# Model-based Programming Environment of Embedded Software for MPSoC

Jan. 24, 2007

Soonhoi Ha

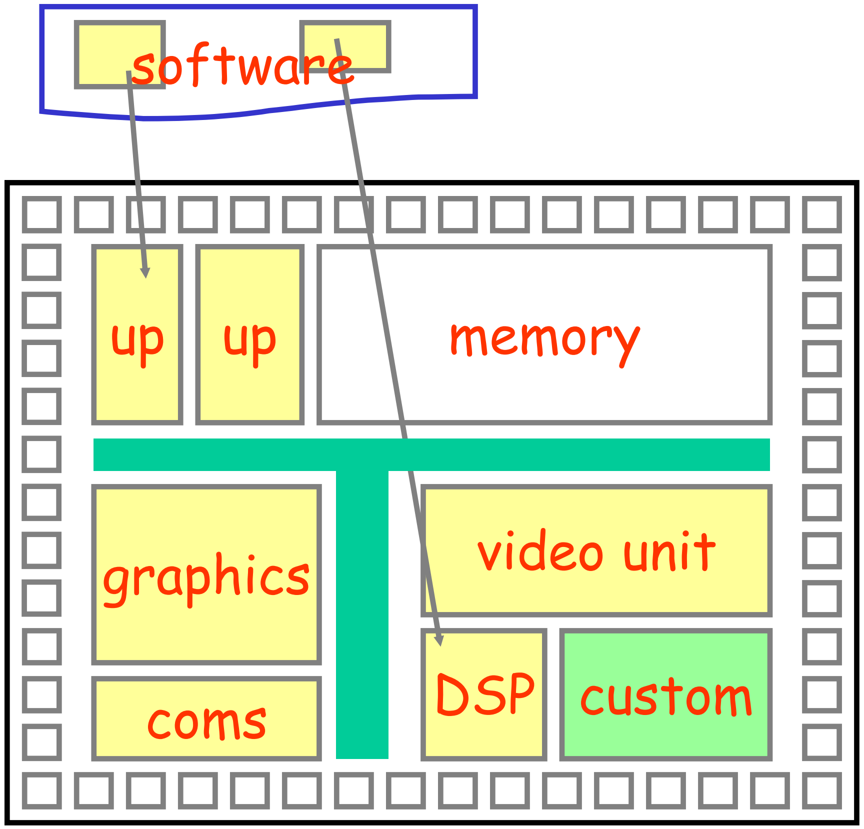
Seoul National University

## Contents

## Deep Submicron Era

- Increased chip density
  - MPSoC (Multiprocessor System on Chip), NoC (Network on Chip)
  - Increased chip density
  - Increased NRE cost
  - Increased manufacturing cost
- Platform based design
  - Platform: common (HW/SW) denominator over multiple applications
  - Pre-built and verified (HW/SW) architecture
  - SW design and system verification are major challenges

# SoC (System-on-a-Chip)



- Assembly of "prefabricated component"
  - Maximize VC(IP) reuse: over 90%
  - New economics: fast and correct design > optimum design
- Design and verification at the system level
  - interface between VCs
  - SW becomes more important

## Motivation

- Current SoC-related projects (in Korea) focus on hardware design and verification

- Software design on MPSoC
  - Embedded software with timing and resource constraints
  - Parallel programming for heterogeneous multiprocessors
  - Fast virtual prototyping

## Related Work

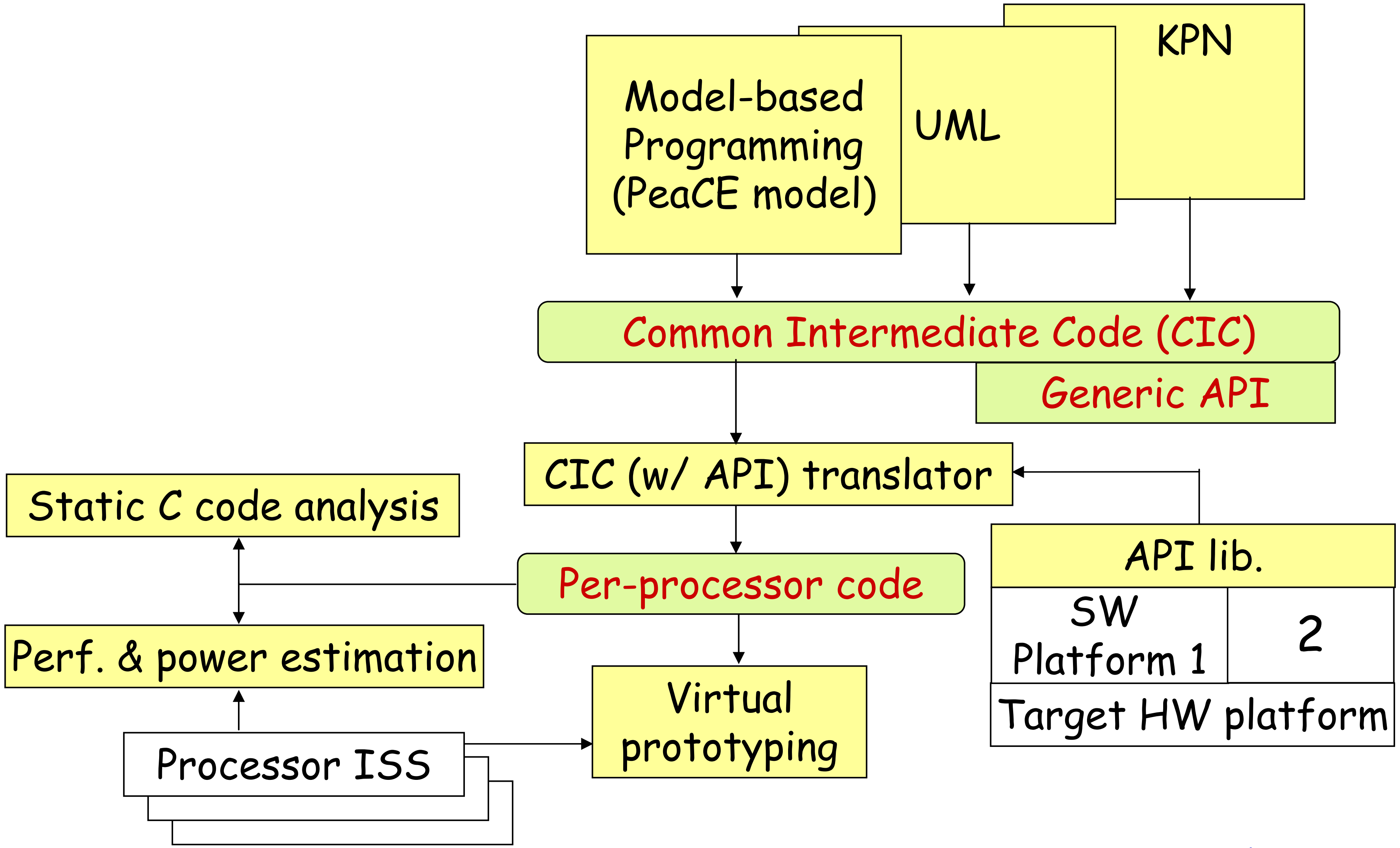
- Model-based software design
  - UML-based tool: IBM Rose RT, Telelogics TAU (iLogix Rapsody): control-oriented application, targetting single processor system
- Parallel Programming Environment
  - MPI, OpenMP: targetting general purpose processors
- Virtual prototyping system
  - Synopsys CCSS, Coware ConvergenSC, ARM MaxSim: focusing on hardware/software co-validation

## Current Practice (in Korea)

- Virtual prototyping
  - Coware ConvergenSC, ARM(Axys) MaxSim
  - Manual software and hardware partitioning
  - TLM (transaction level modeling) simulation
    - Simulation speed becomes important as the number of processors increases. (4 processors – under 100 Kcycles/sec)
  - Software debugging capability is limited
- S/W programming
  - Mainly manual design
  - Start considering UML based tool: (ex) Telelogic TAU
  - Limited capability: documentation, code structure.

