



**RWTHAACHEN**  
**UNIVERSITY**



# MAPS compiler for MPSoC

Rainer Leupers

Software for Systems on Silicon (SSS)

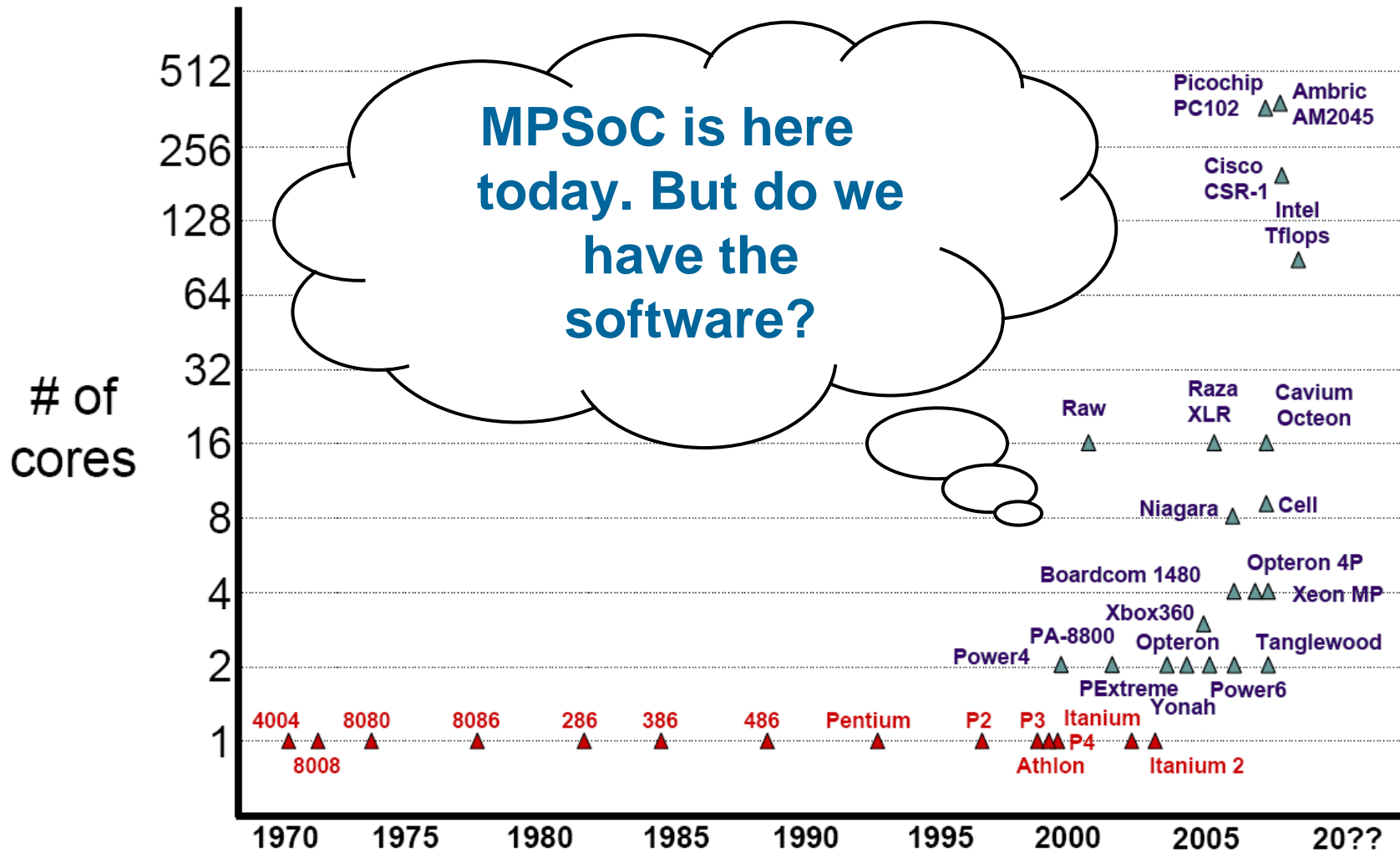
RWTH Aachen University



---

Institute for Integrated Signal Processing Systems

# MPSoC programming challenge



Rabbah, IBM, ESWeek 2007

# Problem Statement

- **New parallel languages? C is hopeless?**

"85% of all embedded developers use C/C++. Any other language is a non-starter... I don't have much hope a new parallel language will get a foothold" (EETimes, 27.9.07)

