



# Native Power Estimation for Embedded System Design-Space Exploration

```
    } else {  
        fcomplex c;  
        c.r = 1.0 / sqrt(z.r * z.r + z.i * z.i);  
        c.i = 0.0;  
        return c;  
    }  
}  
  
fcomplex RCmul(float x, fcomplex a)  
{  
    fcomplex c;  
    float w;  
    if ((z.r == 0.0) && (z.i == 0.0)) {  
        c.r = 0.0;  
        c.i = 0.0;  
    } else {  
        w = sqrt((sqrt(z.r * z.r + z.i * z.i) * x));  
        c.r = a.r * w;  
        c.i = a.i * w;  
    }  
    return c;  
}  
  
fcomplex Cinv(fcomplex z)  
{  
    fcomplex c;  
    c.r = 1.0 / (z.r * z.r + z.i * z.i);  
    c.i = -z.i * c.r;  
    return c;  
}
```

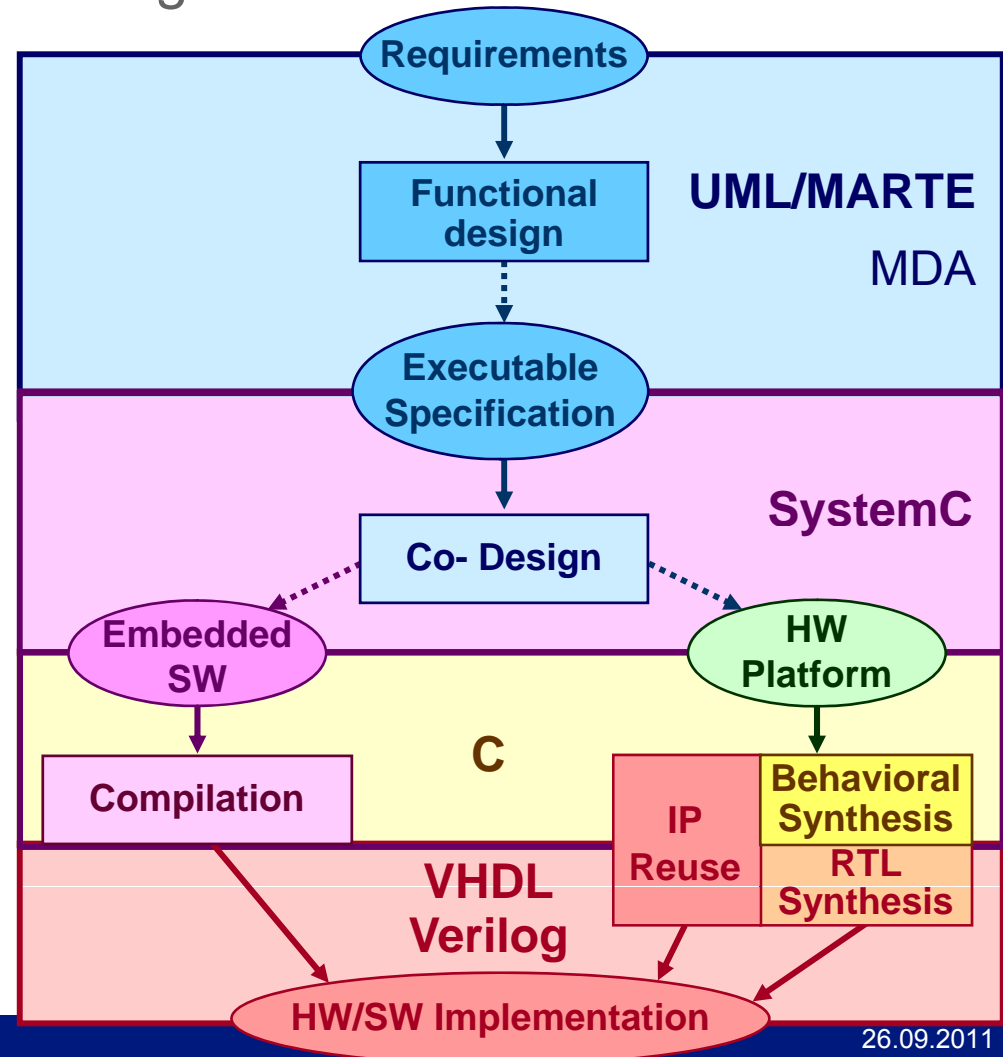
Eugenio Villar & Héctor Posadas  
Microelectronics Engineering Group  
University of Cantabria



## Context

### HW/SW Embedded Systems Design Flow

- HW/SW Simulation
- Performance Analysis
  - avoiding slow design iterations
- Design Verification
  - At the different abstraction levels





## ▶ Agenda

- Motivation: Why SW performance analysis
- Technologies: How SW performance analysis
- SCoPE: SW performance analysis for DSE
  - Improvements from Complex
- Conclusions



## ▶ Motivation

- The MPSoC
  - Multi-processing platform
    - ASIC
    - FPGA
    - Commercial multi-processing platform
  - SW-centric design methodology
    - Most of the functionality implemented as Embedded SW
    - With 'some' application-specific HW



## ► Motivation

- Computing needs Time
  - Edward A. Lee
  - Communications of the ACM, 52(5):70-79, May 2009
  
