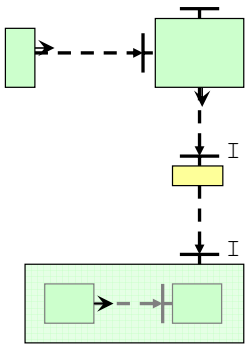
Nomadik Multiprocessing Framework, a component-based programming model for MP-SoC

Jean-Philippe FASSINO
ST Microelectronics

Component model concepts

- Components
  - A runtime entities
    - That can be manipulate
    - Multi-instances
  - The unit of development & deployment
    - No predefined granularity
    - Recursive composition (a.k.a. composite)
- Interfaces
  - Strict separation between interface and implementation
  - A component owns one or more interfaces
- Bindings
  - A communication channel between components
    - Oriented from a client to a server interface
  - Primitive (language, i.e. method call on C)
  - Composite (configuration of binding components)
    - Could enforce a inter-component communication model transparently
      - For examples RPC stubs



Multi-core Partitioning Principles

- Separate control (on Host) …
  - Multimedia network building
    - Dynamic component loading, instantiation & binding
  - Memory management of each memories
    - SDRAM, ESRAM, TCM
  - Quality of Services

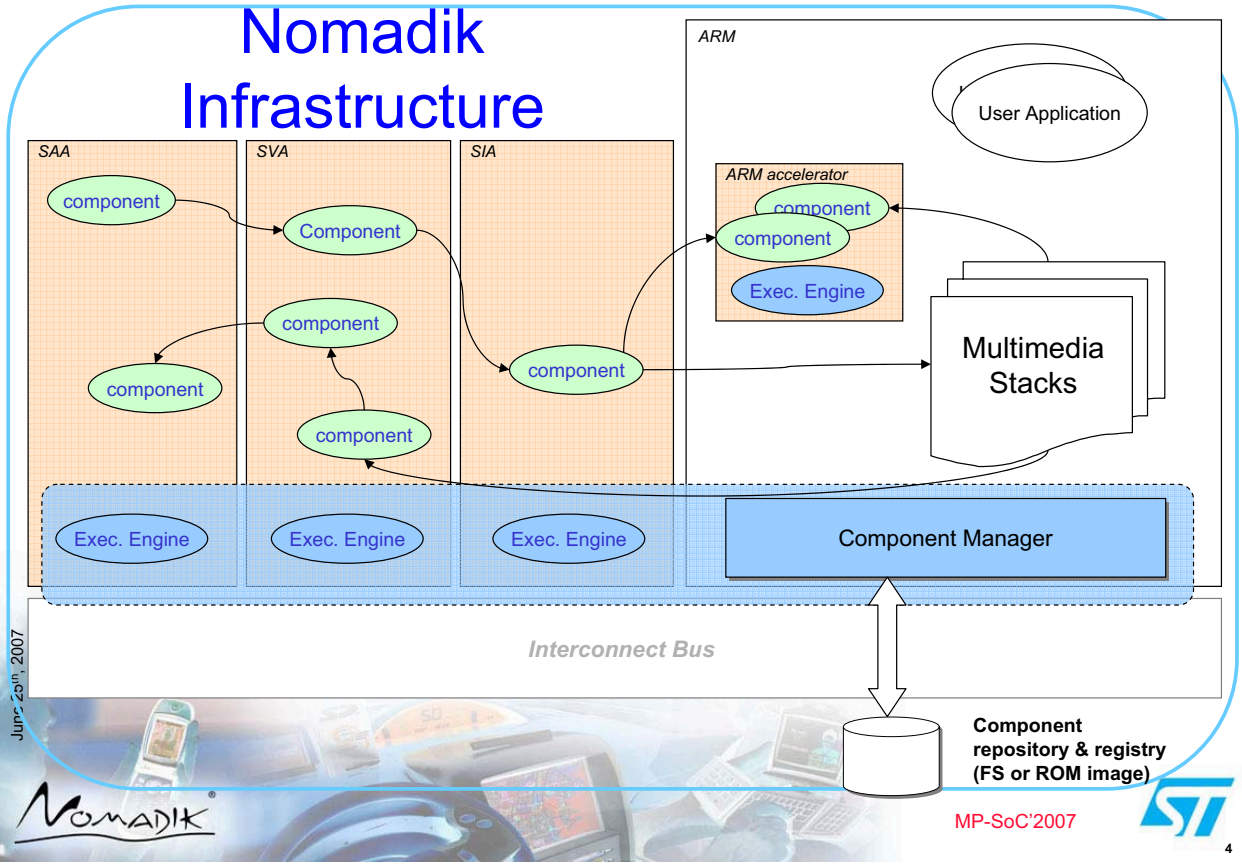
- … from multimedia processing (on MPC)
  - Light synchronous execution engine
  - Algorithms