D. Kleidermacher, CTO, Green Hills Software

- **Who takes Complexity of Parallelism?**

Hidden from programmers as far as possible

- **Push-button MPSoC programming framework?**

Workbench style preferred

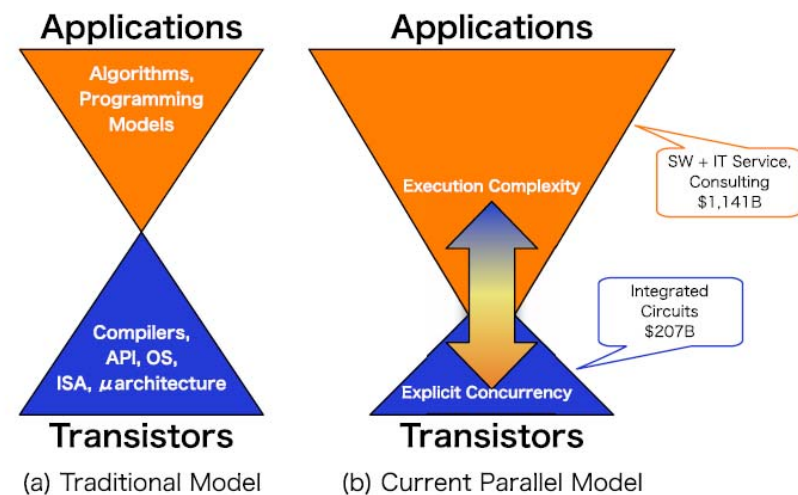


Figure 2. Cost and Complexity Exposed to Programmers

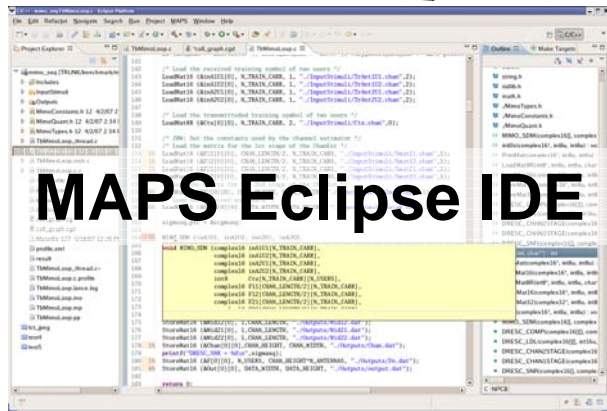
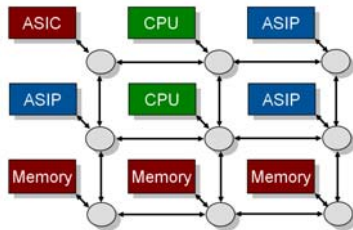
Hwu et al, DAC 2007