- Computing needs Energy
  - Eugenio Villar
  - Yet to be published





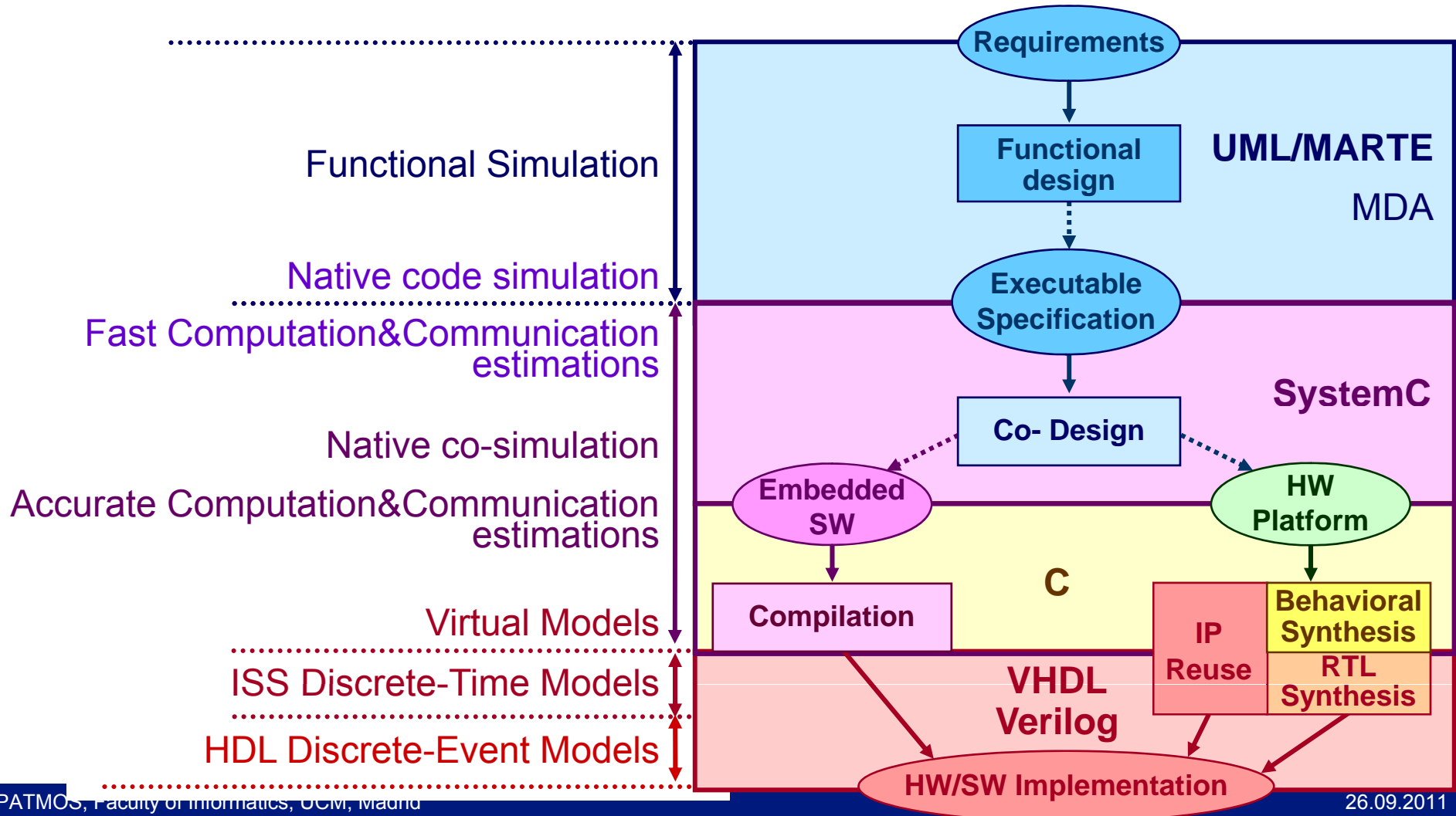
## ► Motivation

- Embedded SW performance analysis
  - SW performance analysis based on SW simulation
    - At any abstraction level
  - As an integral part of the MPSoC simulation
  - Essential for MPSoC verification
    - At any abstraction level
  - Essential for DSE
    - During architectural design