June 25th, 2007

MP-SoC'2007

3

Nomadik Infrastructure



June 25th, 2007

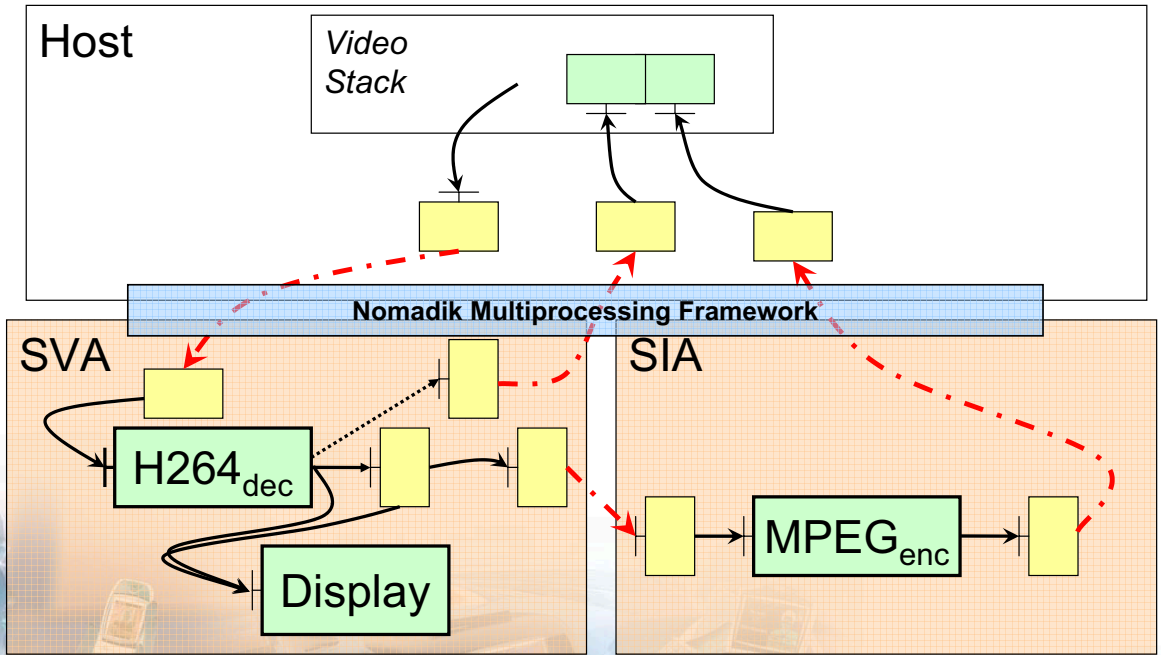
MP-SoC'2007

4

Benefits

- A multiprocessing framework for use-cases that require more than one MPC
- A component-based approach for mastering software complexity and allowing adaptation and integration with bounded effort
- The interface for hiding programmers some complexities of non functional aspects
- Flexibility and extensibility for custom proprietary software
- Dynamic loading for reducing memory footprint and increasing performance
- Code reusability for reduce time to market
- A clear model for developing powerful tools

Typical Multiprocessing use-case



Implementing a distributed vector length calculator $\sqrt{(x^2 + y^2 + z^2)}$

vector/squareadder.itf
```
void sa(t_uint32 vect[3]);
```

scalar/uniop.itf
```
void oper(t_uint32 value);
```

SquareAdder.conf
```
provides vector.squareadder as input
requires scalar.uniop as output
property minstack 128
```

SquareAdder.c
```
#include <SquareAdder.nk>
void METH(sa)(t_uint32 vect[3]) {
  int i;
  t_uint32 len2 = 0;
  for(i = 0; i < 3; i++) {
    len2 += vect[i] * vect[i];
  }
  output.oper(len2);
}
```

Sqrt.conf
```
provides scalar.uniop as A
requires scalar.uniop as push
property mpc SVA
```

Sqrt.c
```
#include <Sqrt.nk>
void METH(oper)(t_uint32 value) {
  t_uint32 square = 1;
  t_uint32 delta = 3;
  while(square <= value){
    square += delta;
    delta += 2;
  }
  push.oper(delta/2 - 1);
}
```