# MAPS Work-flow

```

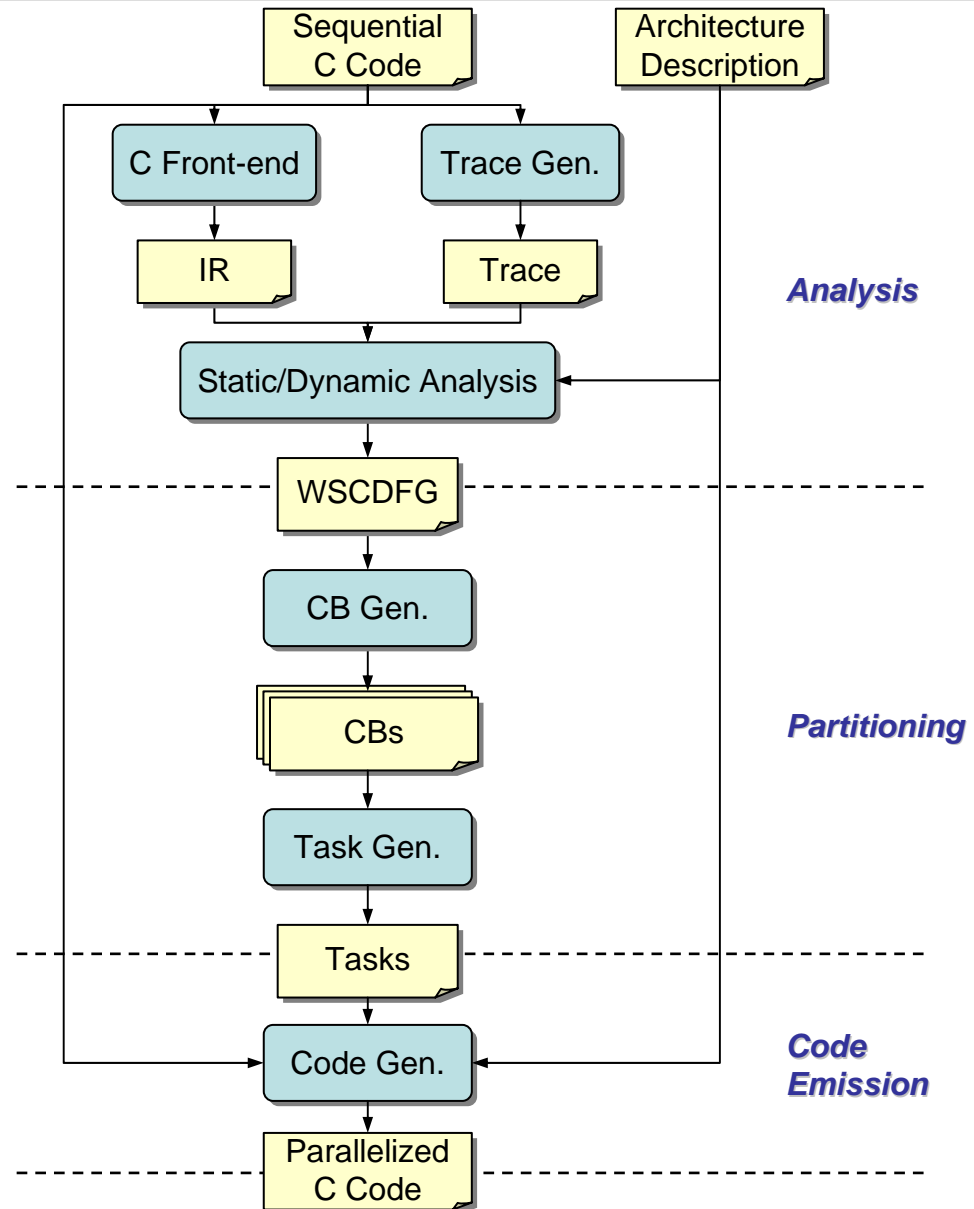
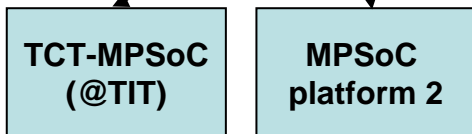
susan_corners(in,r,bp,max_no_corner_list,x_size,y_size)
uchar *in, *bp;
int *in, *bp, *max_no, *x_size, *y_size;
CORNER_LIST corner_list;
{
int n,x,y,sq,xx,yy,
1,j,*eggs,*cgy;
float divider;
uchar c,*p,*cp;

memset(r,0,x_size * y_size * sizeof(int));
cgy=(int *)malloc(x_size*y_size*sizeof(int));
cgy=(int *)malloc(x_size*y_size*sizeof(int));
    
```