## Contents

# HOPES Proposal

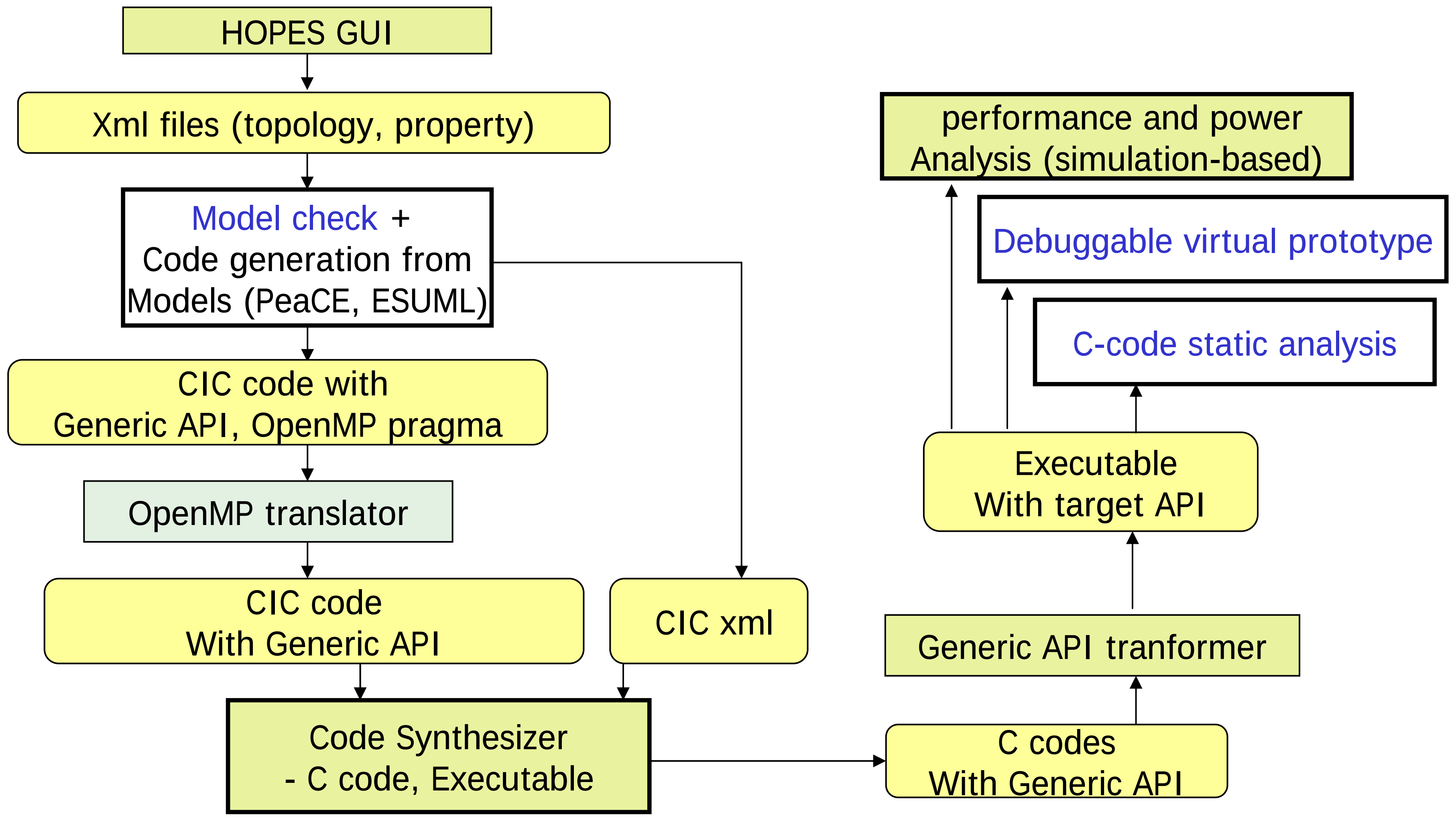


## Key Techniques

- SW design techniques
  - Model-based specification
  - Partitioning and automatic code generation from specification
  - Parallel programming (task parallelism, data parallelism)
  - Generic APIs (independent of OS and target architecture)
- SW verification techniques: 3-phase verification
  - At specification level: static analysis of program specification and functional simulation
  - After code generation: static analysis of C code
  - At simulation time: debugging on virtual prototyping environment

# Design Flow

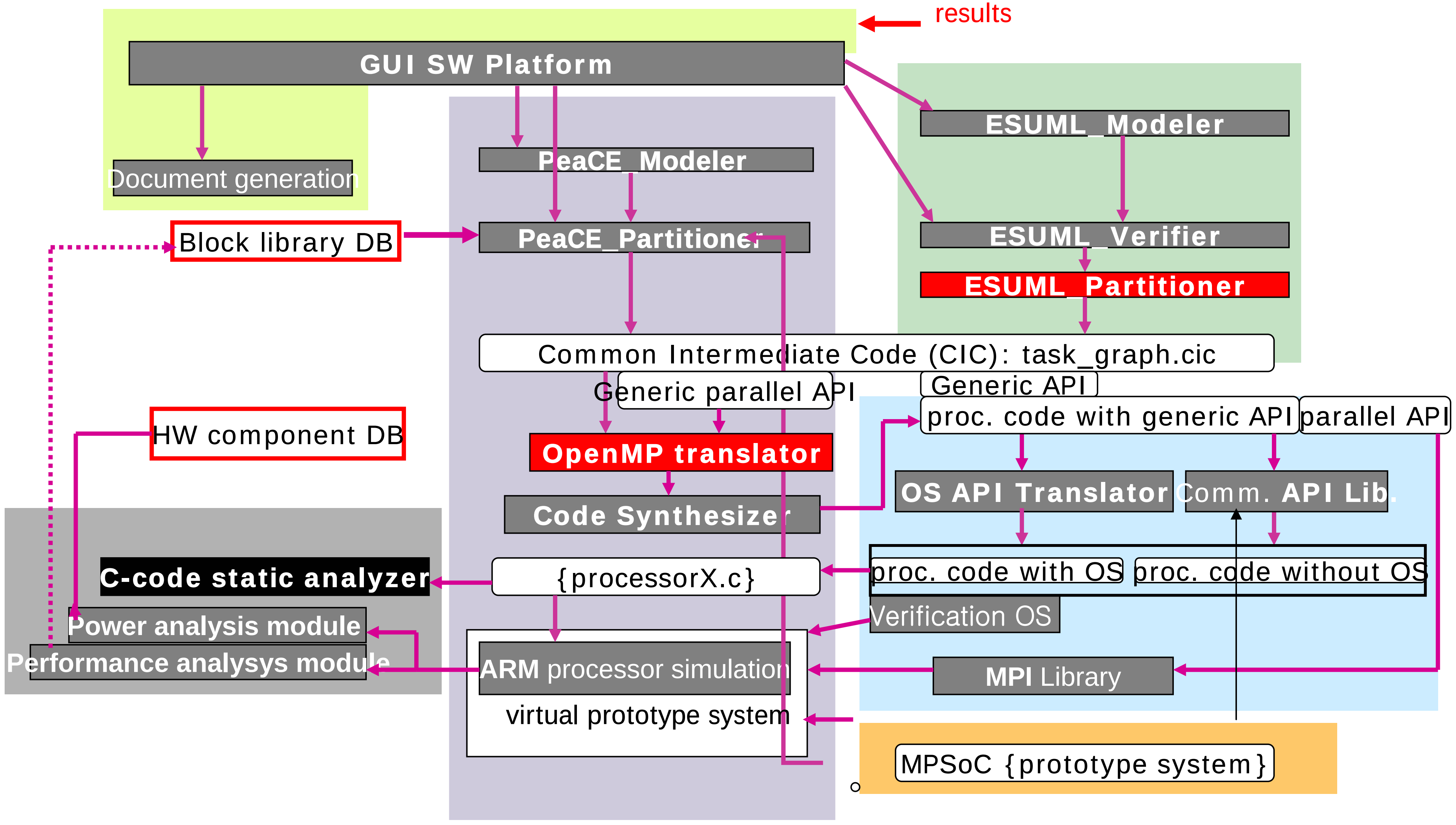
HOPES


HOPES GUI
Xml files (topology, property)
Model check + Code generation from Models (PeaCE, ESUML)
CIC code with Generic API, OpenMP pragma
OpenMP translator
CIC code With Generic API
CIC xml
Code Synthesizer - C code, Executable
C codes With Generic API
Generic API tranformer
Executable With target API
C-code static analysis
Debuggable virtual prototype
performance and power Analysis (simulation-based)

