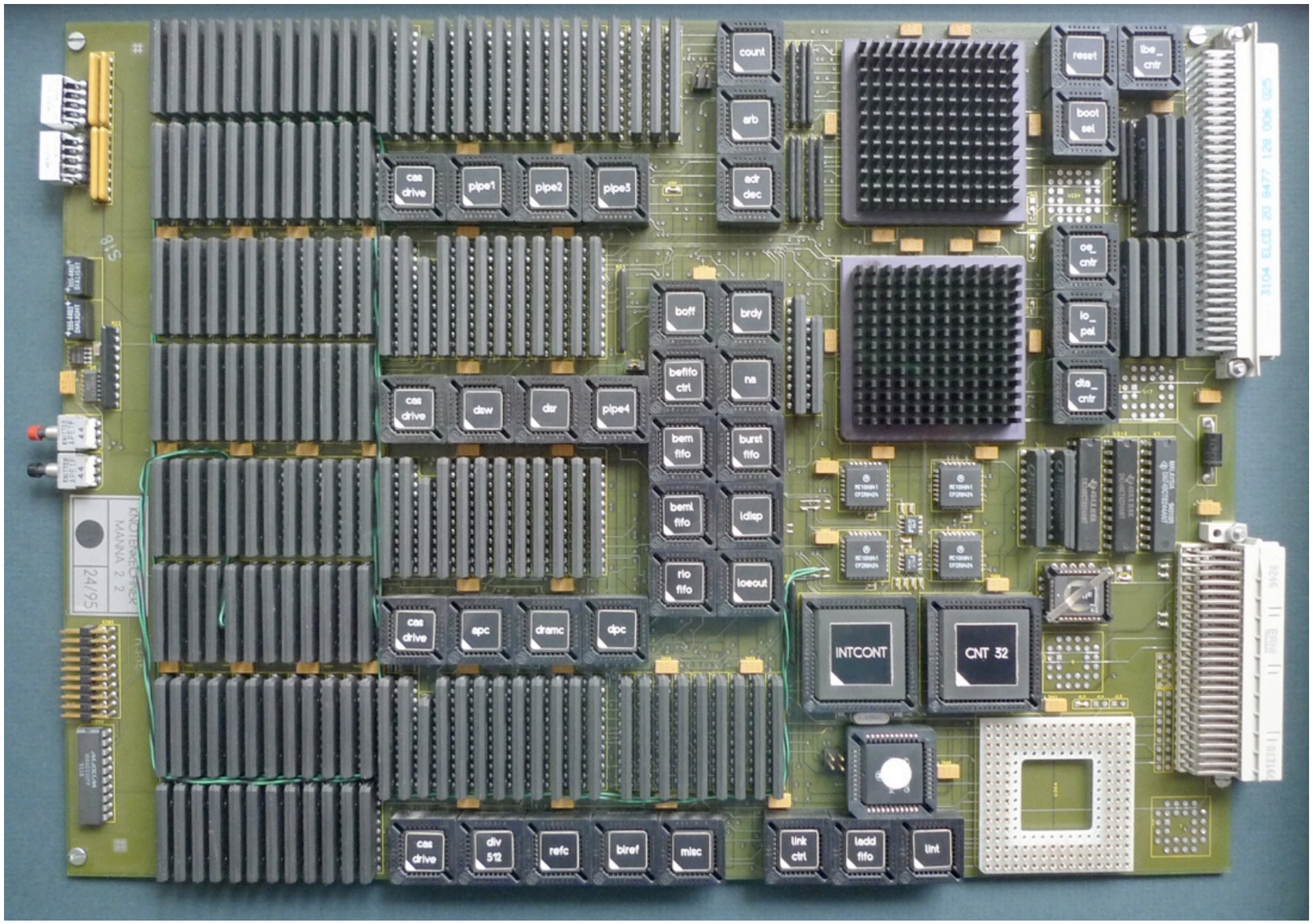


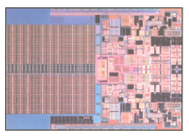






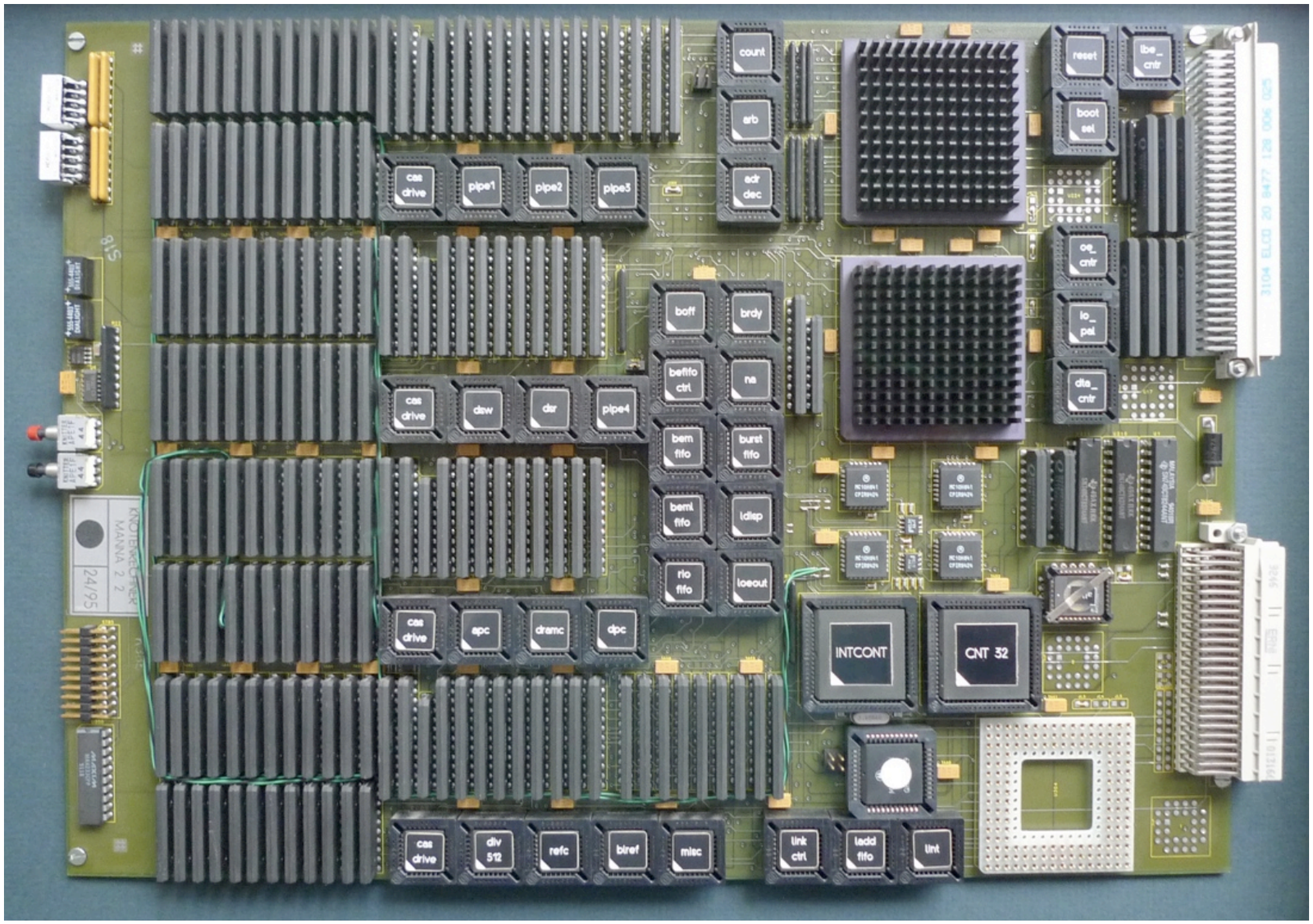
Déjà vu


count
arb
reset
boot sel
cas drive
pipe1
pipe2
pipe3
adr dec
boff
brdy
beffifo ctrl
na
cas drive
daw
dar
pipe4
burst fifo
bem fifo
bemi fifo
idlap
rio fifo
loeout
cas drive
apc
dramc
dpc
INTCONT
CNT 32
cas drive
div 512
refc
biref
misc
link ctrl
ladd fifo
lint