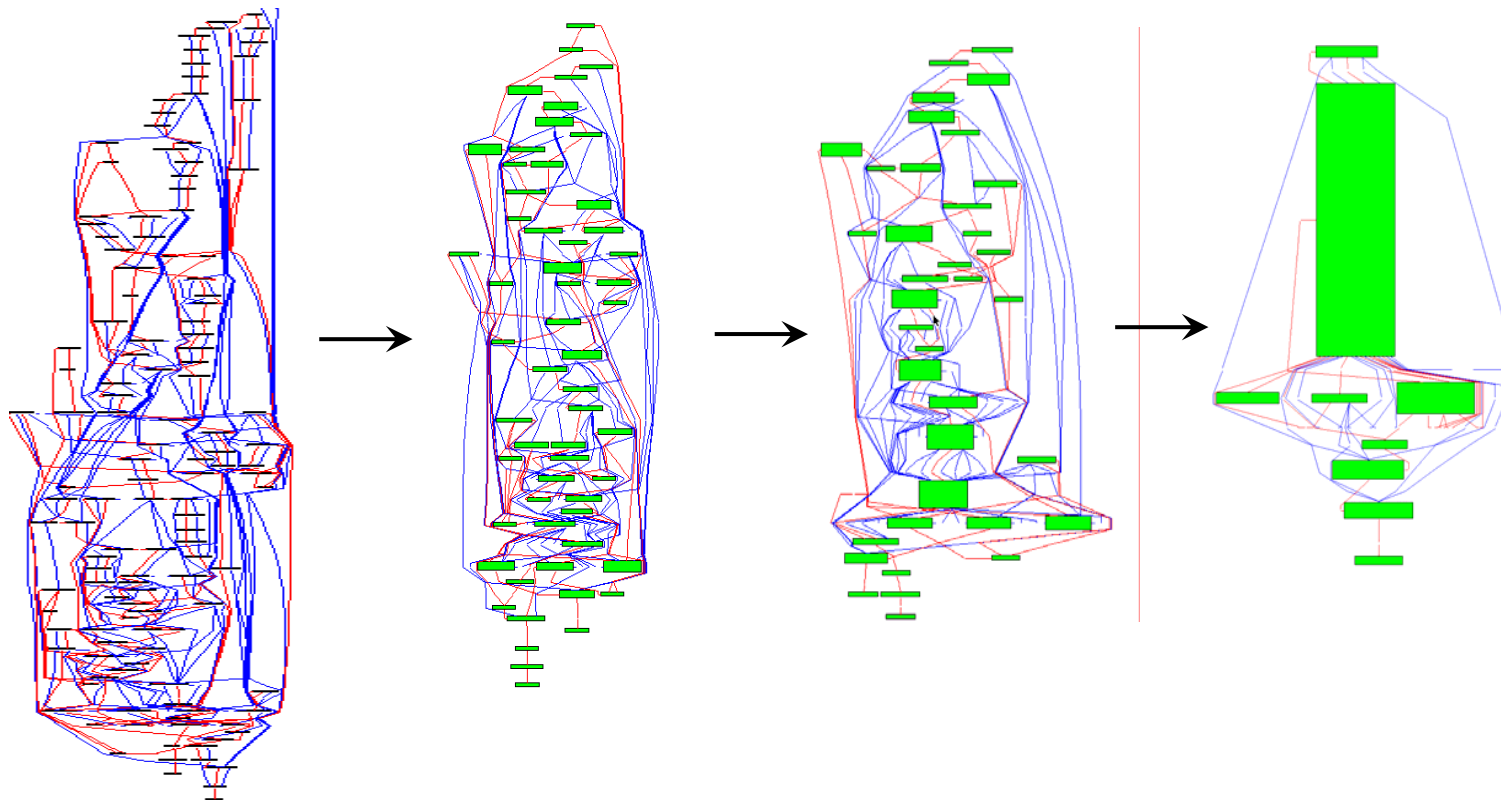
**MAPS Eclipse IDE**

↓  
Parallelized C Code



# CB-to-task clustering

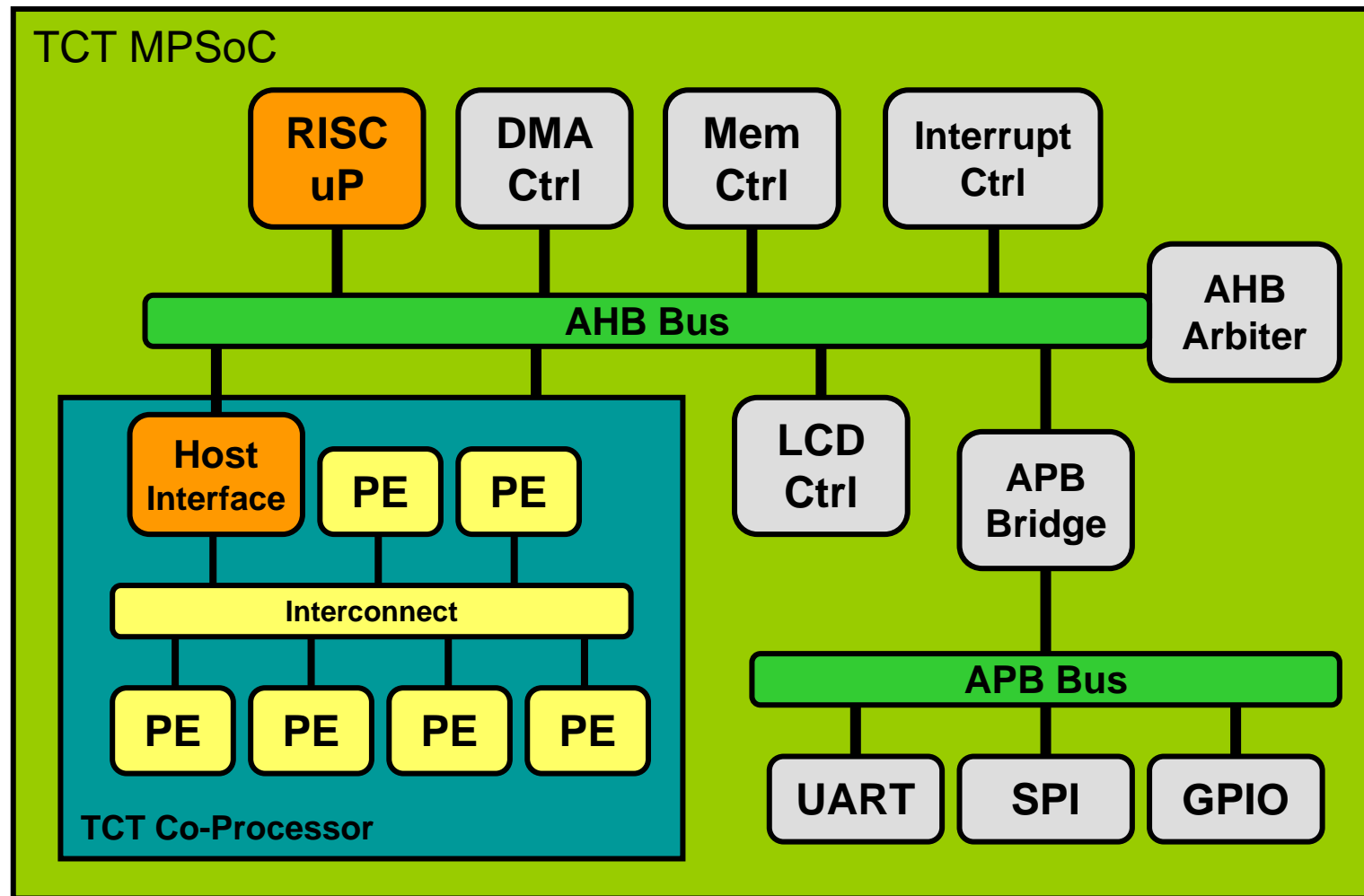
- **CAHC: Constrained Agglomerative Hierarchical Clustering** (inspired by *Agglomerative Clustering* in Data Mining)
- Enabled by advanced data flow analysis



# MAPS IDE snapshot

The screenshot displays the MAPS IDE interface. On the left, the source code for `test5.c` is shown, featuring several nested loops and arithmetic operations. The code is color-coded by block. On the right, the control flow graph (CFG) is visualized, with nodes representing basic blocks (e.g., NPCB 25, NPCB 22, NPCB 27, NPCB 35, NPCB 20, NPCB 33, NPCB 38, NPCB 29) and edges representing control flow. A yellow box highlights the C Source and IR Code for a specific block, showing the mapping between high-level code and intermediate representation.

```
C Source:  
...  
85: for (j = 0; j < 10; j++)  
...  
87: la4 = la4+3;  
88: lb4 = la4+lb4;  
...  
89: lc4 = lc4+1;  
...  
IR Code:  
113: t34 = j_4 < 10;  
99: t34 = j_4 < 10;  
112: j_4 = t36;  
111: t36 = t35 + 1;  
98: j_4 = 0;  
110: t35 = j_4;  
114: if (t34) goto LL17;  
109: LL16:  
108: lc4_24 = t39;  
107: t39 = lc4_24 + 1;  
106: lb4_17 = t38;  
105: t38 = la4_9 + lb4_17;  
104: la4_9 = t37;  
103: t37 = la4_9 + 3;  
115: LL15:  
102: LL17:  
101: if (t40) goto LL15;  
100: t40 = lt34;
```