# Proposed design flow

- Model-based programming
  - PeaCE model (dataflow + FSM + task model)
  - ESUML (embedded system UML) model
- CIC (Common Intermediate Code)
  - OpenMP pragma + generic API
- Static Analysis
  - Buffer overrun, memory leak, null dereference, stack size
- Virtual prototyping
  - Performance and power estimation
  - with source-level debugging capability

## Software Module Interface



## Contents

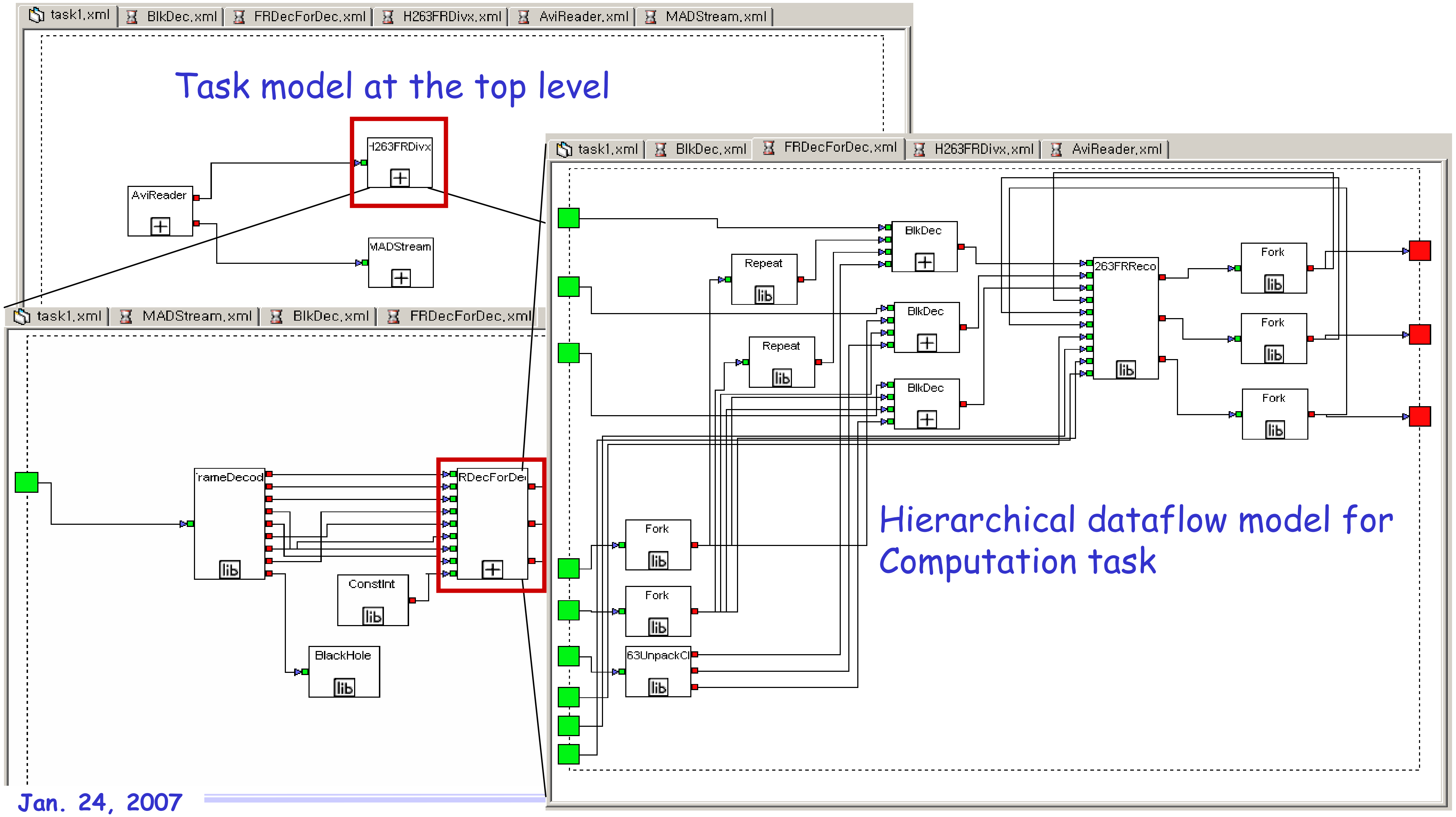
## PeaCE Model

- Top model: Task model
- Computation task model: Dataflow model
  - Extended SDF model: SPDF (Synchronous Piggybacked Data Flow) and FRDF (Fractional Rate Data Flow)
- Control task model: FSM model
  - Hierarchical and concurrent FSM model: fFSM (flexible Finite State Machine)

## PeaCE Project

- PeaCE
  - **P**tolemy **e**xtension **a**s a **C**odesign **E**nvironment based on Ptolemy classic
  - open-source research platform
  - Officially released in DAC 2005. (version 1.0)
- PeaCE home page
  - http://peace.snu.ac.kr/research/peace

# Divx Player: Top Model



Task model at the top level

Hierarchical dataflow model for Computation task

# SPDF Model

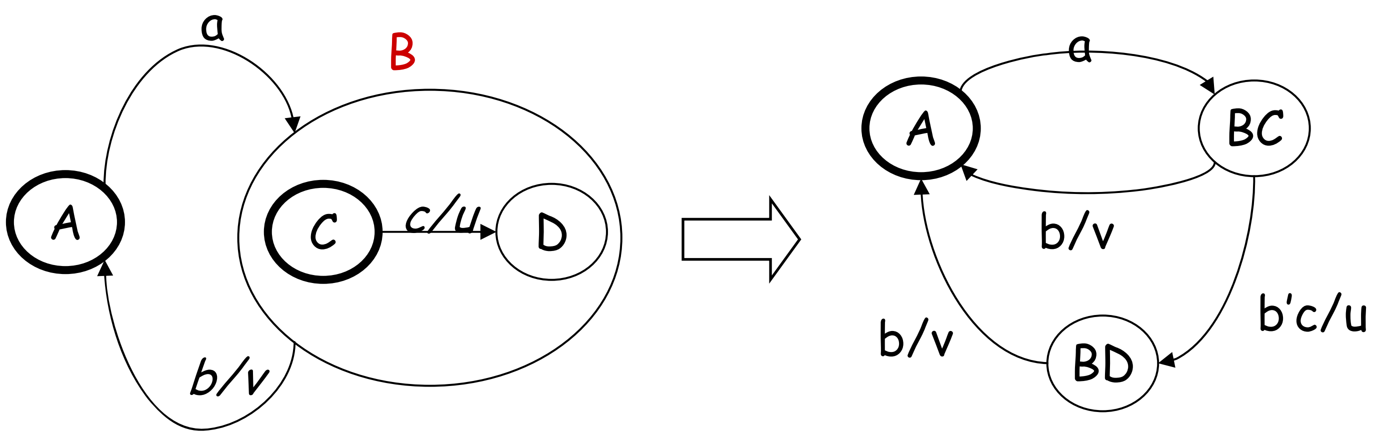
- Basic SDF Model
  – A node represents a function block (ex: FIR, DCT)
  – A block is fireable (executable when it receives all required number of input samples)
  – Statically scheduled at compile-time:
    sample rate inconsistency, deadlock condition

- Our extensions
  – Generic message type data -> code generation only
  – FRDF: fractional rate data flow
  – SPDF: synchronous piggybacked dataflow