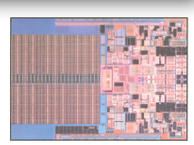


Déjà vu

Dualprozessor



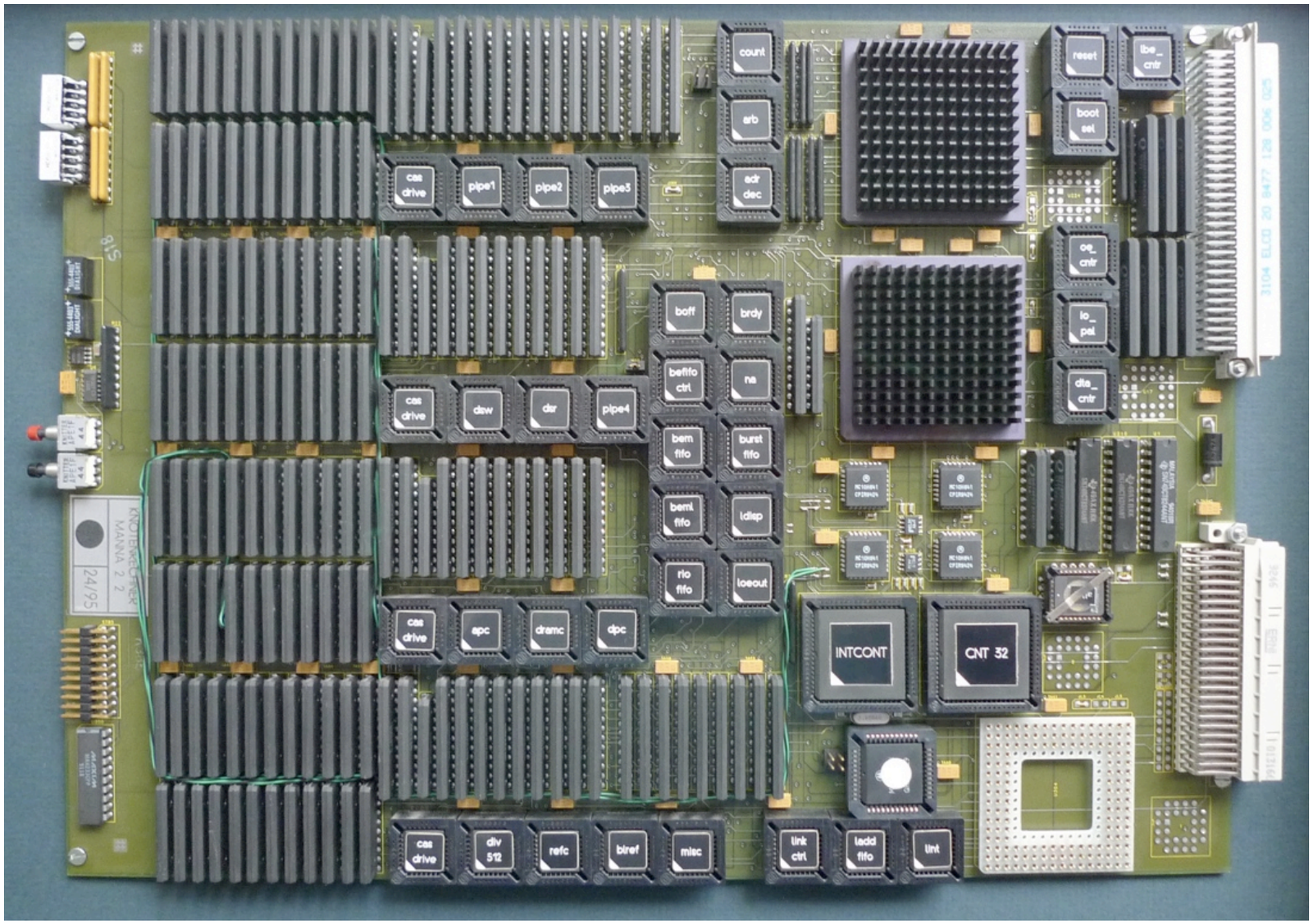
Dualcore



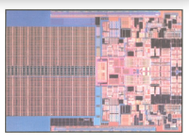
Déjà vu

SUPRENUM

Dualprozessor

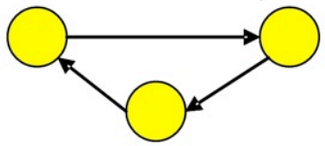


Dualcore



Ebenen von Parallelität

process-level, thread-level



Hw + Sw Control

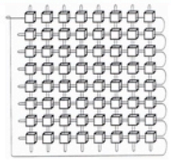
Multi-Core

loop-level