## SW Simulation Technologies

### Embedded SW simulation

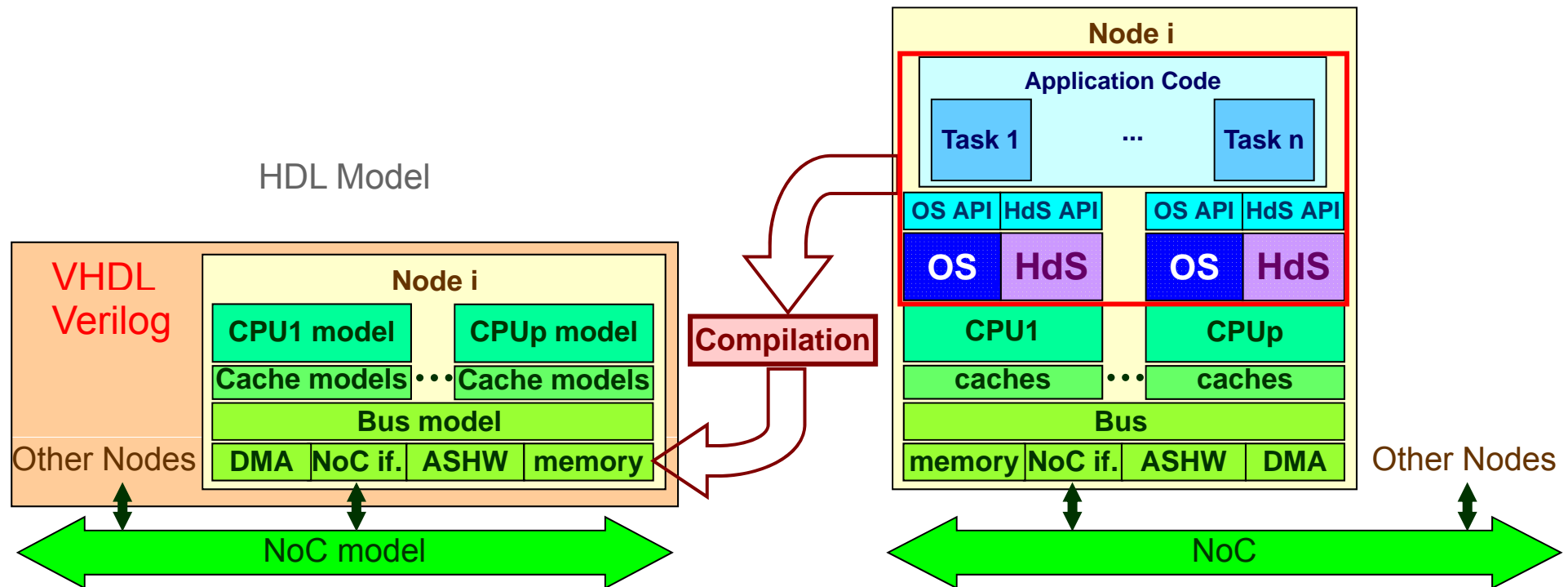




# SW Simulation Technologies

- HDL simulation

Embedded System Architecture



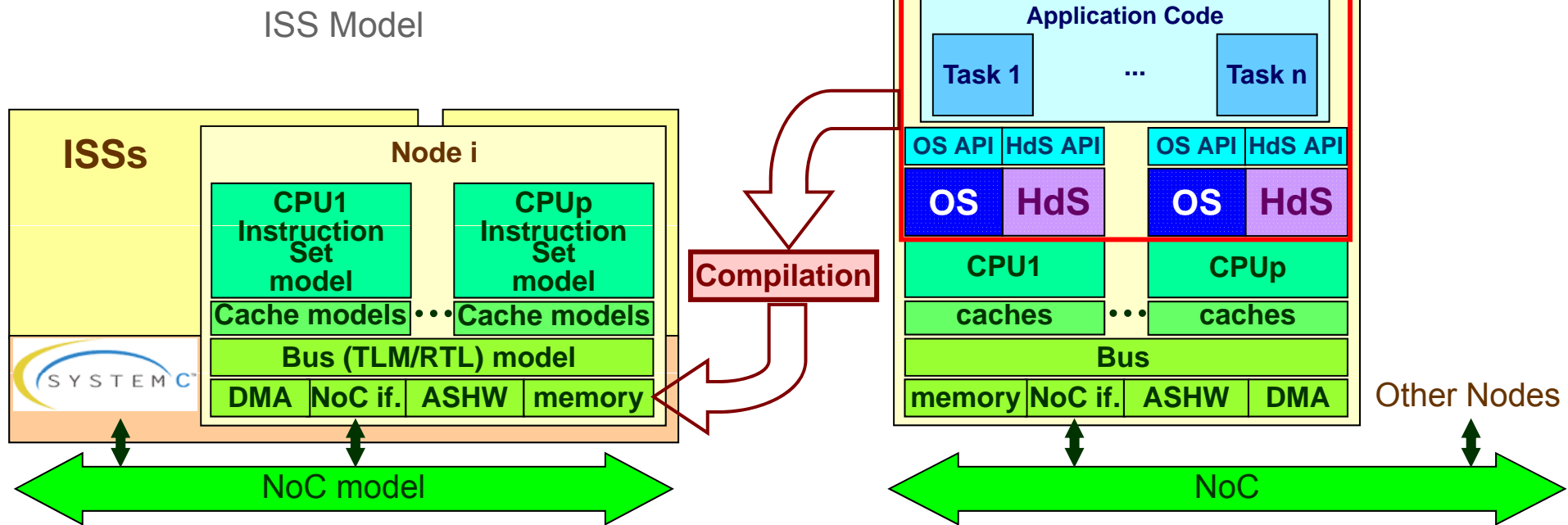




# SW Simulation Technologies

- ISS simulation

Embedded System Architecture

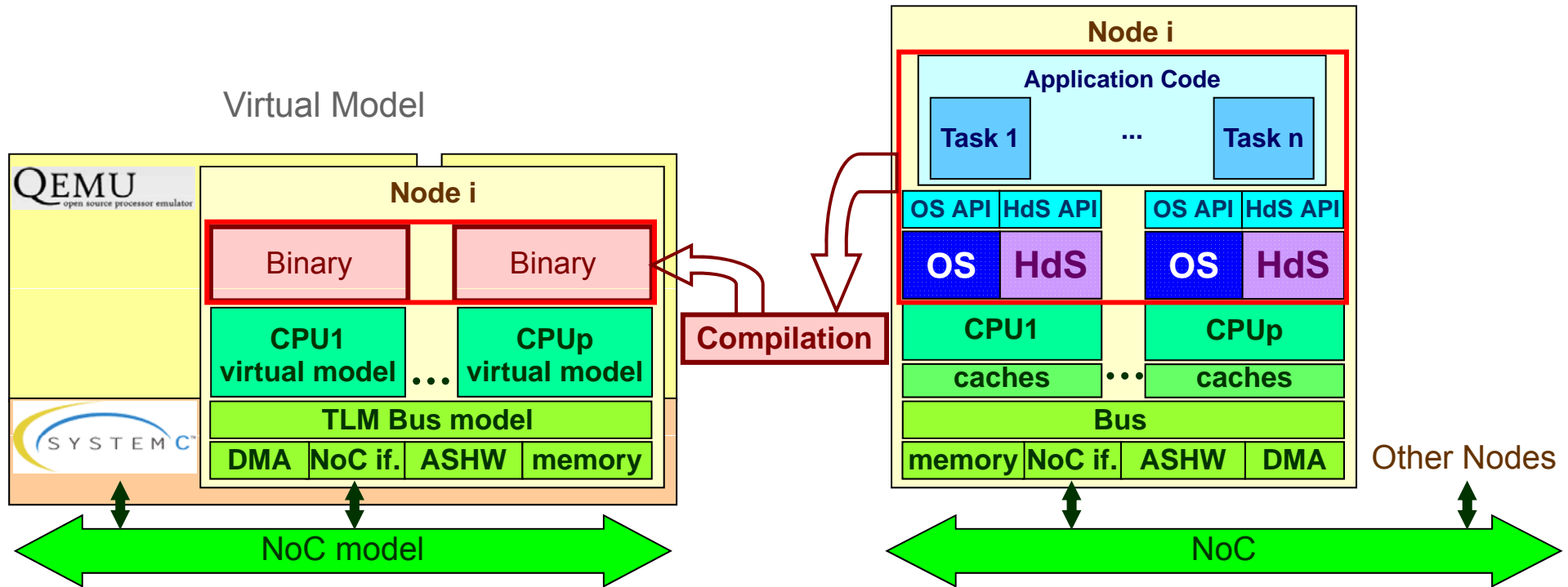




# SW Simulation Technologies

## Virtualization

Embedded System Architecture



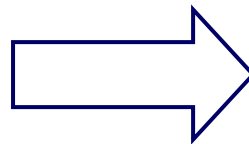


## ▶ SW Simulation Technologies

- Virtualization (QEMU)
  - Detailed model
    - High modeling cost
    - Late design steps
  - Faster than ISS

### PowerPC (200 MHz)

```
# r1 = r1 - 16  
addi r1,r1,-16
```



### Intel Core i5 (2.40 GHz)

```
# movl_T0_r1  
# ebx = env->regs[1]  
mov 0x4(%ebp),%ebx  
  
# addl_T0_im -16 # ebx = ebx - 16  
add $0xffffffff0,%ebx # movl_r1_T0  
  
# env->regs[1] = ebx  
mov %ebx,0x4(%ebp)
```



## ▶ SW Simulation Technologies

- Virtualization (QEMU)
  - Functional emulation
  - Rough timed simulation
    - i.e. 1 cycle per instruction
  - Large effort needed for more accurate modeling
    - Execution times
    - Power consumption
    - Caches
    - ...
  - Requires a specific Virtual Model for each processor
- Commercial tools
  - OVP, FastModels, Synopsys (CoWare), etc.



## ▶ SW Simulation Technologies

- Native simulation
  - Embedded code directly executed by the host
  - Good accuracy by back-annotation
  - Fast execution time