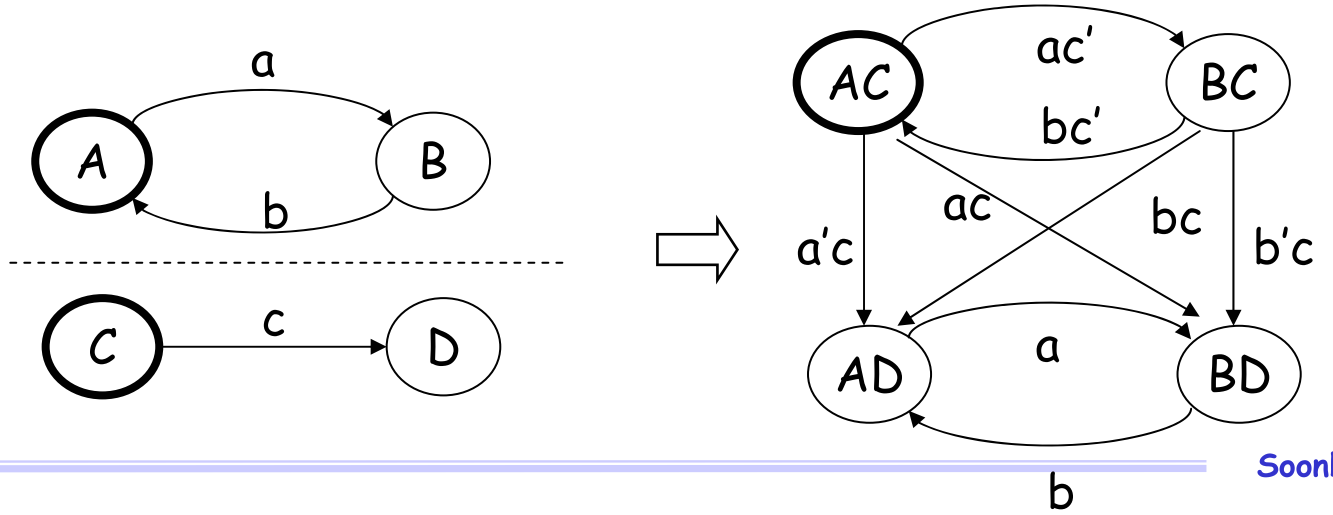
# fFSM (flexible FSM) Model

HOPES

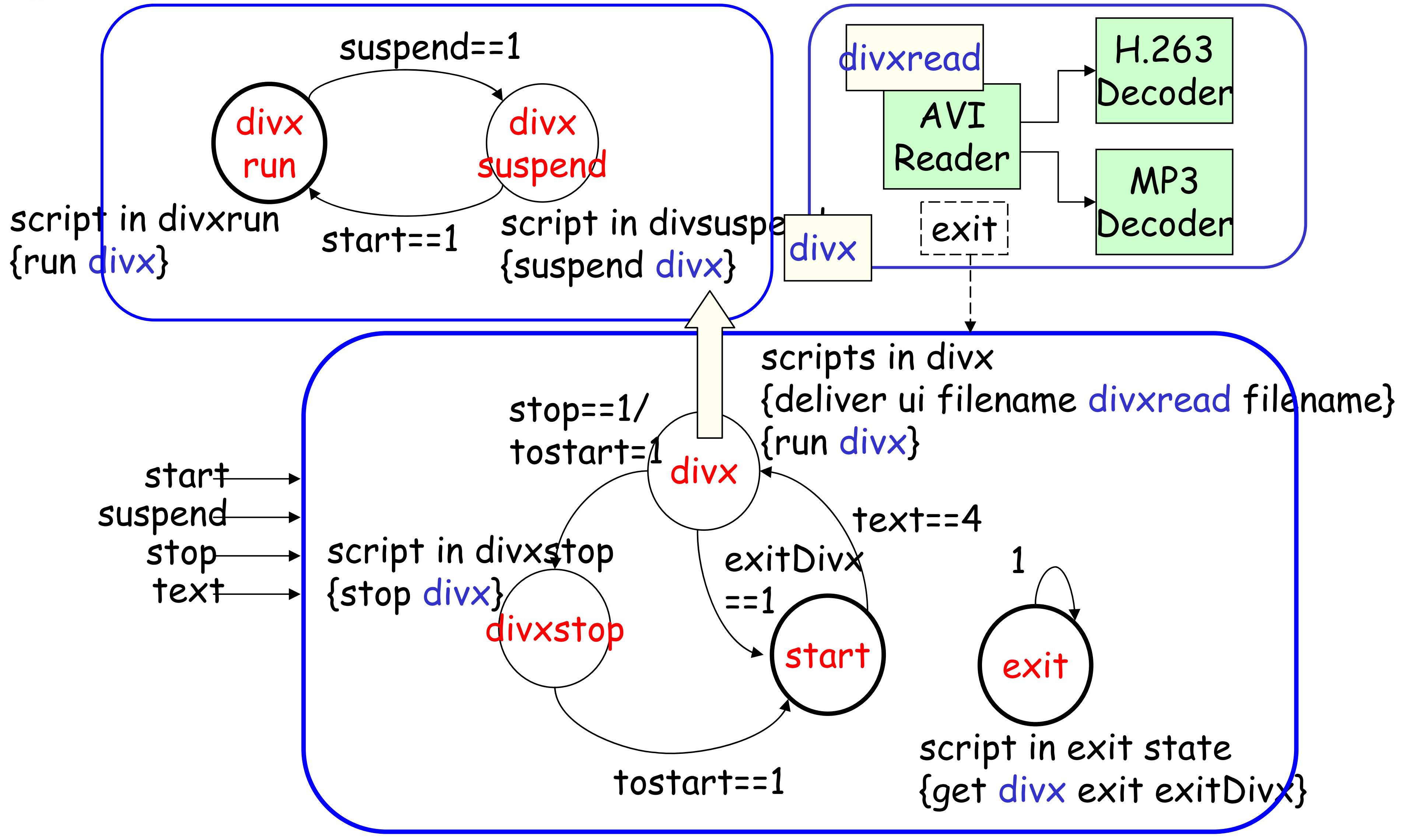
- Hierarchical FSM



- Concurrent FSM



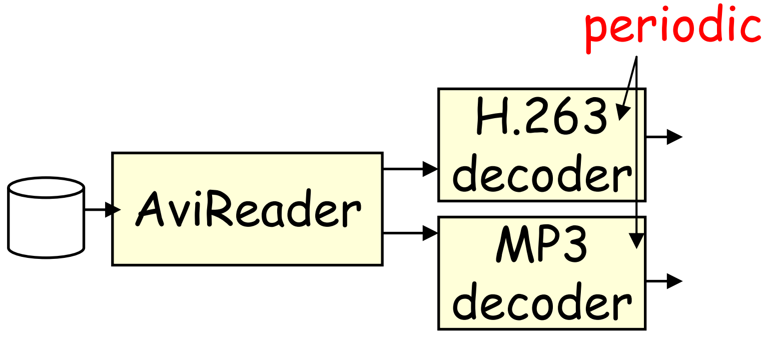
# fFSM + SPDF

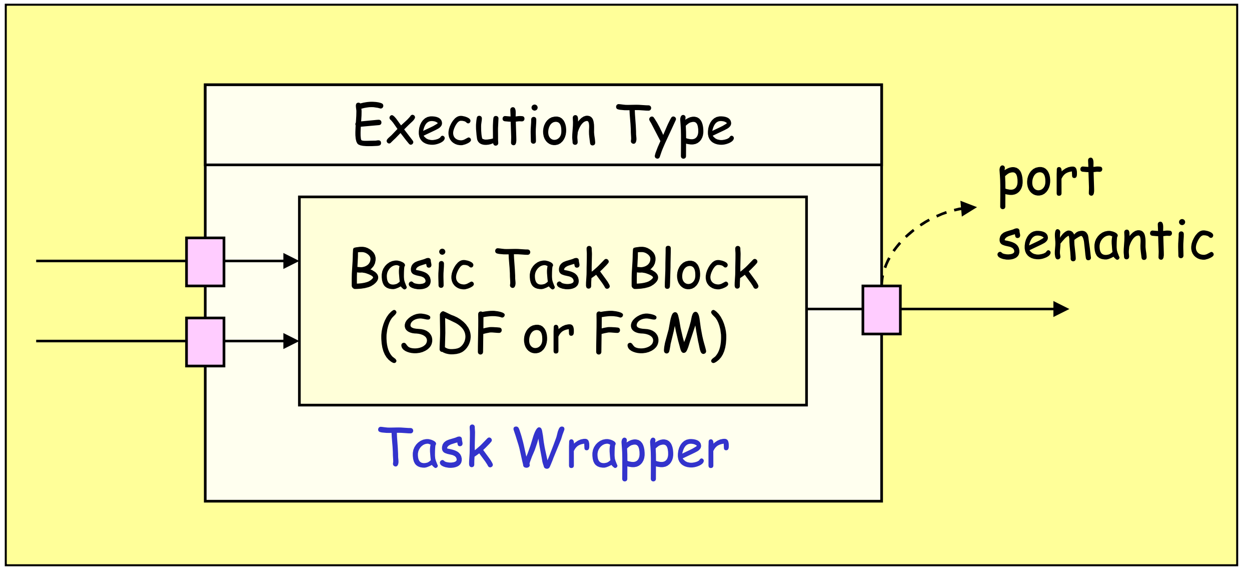


suspend==1
divx run
divx suspend
script in divxrun {run divx}
start==1
script in divsuspend {suspend divx}
divxread
AVI Reader
H.263 Decoder
MP3 Decoder
divx
exit
scripts in divx {deliver ui filename divxread filename} {run divx}
stop==1/ tostart=1
divx
start
suspend
stop
text
text==4
exitDivx ==1
1
script in divxstop {stop divx}
divxstop
start
exit
tostart==1
script in exit state {get divx exit exitDivx}