- **Drastically simple programming model**

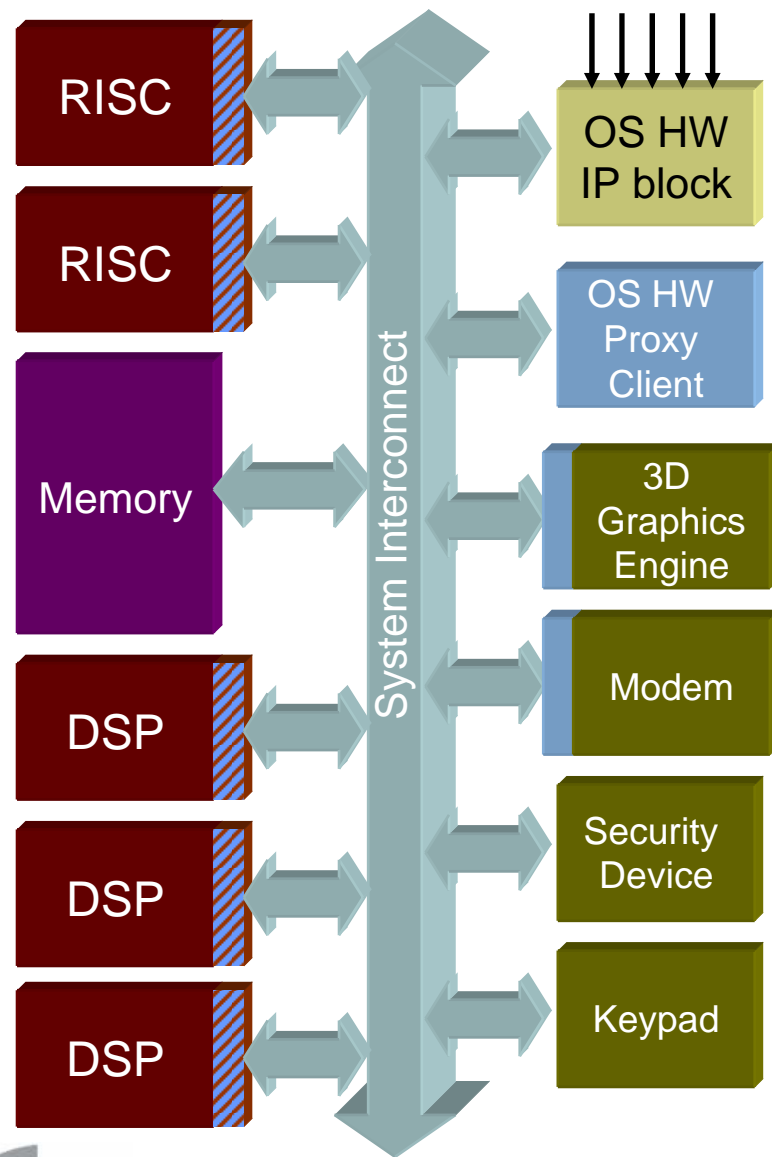
- Syntax:

**THREAD**(*name*) { *statements* }



- **Thread scope header** introduces a new *thread* labeled “*name*” in the program
- **Thread scope** can have any compound statements in C, and also other thread scopes (*thread scopes can be nested*)
- **Very simple coding style: seamless transition from sequential C codes**
- **Compiler support for automatic communication insertion**

# Towards HW supported MPSoC multiprocessing



- **Tasks need to be synchronized and need to communicate**
- **In general-purpose computing platforms: OS/RTOS supported**
- **Embedded MPSoC: OS overhead (performance/energy) is huge**
- **Research on supporting OS functionality with special HW block**
- **Must be configurable, programmable**
- **ASIP implementation ideal!**