```
FOR i=0 TO N DO
    FOR j=0 TO M DO
        ...
```

Hw-Ctrl.+ Func.

Processor Array

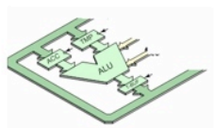


instruction-level

```
ADD R1, R2, R3
MUL R4, R1, $4
JMP $42
```

Hw-Ctrl. / VLIW

FUs

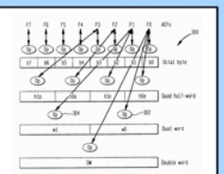


word-level, bit-level

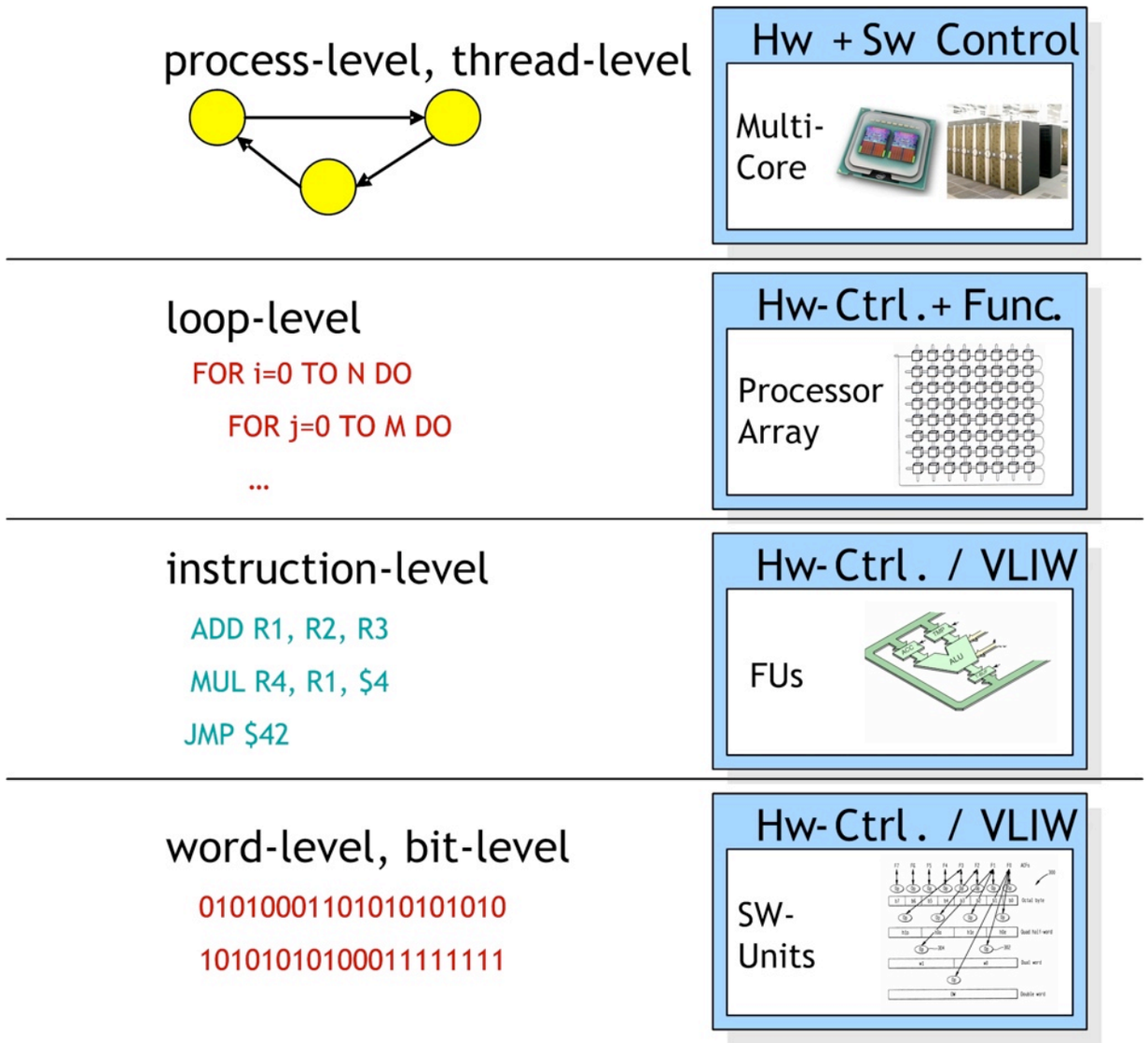
```
01010001101010101010
10101010100011111111
```