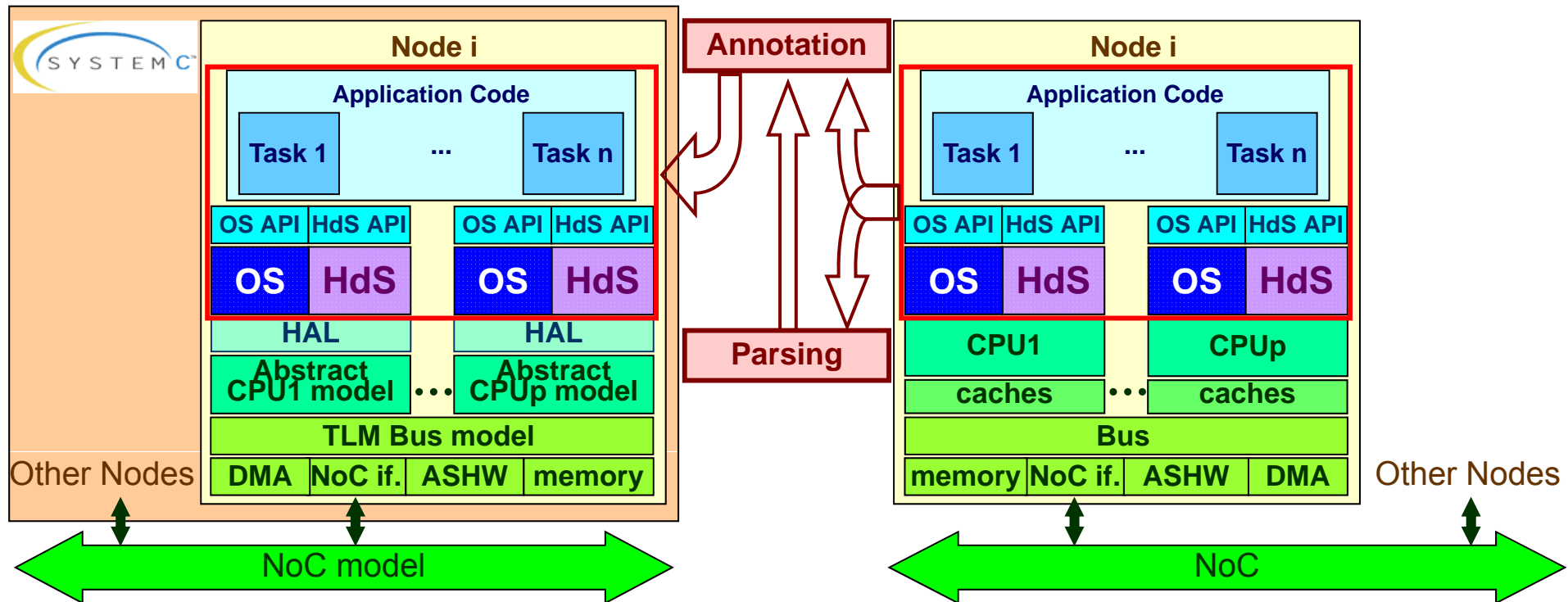


# SW Simulation Technologies

- Native simulation based on HAL API

Virtual Model

Embedded System Architecture



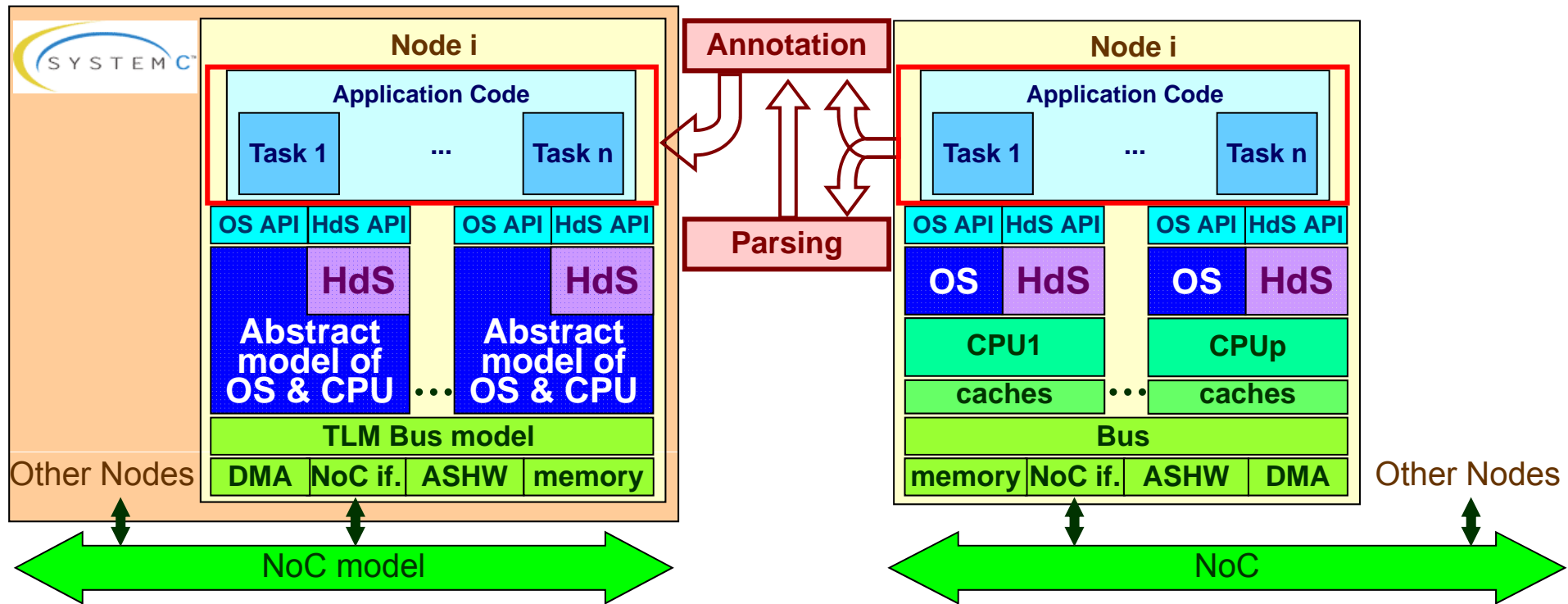


# SW Simulation Technologies

- Native simulation based on OS API

Virtual Model

Embedded System Architecture





## SW Simulation Technologies

### Basic code annotation in native simulation

...	
Overflow = 0;	
s = 1L;	
<b>for</b> (i = 0; i < L_subfr; i++) {	→ Sim_Time += 20;
Carry = 0;	
s = L_macNs(s, xn[i], y1[i]);	→ Sim_Time += 25;
<b>if</b> (Overflow != 0) {	→ Sim_Time += 15;
<b>break</b> ; }	
<b>if</b> (Overflow == 0) {	
exp_xy = norm l(s);	→ Sim_Time += 10;
<b>if</b> (exp_xy <= 0)	
xy = round(L_shr (s, -exp_xy));	→ Sim_Time += 10;
<b>else</b>	
xy = round(L_shl (s, exp_xy)); }	→ Sim_Time += 10;
mq_send(queue1, &xy, p, t);	→ wait included
...	

#### Global variable

int Sim\_Time = 0;

→ Sim\_Time += 20;

→ Sim\_Time += 25;

→ Sim\_Time += 15;

→ Sim\_Time += 10;

→ Sim\_Time += 10;

→ Sim\_Time += 10;