## Task Model

- Task execution semantics
  - periodic, sporadic, functional
- Port properties
  - data rate : static or dynamic
  - data size : static or variable
  - port type : queue or buffer



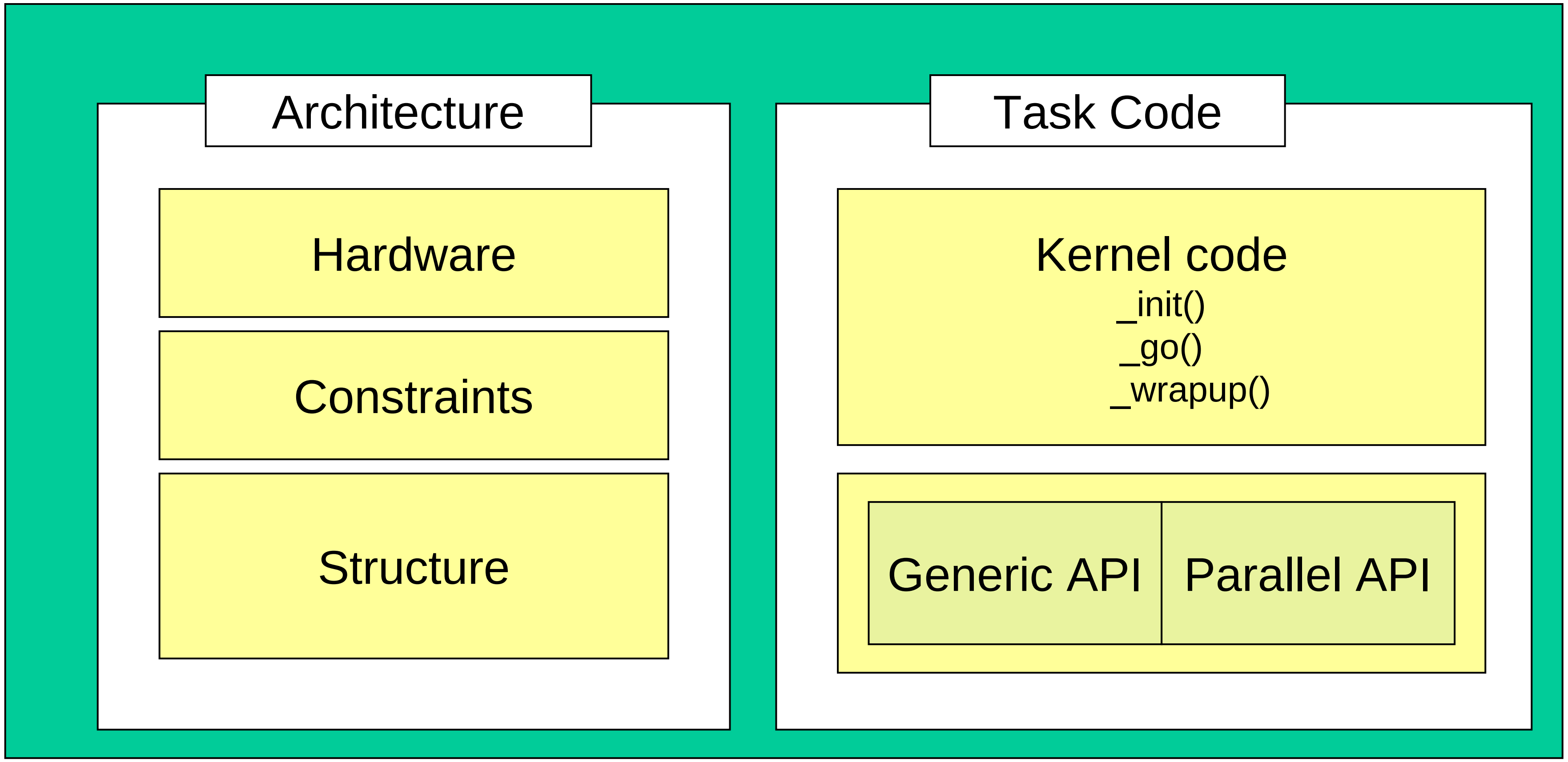


## CIC Code Generation

- Application tasks
  - Task in task-level specification model → task_name.cic
- Generic API
- Partitioning considering data parallelism
  - openMP programming

```
void h263decoder_go (void) {
       ...
        l = MQ_RECEIVE("mq0", (char *)(ld_106->rdbfr), 2048);
        ...
        # pragma omp parallel for
        for(i=0; i<99; i++) {
             //thread_main()
           ....
        }
         // display the decode frame
           dither(frame);
}
```