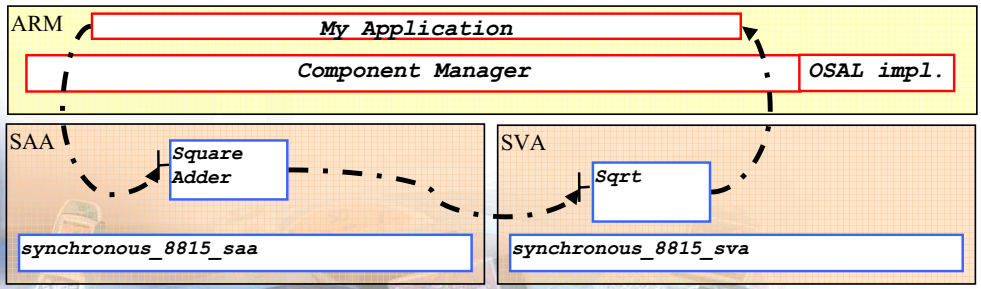
June 25th, 2007

NOMADIK

7

Deploy the network according your architecture

- **CM_instantiateComponent**(SAA, "SquareAdder", &saH);
- **CM_instantiateComponent**(SVA, "Sqrt", &sqrtH);
- **CM_bindComponent**(saH, "output", sqrtH, "A", 1);
- **CM_bindComponentFromHost**(saH, "input", &senditf, 1);
- **CM_bindComponentToHost**(sqrtH, "push", cb_itf, 1);



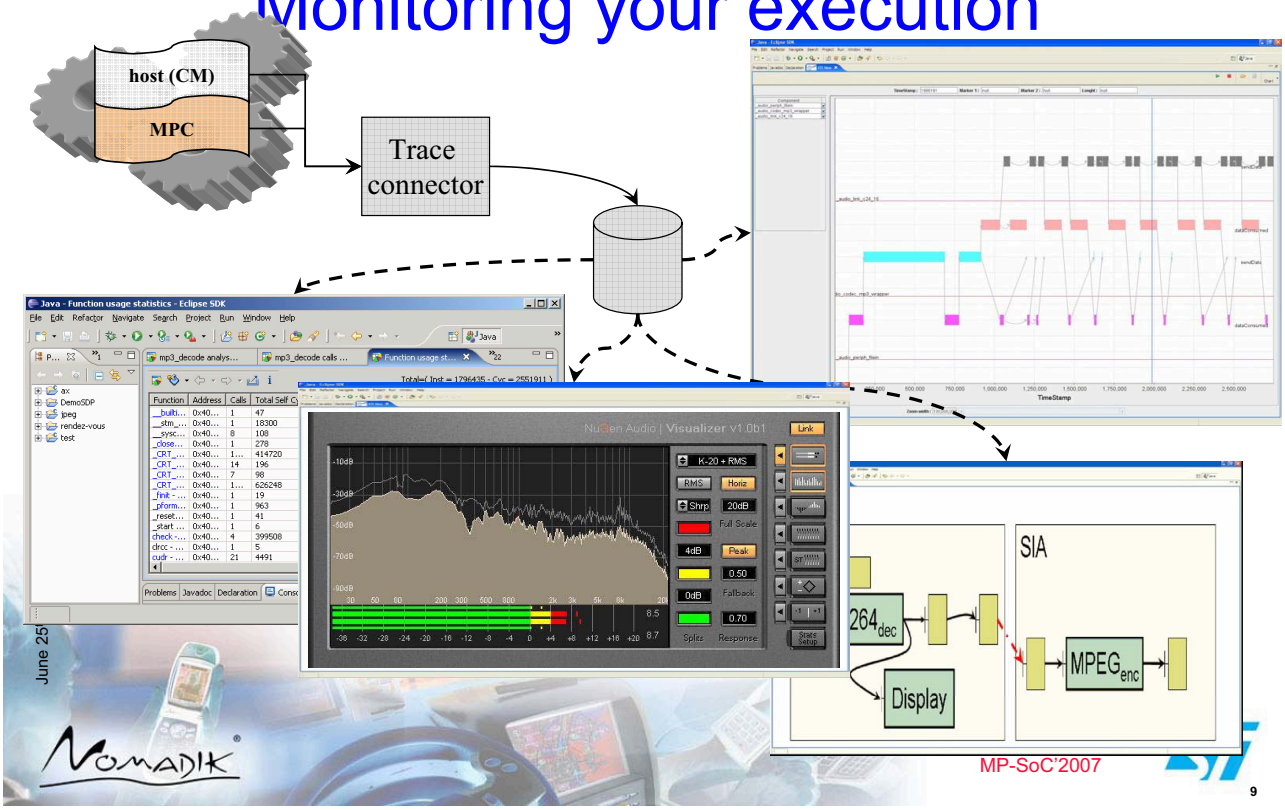
June 25th, 2007

NOMADIK

MP-SoC'2007

8

Monitoring your execution



Benchmarks

- Communication
  - Host -> MPC: 10 us
  - MPC -> Host: 10 us
  - MPC -> MPC: 8 us
- Memory footprint
  - Host component manager:
  - MPC Execution Engine: 12 KBytes
  - MPC stubs:
    - Clients: 216 Bytes
    - Server: 200 Bytes

Backup slides

Development Tool Chain