Hw-Ctrl. / VLIW

SW-Units



Ebenen von Parallelität



„*Resource-Aware*“

„*Resource-Aware*“

1. invadieren (*invade*)
   - Ressourcen anfordern/reservieren

„*Resource-Aware*“

1. invadieren (*invade*)
   - Ressourcen anfordern/reservieren
2. infizieren (*infect*)
   - mit den erhaltenen Ressourcen umgehen

„*Resource-Aware*“

1. invadieren (*invade*)
   - Ressourcen anfordern/reservieren
2. infizieren (*infect*)
   - mit den erhaltenen Ressourcen umgehen
3. zurückziehen (*retreat*)
   - invadierte Ressourcen befreien

„*Resource-Aware*“

1. invadieren (*invade*)
   - Ressourcen anfordern/reservieren
2. infizieren (*infect*)
   - mit den erhaltenen Ressourcen umgehen
3. zurückziehen (*retreat*)
   - invadierte Ressourcen befreien

Invasionseinheiten

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

1. Kandidat (*candidate*)

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

1. Kandidat (*candidate*)
2. Exemplar (*instance*)

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

1. Kandidat (*candidate*)
2. Exemplar (*instance*)
3. Inkarnation (*incarnation*)

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

1. Kandidat (*candidate*)
2. Exemplar (*instance*)
3. Inkarnation (*incarnation*)
4. Ausführung (*execution*)

Invasionseinheiten

„invasive-let“
i-let

Program section being aware of potential parallel execution

1. Kandidat (*candidate*)
2. Exemplar (*instance*)
3. Inkarnation (*incarnation*)
4. Ausführung (*execution*)

Systemabstraktionen

Systemabstraktionen

1. Anspruch (*claim*)
   - von (Hardware-) Ressourcen
   - zur parallelen *i-let* Ausführung

Systemabstraktionen

1. Anspruch (*claim*)
   - von (Hardware-) Ressourcen
   - zur parallelen *i-let* Ausführung
2. Gespann (*team*)
   - von (Programm-) Kontrollflüssen
   - von *i-let* Inkarnationen/Ausführungen