# CIC Format

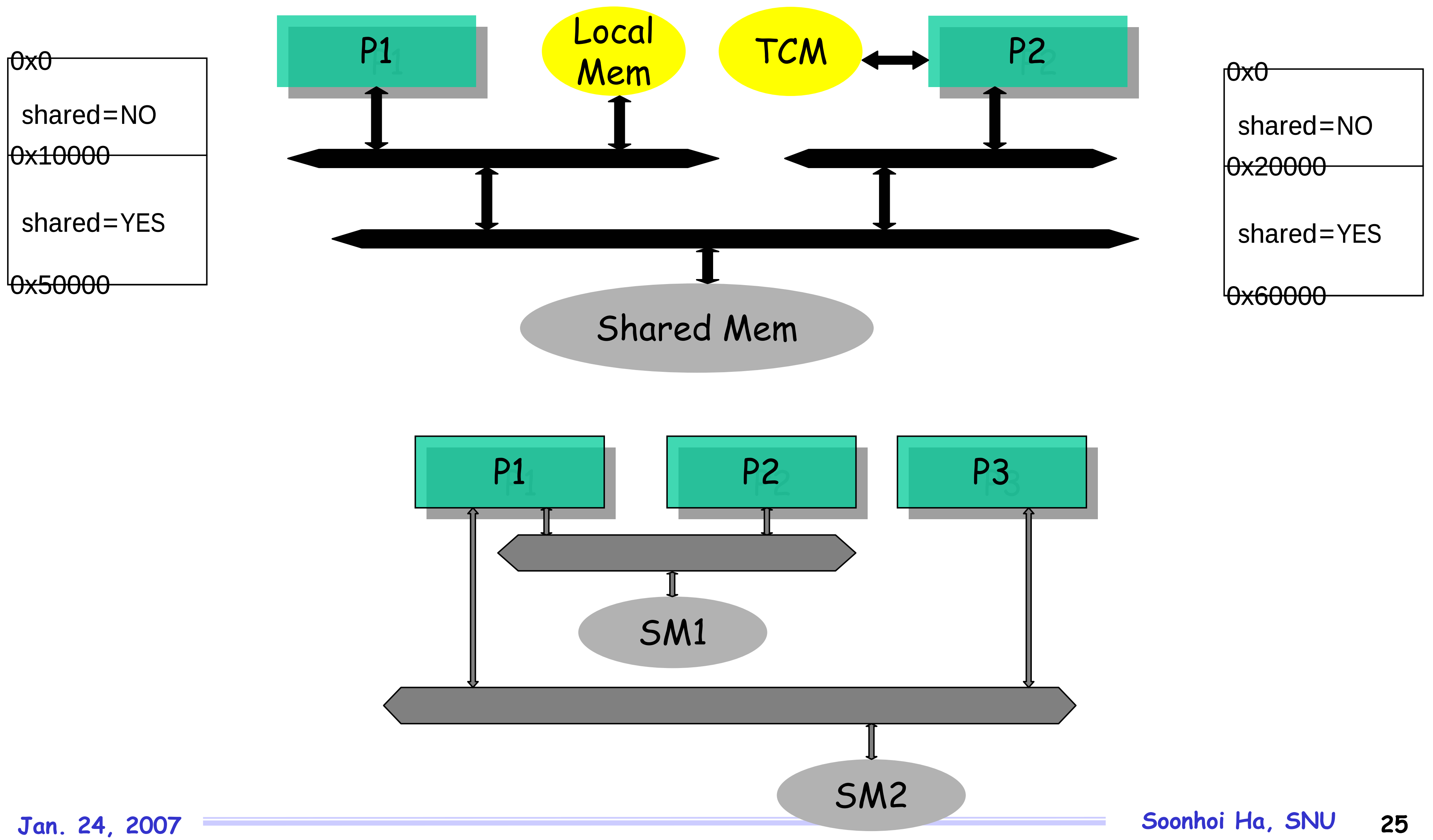

Architecture
Hardware
Constraints
Structure
Task Code
Kernel code
_init()
_go()
_wrapup()
Generic API
Parallel API

# DivX player CIC xml (Hardware)

```
<?xml version="1.0" ?>
 <CIC_XML>
  <hardware>
   <processor name="arm926ej-s0">
     <index>0</index>
     <localMem name="lmap0">
       <addr>0x0</addr>
       <size>0x10000</size>  // 64KB
     </localMem>
     <sharedMem name="shmap0">
       <addr>0x10000</size>
       <size>0x40000</size>  // 256KB
       <sharedWith>1</sharedWith>
     </sharedMem>
     <OS>
       <support>TRUE</support>
     </OS>
   </processor>
```

```
   <processor name="arm926ej-s1">
     <index>0</index>
     <localMem name="lmap1">
       <addr>0x0</addr>
       <size>0x20000</size>  // 128KB
     </localMem>
     <sharedMem name="shmap1">
       <addr>0x20000</size>
       <size>0x40000</size>  // 256KB
       <sharedWith>0</sharedWith>
     </sharedMem>
     <OS>
       <support>TRUE</support>
     </OS>
   </processor>
  </hardware>
```

# Hardware Architectures



# DivX player CIC xml (Constraint)

HOPES

```
<constraints>
 <memory>16MB</memory>
 <power>50mWatt</power>
  <mode name="default">
  <task name="AviReaderI0">
   <period>120000</period>
   <deadline>120000</deadline>
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>186</execTime>
    </subtask>
   </task>
   <task name="H263FRDivxI3">
   <period>120000</period>
   <deadline>120000</deadline>
```

```
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>5035</execTime>
    </subtask>
   </task>
   <task name="MADStreamI5">
   <period>120000</period>
   <deadline>120000</deadline>
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>13</execTime>
    </subtask>
  </task>
 </mode>
</constraints>
```

# DivX player CIC xml (Structure)

```
<structure>
  <mode name="default">
    <task name="AviReaderI0">
      <subtask name="arm926ej-s0">
      <procMap>0</procMap>
      <fileName>AviReaderI0_arm926ej_s0.cic</fileName>
      </subtask>
    </task>
    <task name="H263FRDivxI3">
      <subtask name="arm926ej-s0">
      <procMap>0</procMap>
      <fileName>H263FRDivxI3_arm926ej_s0.cic</fileName>
      </subtask>
    </task>
    <task name="MADStreamI5">
     <subtask name="arm926ej-s0">
     <procMap>0</procMap>
     <fileName>MADStreamI5_arm926ej_s0.cic</fileName>
     </subtask>
   </task>
 </mode>
```

```
  <queue>
    <name>mq0</name>
    <src>AviReaderI0</src>
    <dst>H263FRDivxI3</dst>
    <size>30000</size>
  </queue>
  <queue>
    <name>mq1</name>
    <src>AviReaderI0</src>
    <dst>MADStreamI5</dst>
    <size>30000</size>
  </queue>
</structure>
</CIC_XML>
```

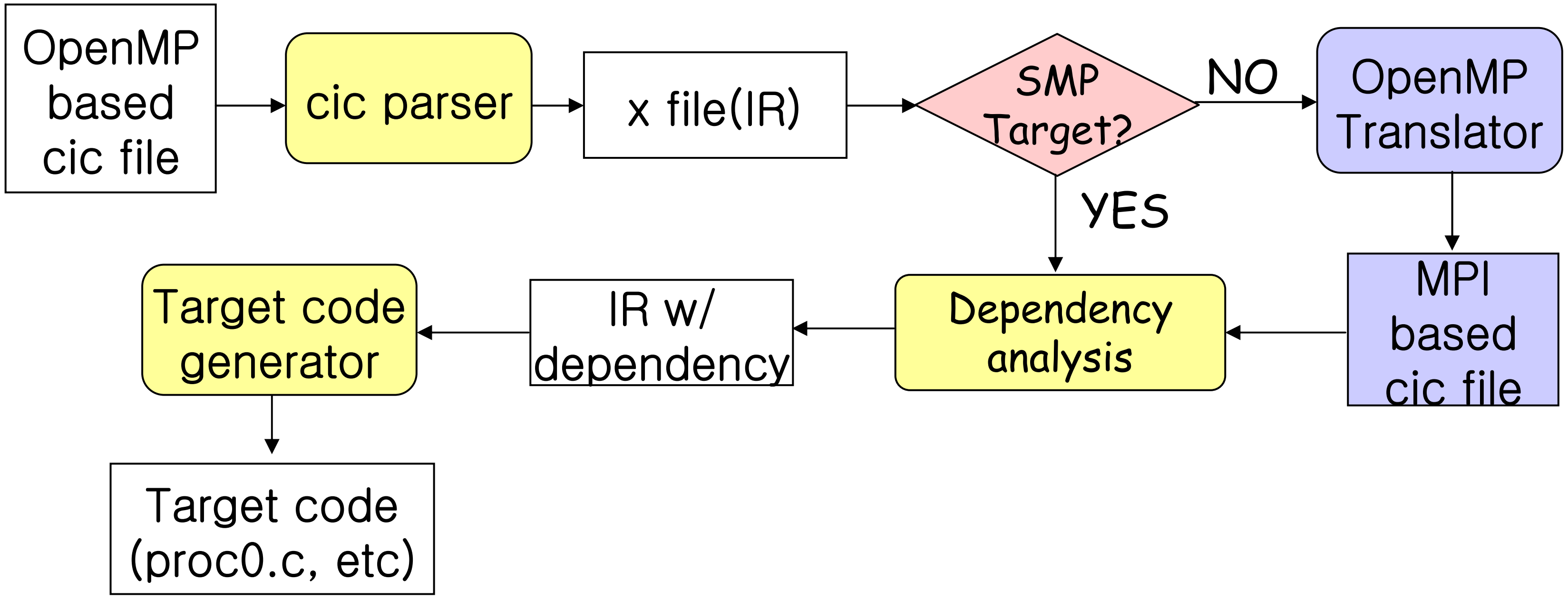
## Task Code: Structure

- Task kernel code
  - _init(): before main loop
  - _go(): in the main loop
  - _wrapup(): after main loop

```
(example) h263dec.cic
 // header file
 // global declaration and definition
 // procedure definition
     h263dec_init() { … }
     h263dec_go() { … }
     h263dec_wrapup() { … }
```

