
RWTH AACHEN UNIVERSITY


C for Process Networks

Stefan Schürmans, Weihua Sheng, Anastasia Stulova, Jeronimo Castrillon

3rd Workshop on Mapping of Applications to MPSoCs,
Schloss Rheinfels, June 29th 2010

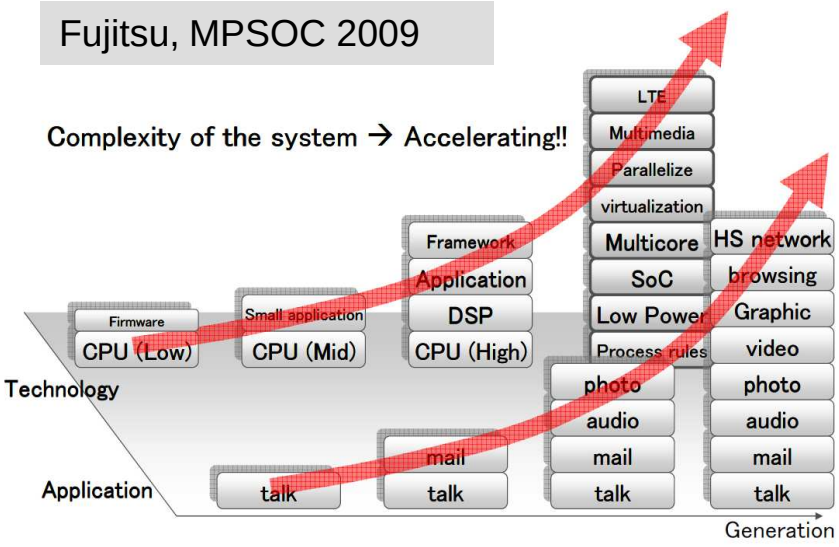
Institute for Integrated Signal Processing Systems

Outline

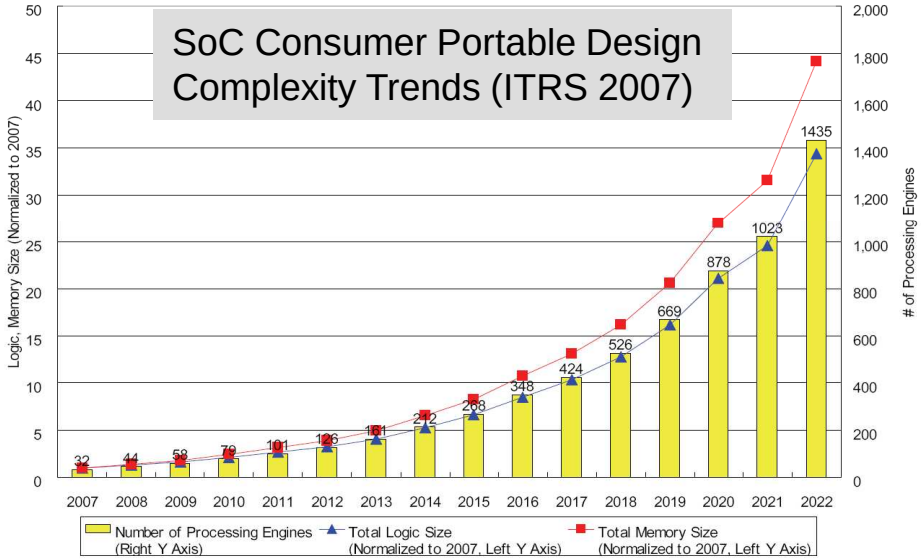
Embedded Systems use MPSoCs

- **Embedded Systems Requirements**
  - High Computational Performance
  - High Energy Efficiency