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

i-let Exemplar

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

i-let Exemplar

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

i-let Exemplar

Programmiermodell

```
claim = invade(type, quantity, properties);
if (!useful(claim)) {
   /* Retry with alternate claim setting or algorithm. */
}

team = assort(claim, code, data);
if (!viable(team)) {
   /* Retry with alternate team setting or fail. */
}

infect(claim, team);     /* employ resource(s) */
retreat(claim);          /* clean-up of resource(s) */
```

i-let Exemplar

Invasives Sortieren

```
claim = invade(SMP, 42, JOIN);
if (parallelism(claim) == 1)
   /* only one processing element: sort serial */
else {
   /* 1 < n <= 42 processing elements */

   team = assort(claim, code, data);   /* create workload */
   if (viable(team))
      infect(claim, team);             /* sort in parallel */

   retreat(claim);              /* await join, clean-up */
}
```

Invasives Ray Tracing

```
if (all pixels see same object) {
   claim = invade(SIMD, ∞, UMA);
   if (useful(claim))   /* n > 1 processing elements */
      team = assort(claim, code(SIMD), data(SIMD));
} else {
   claim = invade(MIMD, 42, VSM);
   if (useful(claim))   /* 1 < n <= 42 processing elements */
      team = assort(claim, code(MIMD), data(MIMD));
}

if (viable(team))
   infect(claim, team);     /* run in parallel */

retreat(claim);             /* clean-up */
```

Forschungsprogramm

Forschungsprogramm

1. Phase (2010-2013): **Wettstreitigkeit**
   - Vorbeugung, Vermeidung, Verringerung
   - Latenzverbergung

Forschungsprogramm

1. Phase (2010-2013): **Wettstreitigkeit**
   - Vorbeugung, Vermeidung, Verringerung
   - Latenzverbergung
2. Phase (2014-2017): **Virtualisierung**

Forschungsprogramm

1. Phase (2010-2013): **Wettstreitigkeit**
   - Vorbeugung, Vermeidung, Verringerung
   - Latenzverbergung
2. Phase (2014-2017): **Virtualisierung**
3. Phase (2018-2021): **Heterogenität**

Forschungsthemen (Phase I)

Forschungsthemen (Phase I)

1. nebenläufige Laufzeitexekutive
   - funktional verteilt, echtzeitfähig, NBS

Forschungsthemen (Phase I)

1. nebenläufige Laufzeitexekutive
   - funktional verteilt, echtzeitfähig, NBS
2. vertikale Konfigurierung
   - querschneidend statisch/dynamisch, AOP

Forschungsthemen (Phase I)

1. nebenläufige Laufzeitexekutive
   - funktional verteilt, echtzeitfähig, NBS
2. vertikale Konfigurierung
   - querschneidend statisch/dynamisch, AOP
3. agentenbasierte Ressourcenvergabe
   - verteilt, Verhandlungs-/Anreizmodelle