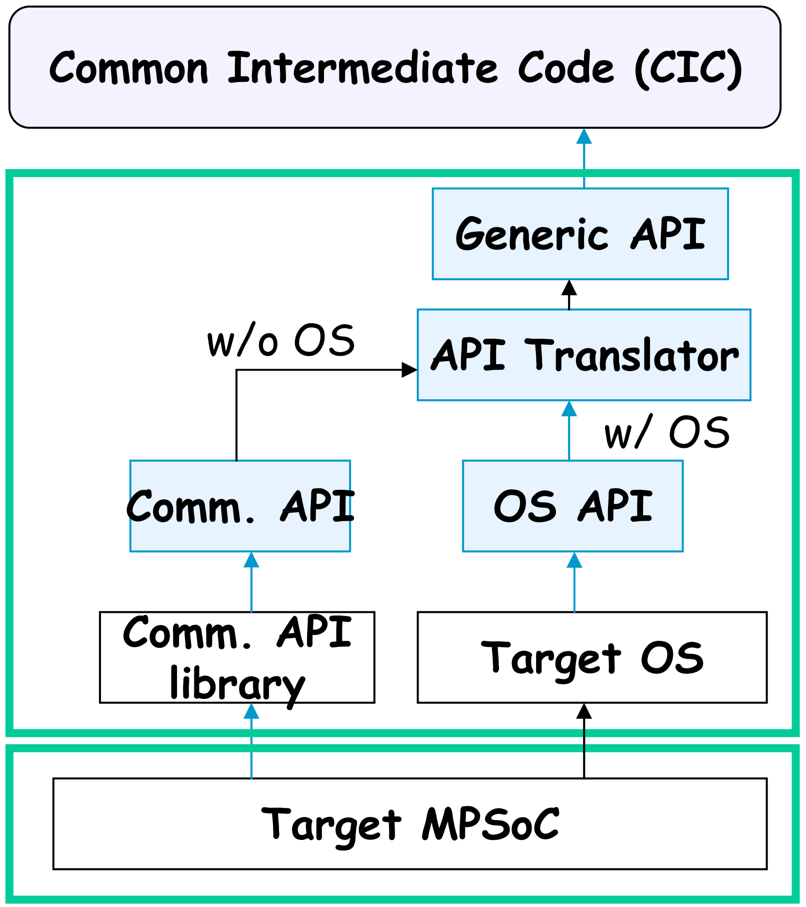
# CIC Translator

- Two types of parallelism
  - Task parallelism: separate task_name.cic files
  - Data parallelism: openMP program
- Create main function for each processor



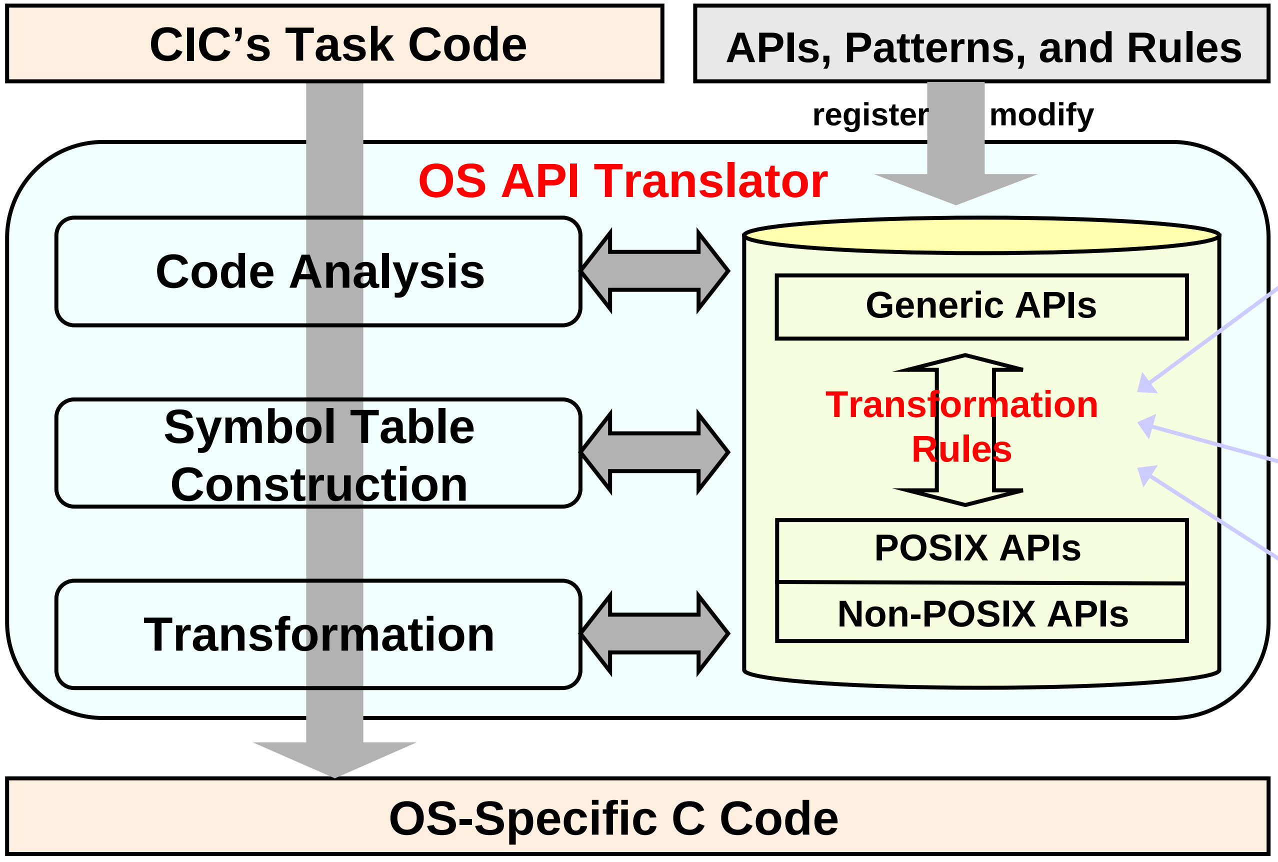
## Generic API

- Generic API
  - OS-indepedpnent API
  - Abstraction from IEEE POSIX 1003.1-2004
  - OS services:
    - Process, memory, synchronization, file system, networking, real-time, thread, I/O, timer, etc.
  - C library functions based on use frequency
    - stdio.h, stdlib.h, time.h, signal.h



## Internal Structure of API Translator

HOPES



**Translation pattern and rules**

**Pattern mapping**
- 1-to-1, 1-to-m, n-to-m
- initialize/finalize
- acquire/release

**parameters**
- type, number, value

**Variable names**
- naming of additional variables

# Static Analysis of Embedded C Code

- Static analysis
  - Buffer overrun error: “Arirac”
  - Detection of unused memory region after definition: “umirac”
  - Stack is estimation: “Stan”

# Airac5

- Static Analyzer for Detecting All Buffer Overrun Errors in C Programs
- Keywords
  - static program analysis: abstract interpretation
  - C: ANSI C + somje GNU C extensions
  - all: detects all buffer-overrun places
  - statically: at compile-time
  - always stops: finishes even for infinite-loop programs
  - modular: separate C files
  - correct: based on a firm theoretical framework

# Airac5's Performance

on P4, 3.2GHz, 4GB

Linux kernel-2.6.4

| Software | #Lines | Time(sec) | #Alarms | #Bugs |
|---|---:|---:|---:|---:|
| vmax301.c | 246 | 7.42 | 1 | 1 |
| cdc-acm.c | 849 | 17.66 | 7 | 2 |
| atkbd.c | 944 | 4.20 | 3 | 2 |
| eata-pio.c | 984 | 15.45 | 16 | 1 |
| ip6_output.c | 1110 | 47.69 | 0 | 0 |
| xfrm_user.c | 1201 | 56.52 | 4 | 1 |
| keyboard.c | 1256 | 90.42 | 31 | 1 |
| af_inet.c | 1273 | 186.06 | 12 | 2 |
| usb-midi.c | 2206 | 53.98 | 12 | 4 |
| mptbase.c | 6158 | 206.04 | 12 | 1 |

# Contents

## Status (1)

- Project
  - Title: Development of Embedded software design and verification techniques for MPSoC, Period: 2005.3.1 - 2008.2.29,
  - Amount: $4,200,000
  - Sponsor: Korean Ministry of Information and Communication