→ wait included



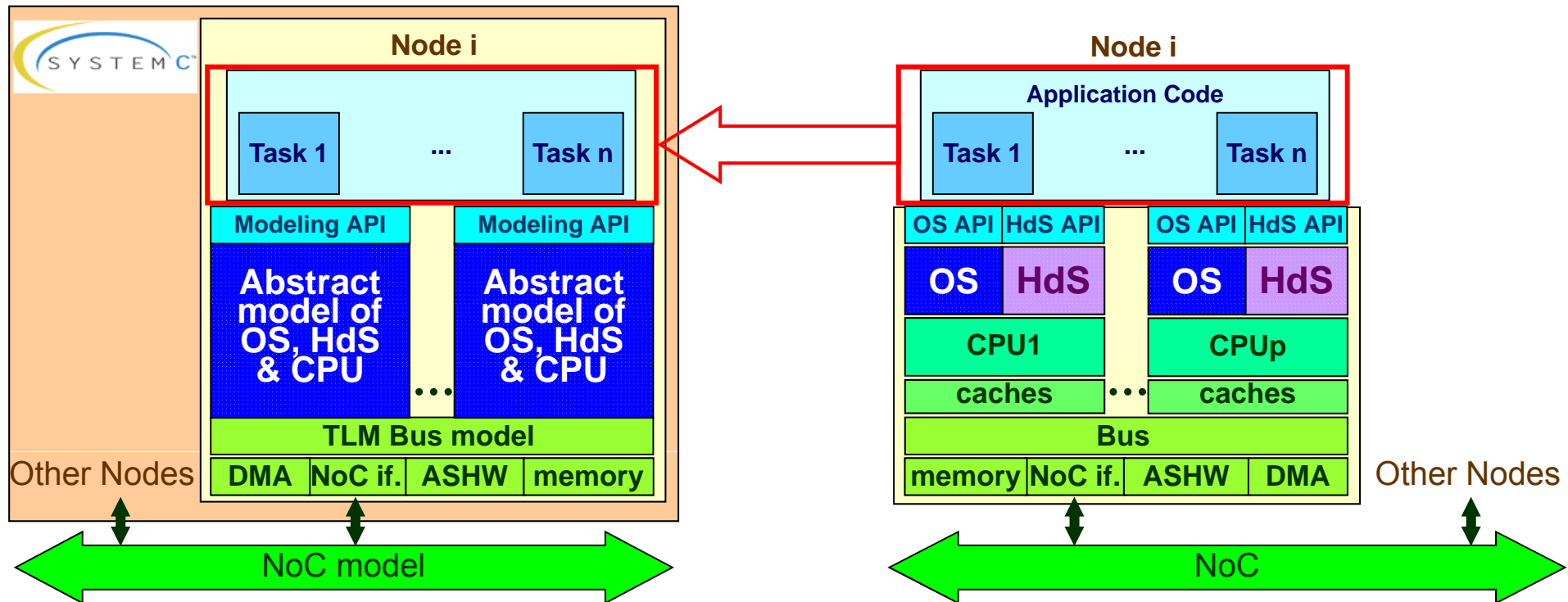


## SW Simulation Technologies

- Functional simulation based on code

Virtual Model

Embedded System Architecture

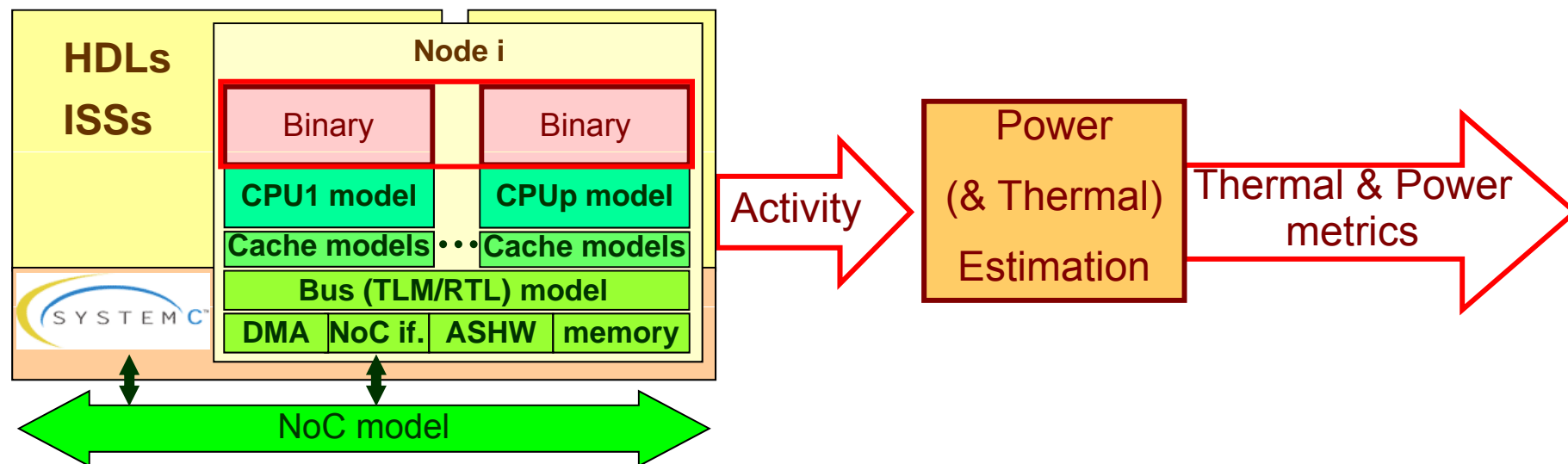




## ▶ SW Simulation Technologies

- Power estimation based on traces
  - Accurate but slow

ISS Model



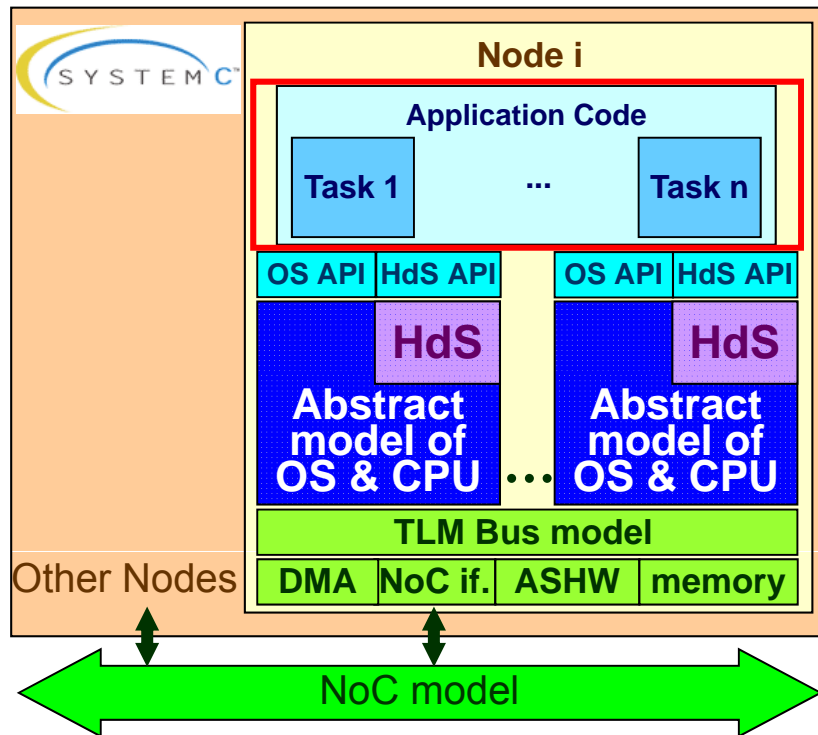


## SW Simulation Technologies

- Power estimation based on back-annotation

Virtual Model

- Same technique as with execution times



**Global variable**  
`int Sim_Energy = 0;`

- Best ratio accuracy/speed



## ▶ SW Simulation Technologies

### ■ Performance/Error comparison

		Technology	Time Estimation	Time & Power Estimation
Functional	Performance		5,000	N.A.
	Error		N.A.	N.A.
Native	Performance		1,000	500
	Error		1.3	1.4
Virtualization	Performance		200	T.B.M.
	Error		1.5	T.B.M.
ISS (cycle-accurate)	Performance		10	1
	Error		1.1 (DT)	1.1
HDL	Performance		1	0.1
	Error		1 (DE)	1

➤ Rough approximate figures



## ▶ SCoPE: SW Performance Estimation for DSE

- Key features
  - Abstract OS modeling
  - Instruction cache modeling
  - Data cache modeling
  - System power estimation
- Novel features
  - Physical memory accesses
  - Separate memory spaces
  - Configurability for Design-Space Exploration
  - Dynamic Voltage-Frequency Scaling
  - Thermal modeling
  - System composition from IP-XACT components
  - Win32 API





## ► SCoPE: SW Performance Estimation for DSE

- Instruction cache modeling
  - Similar to time modeling

```

...
Overflow = 0;
s = 1L;
for (i = 0; i < L_subfr; i++) {
    Carry = 0;
    s = L_macNs(s, xn[i], y1[i]);
    if (Overflow != 0) {
        break;
    }
    if (Overflow == 0) {
        exp_xy = norm_l(s);
        if (exp_xy <= 0)
            xy = round(L_shr (s, -exp_xy));
        else
            xy = round(L_shl (s, exp_xy));
    }
}
...

```

```

struct icache_line { char num_set; char hit; }
...
static icache_line line_124 = {0};
static icache_line line_125 = {0};
static icache_line line_126 = {0};
...