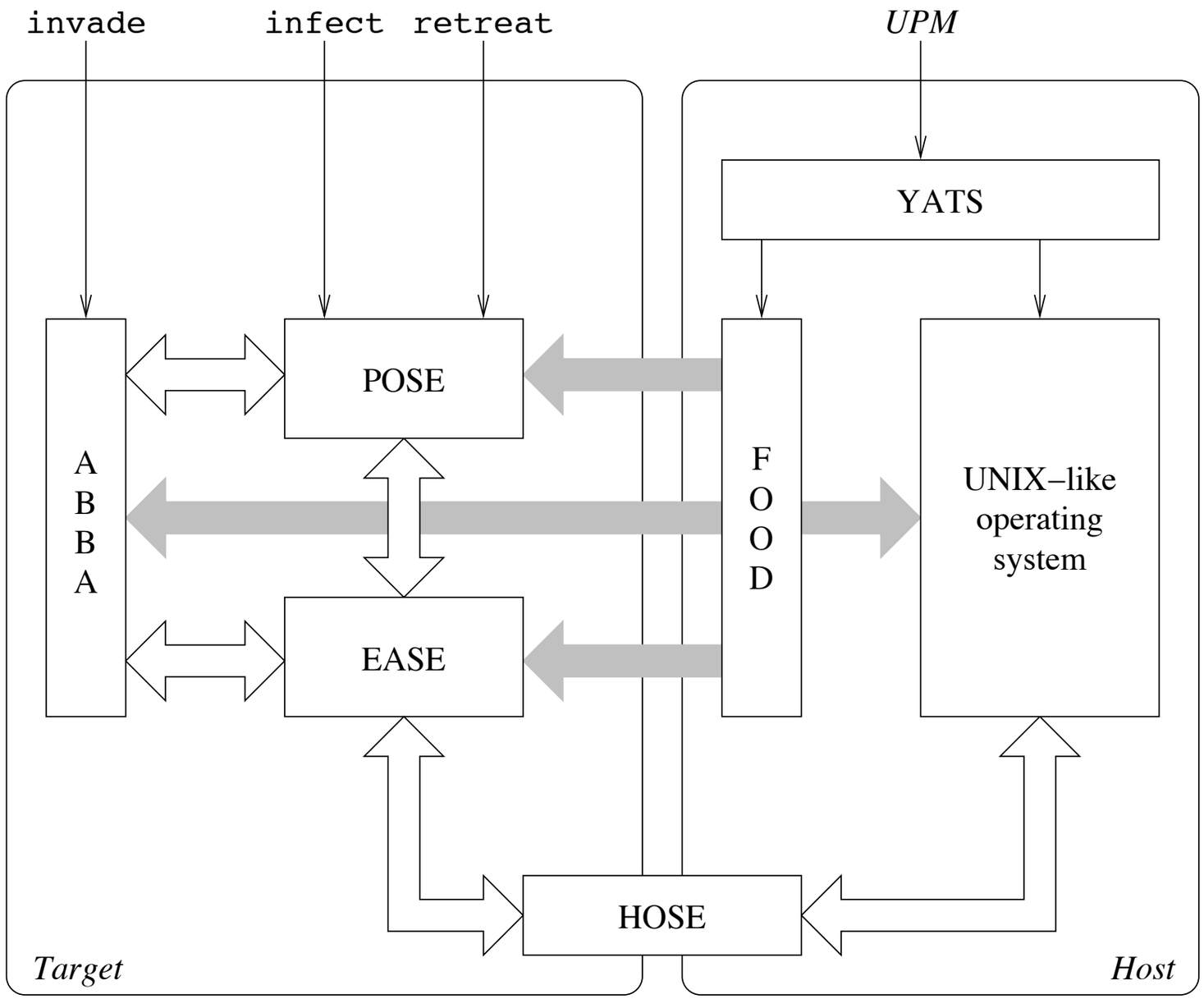
Forschungsthemen (Phase I)

1. nebenläufige Laufzeitexekutive — FAU
   - funktional verteilt, echtzeitfähig, NBS
2. vertikale Konfigurierung — FAU
   - querschneidend statisch/dynamisch, AOP
3. agentenbasierte Ressourcenvergabe — KIT
   - verteilt, Verhandlungs-/Anreizmodelle

Funktionale Hierarchie

Level	Purpose	Functional Unit	RT
6	invasive-parallel programm(s)	*i-let* instances	
5	feature-oriented operation/deployment	FOOD, YATS	2
4	agent-based bargaining of resources	ABBA	3
3	operating system extensions	POSE	2
2	host operating system	Unix-like	1
1	hardware/software edge fusion	EASE, HOSE	1
0	instruction set architecture	CPU, ASIP, TCPA	

Zusammenspiel: iRTSS



Resümee

- Begutachtung in Q1/2010

- im Falle des Scheiterns als SFB/TRR:
  - FOR
  - Paketanträge
  - Normalverfahren

Nachwort

„Die Leute hier haben überhaupt keine Ahnung von Parallel Computing — für die ist HPC nichts weiter als MapReduce.“

(Frühstück mit Karsten Schwan, SOSP’09)

Nachwort