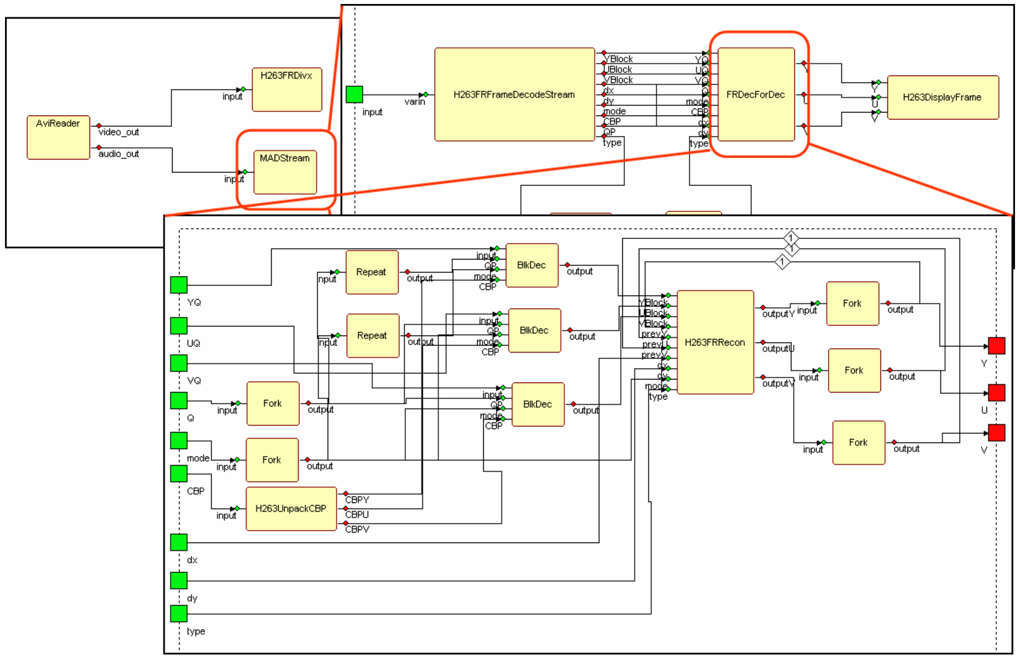
- 1st year (2005.3 - 2006.2)
  - Construct the proposed design flow for single processor target
  - Define CIC format and software interface

# Status (2)

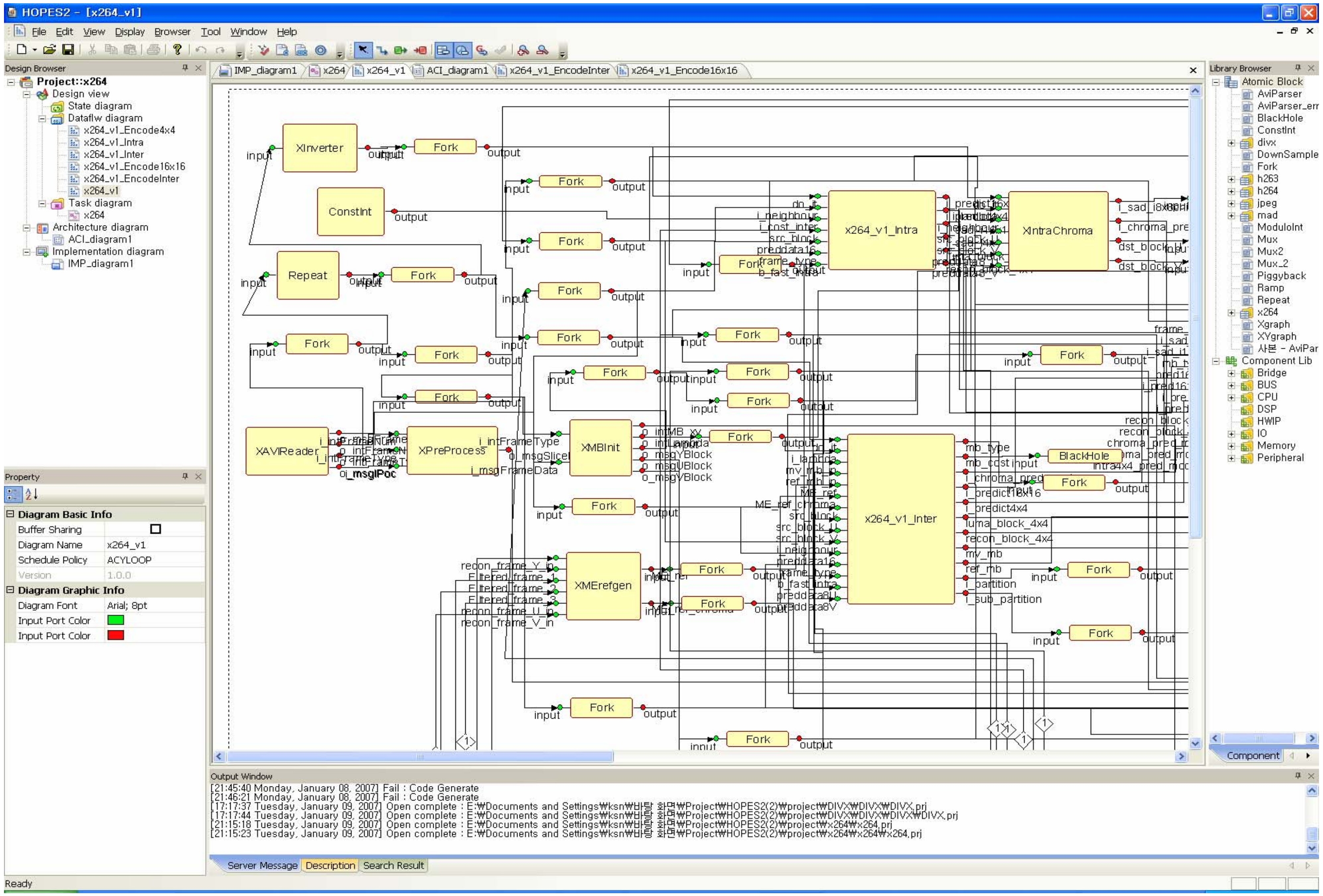
- 2nd year (2006.3 - 2007.2)
  - Extend the design flow to multiprocessor systems
    - Target independent: communication architecture and OS
  - Target independent framework is almost set-up.
- 3rd year (2007.3 - 2008.2)
  - Application to various target architectures with real applications
    - Virtual prototyping
    - Array processor of mtekVision inc.: ARM9 + SIMD engine
    - MPCore: SMP of ARM11

# HOPES Design Flow

- Model-based programming (ex: divx player)
  - PeaCE Model
    - Dataflow + (FSM) + task model
  - Fault simulation
    - sample rate inconsistency (Y;U;V= 3:1:1)
    - deadlock error
  - Functional simulation
    - Host machine
  - C code static analysis
    - Buffer overrun error detection
    - Fault simulation: wrong array size in “AviFileReader task”
  - Runtime verification
    - Simulation with Realview ARM debugger
    - Fault simulation
      - **Logical error insertion**



# H.264 Encoder



Property

**Diagram Basic Info**

| | |
|---|---|
| Buffer Sharing | ☐ |
| Diagram Name | x264_v1 |
| Schedule Policy | ACYLOOP |
| Version | 1.0.0 |

**Diagram Graphic Info**

| | |
|---|---|
| Diagram Font | Arial; 8pt |
| Input Port Color | 🟩 |
| Input Port Color | 🟥 |

Output Window

```
[21:45:40 Monday, January 08, 2007] Fail : Code Generate
[21:46:21 Monday, January 08, 2007] Fail : Code Generate
[17:17:37 Tuesday, January 09, 2007] Open complete : E:₩Documents and Settings₩ksn₩바탕 화면₩Project₩HOPES2(2)₩project₩DIVX₩DIVX₩DIVX.prj
[17:17:44 Tuesday, January 09, 2007] Open complete : E:₩Documents and Settings₩ksn₩바탕 화면₩Project₩HOPES2(2)₩project₩DIVX₩DIVX₩DIVX₩DIVX.prj
[21:15:18 Tuesday, January 09, 2007] Open complete : E:₩Documents and Settings₩ksn₩바탕 화면₩Project₩HOPES2(2)₩project₩x264₩x264.prj
[21:15:23 Tuesday, January 09, 2007] Open complete : E:₩Documents and Settings₩ksn₩바탕 화면₩Project₩HOPES2(2)₩project₩x264₩x264₩x264.prj
```

## Conclusion

- HOPES is a newly launched project to make a embedded software development environment for MPSoCs
  - Support of diverse models
  - Target independent environment + target specific libraries
  - 3-phase verification: model-level, C code static analysis, run-time simulation
  - Integration of software modules at various stages
  - http://peace.snu.ac.kr/hopes (sorry, only Korean for now. English homepage will be open soon)

Thank you !