```

→ If (line\_124.hit == 0) insert\_line(&line\_124);

→ If (line\_125.hit == 0) insert\_line(&line\_125);

→ If (line\_126.hit == 0) insert\_line(&line\_126);



## ►23 SCoPE: SW Performance Estimation for DSE

- Data cache modeling<sup>Ⓟ</sup>
  - Use modified native addresses to get data variable addresses
- Global array with all memory line status
- L2 modeling

```

...
Overflow = 0;
s = 1L;
for (i = 0; i < L_subfr; i++) {
    Carry = 0;
    s = L_macNs(s, xn[i], y1[i]);
    if (Overflow != 0) {
        break;
    }
    if (Overflow == 0) {
        exp_xy = norm_l(s);
        if (exp_xy <= 0)
            xy = round(L_shr (s, -exp_xy));
        else
            xy = round(L_shl (s, exp_xy));
    }
}
...

```

bool cache[dcache\_size/line\_size];

→ If (cache[GET\_TAG(&Overflow)] == 0)  
     insert\_line(GET\_TAG(& Overflow));

→ If (cache[GET\_TAG(&s)] == 0)  
     insert\_line(GET\_TAG(& s));

→ If (cache[GET\_TAG(&Carry)] == 0)  
     insert\_line(GET\_TAG(& Carry));

→ If (cache[GET\_TAG(&i)] == 0)  
     insert\_line(GET\_TAG(&i));

→ If (cache[GET\_TAG(&xn[i])] == 0)  
     insert\_line(GET\_TAG(& xn[i])); ...

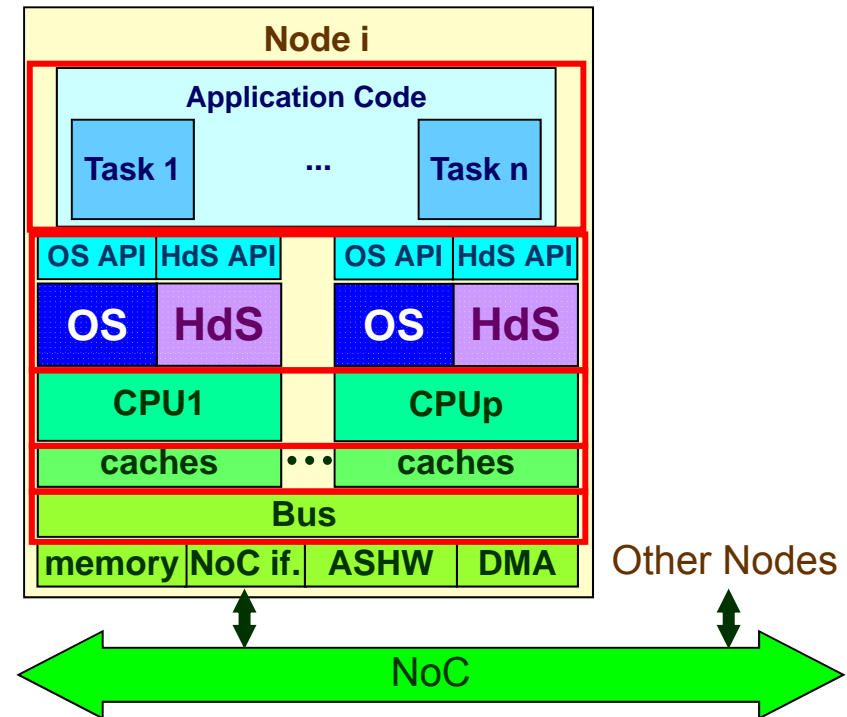
→ If (cache[GET\_TAG(&exp\_xy)] == 0)  
     insert\_line(GET\_TAG(& exp\_xy));



## ► SCoPE: SW Performance Estimation for DSE



- System power estimation
  - Application code
    - Instruction counting from binary
  - HW-dependent SW
    - Function power estimation
  - Caches
    - Counting memory accesses
    - Cache misses
  - Bus
    - Actual bandwidth
      - Cache misses
      - DMA accesses
      - HW accesses
  - HW & NoC
    - SystemC power models

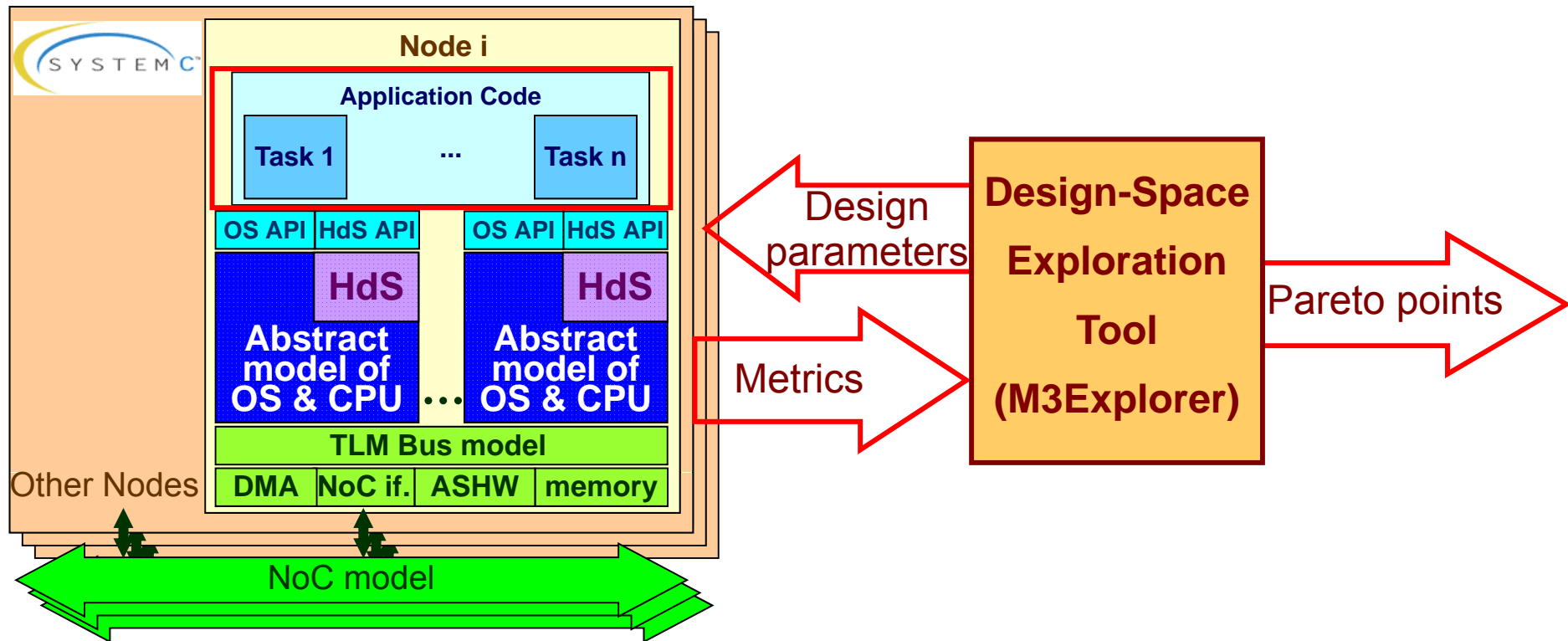






## SCoPE: SW Performance Estimation for DSE

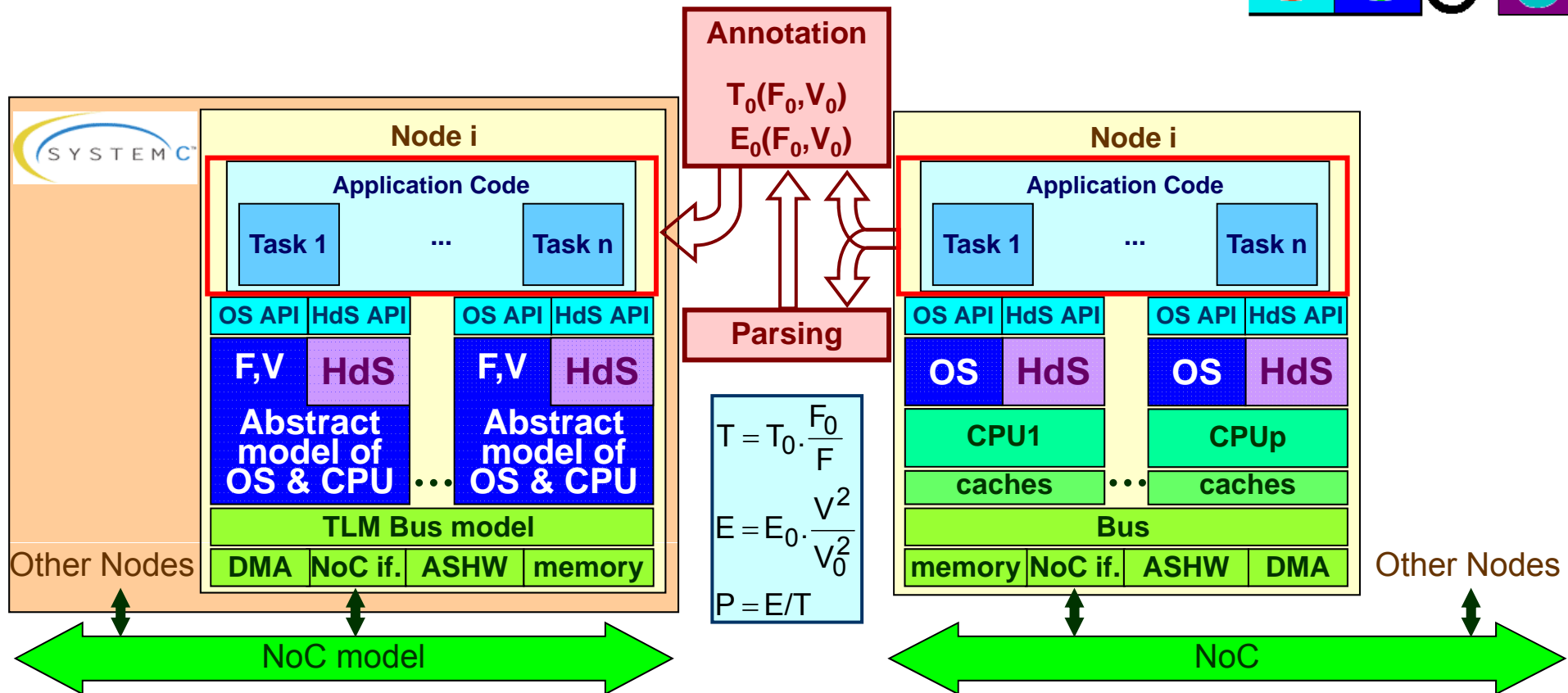
- Design-Space Exploration
  - Configurable model





# SCoPE: SW Performance Estimation for DSE

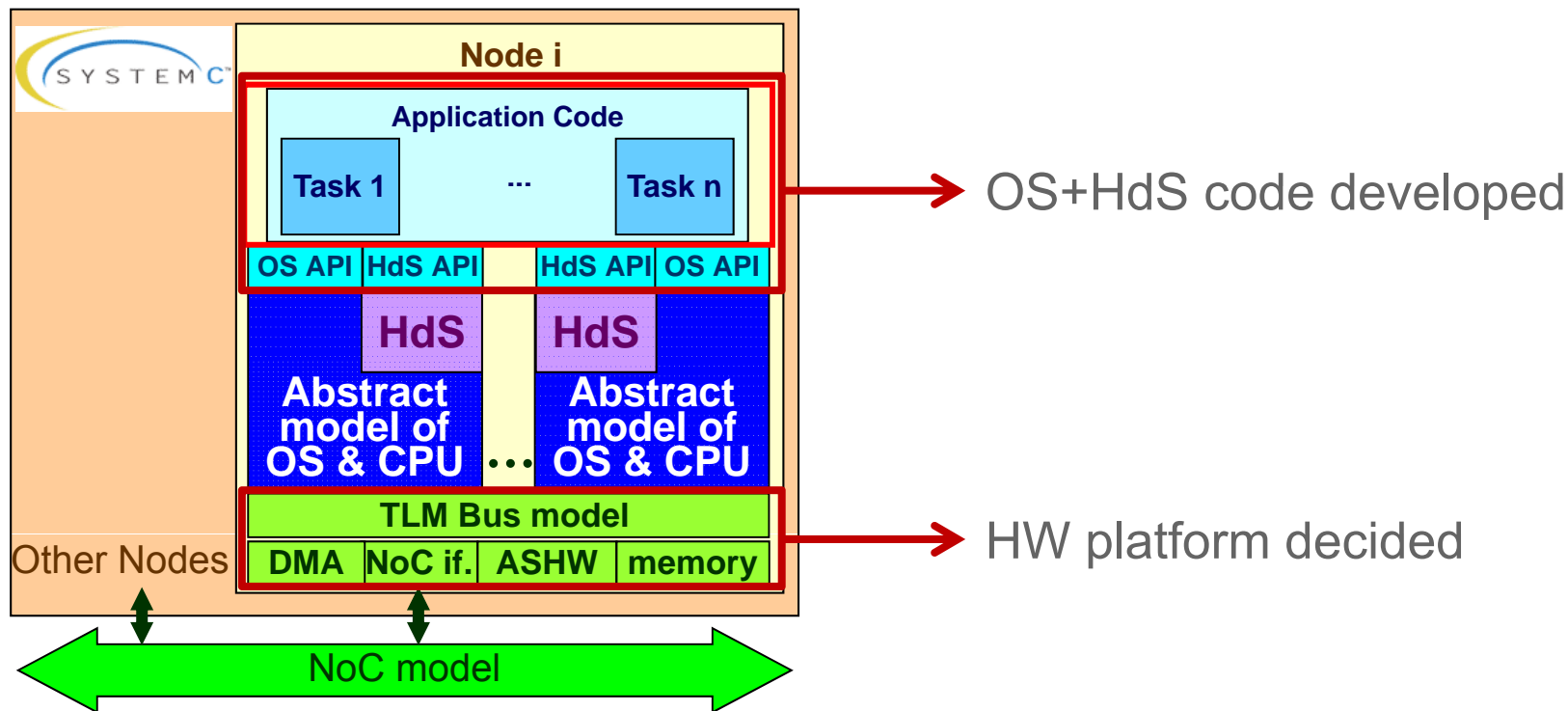
- Dynamic Voltage-Frequency Scaling





## ► SCoPE+: Improvements from Complex

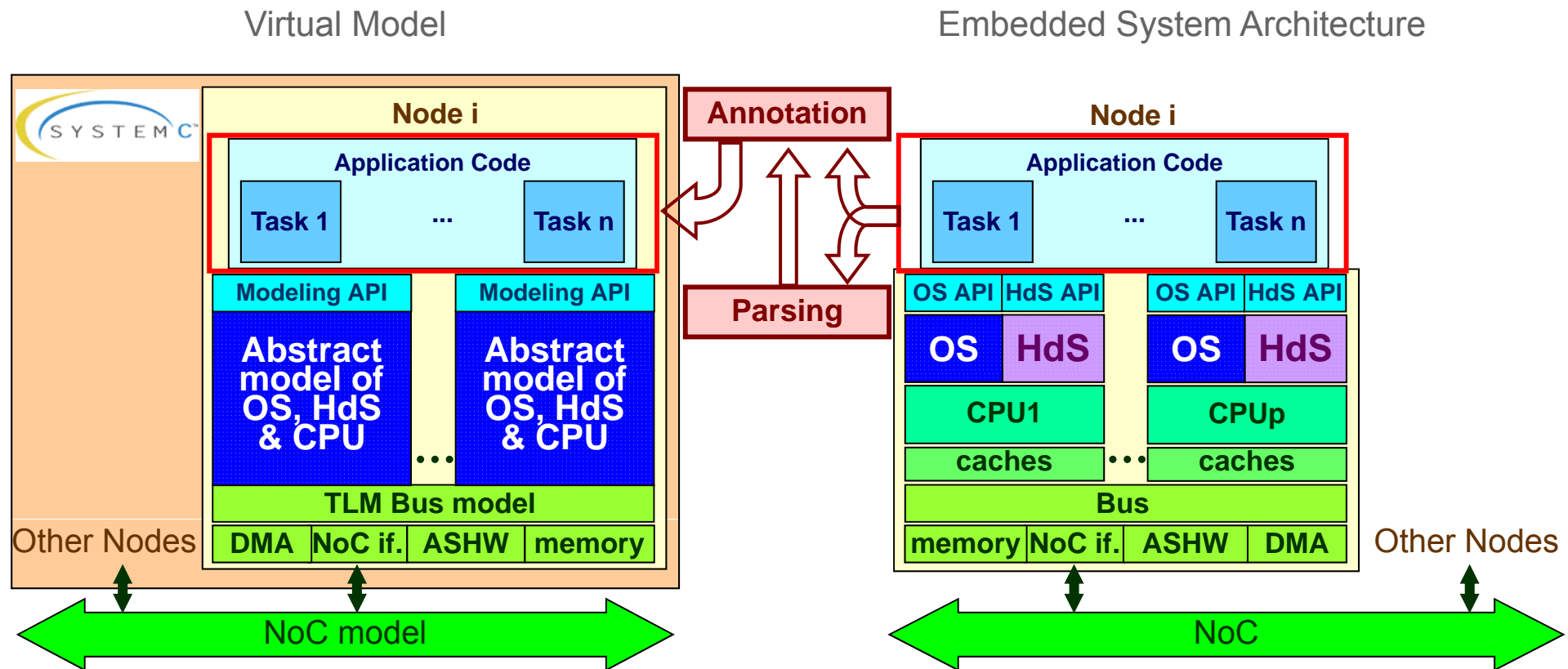
- Limitations of application code simulation
  - Valid for co-simulation after partitioning





## SW Simulation Technologies

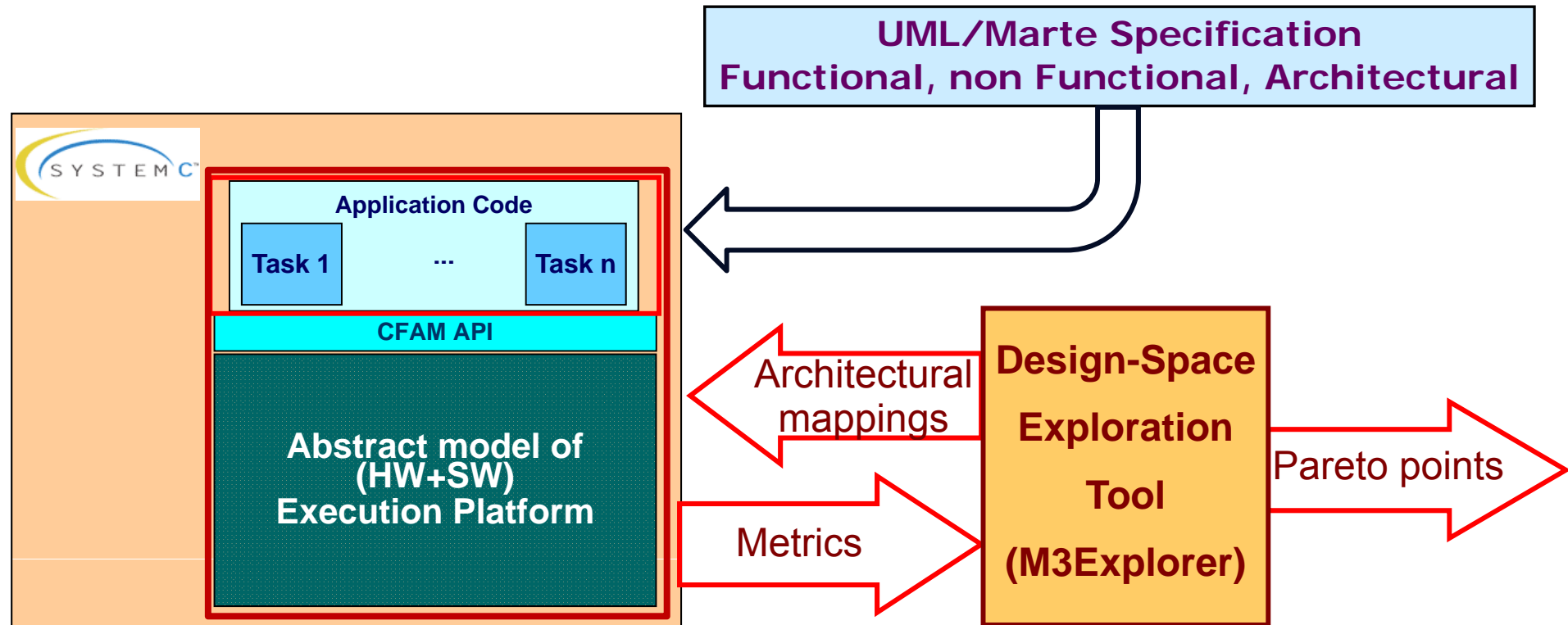
- Functional, back-annotated simulation based on code





## ► SCoPE+: Improvements from Complex

- Performance estimation before partitioning





## ▶ Conclusions

- SW simulation and performance analysis
  - Essential Design Technology
  - HW/SW Embedded Systems
  - At different design steps
    - Different modeling and simulation technologies
    - Various performance\*accuracy products
  
- SCoPE
  - SystemC Native Co-Simulation Technology
  - Specially tuned to performance analysis
    - Design-Space Exploration



---

▶ **Thank you for your attention**

---

- ▶ **Slides available at:**
  - ▶ [www.teisa.unican.es/en/publicaciones](http://www.teisa.unican.es/en/publicaciones)
  
- ▶ **Open-source SCoPE available at:**
  - ▶ [www.teisa.unican.es/scope](http://www.teisa.unican.es/scope)