Fujitsu, MPSOC 2009

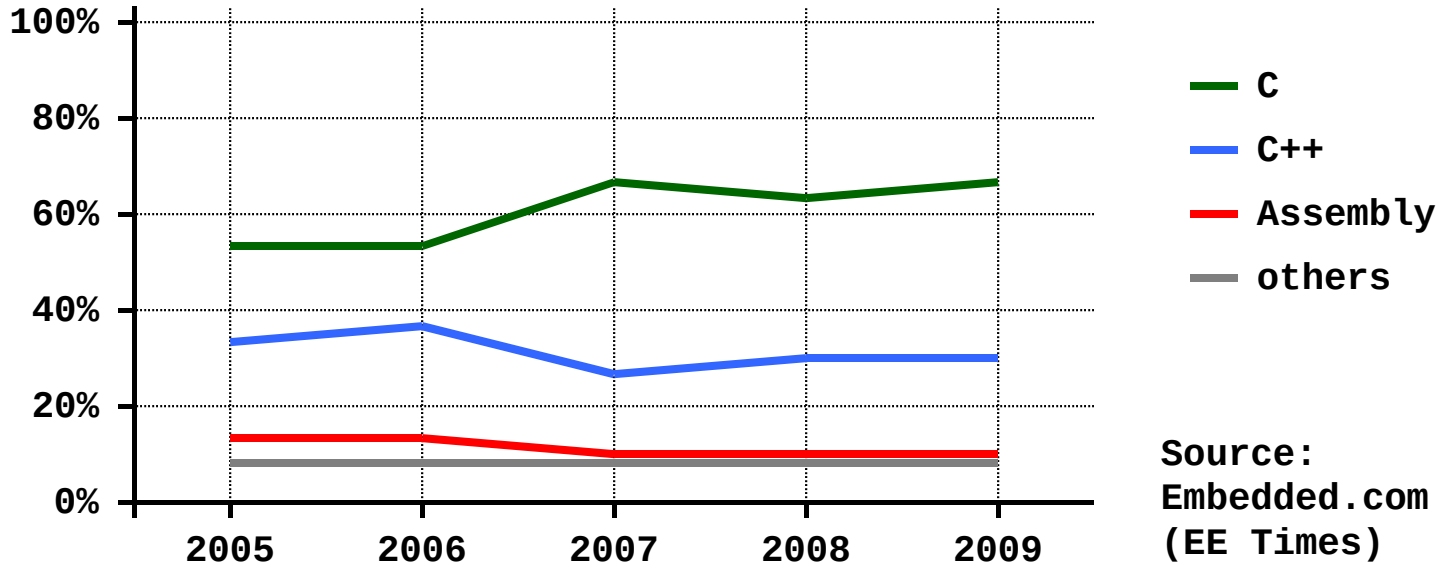


SoC Consumer Portable Design Complexity Trends (ITRS 2007)



➔ ***Parallel Hardware: MPSoCs***

Programming Languages for Embedded Systems

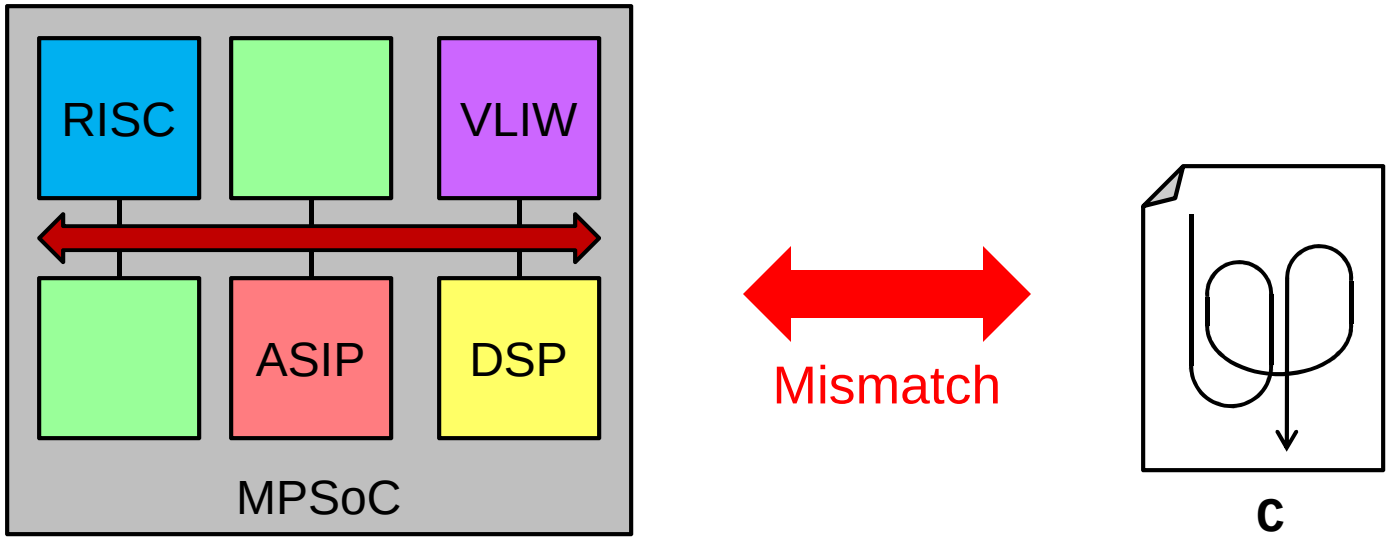


- **Dominant Embedded Systems Programming Language: C**
  - Existing legacy software
  - New projects
    - Programmers use language they know

Mismatch between HW and SW

Parallel Hardware

Sequential Software



- **Possible Solutions:**
  - Partition sequential software
  - Write parallel software
    e.g. Process Networks (PN)

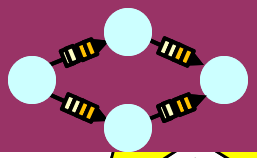
Tool Support required

Outline

MAPS (MPSoC Application Programming Studio) Approach

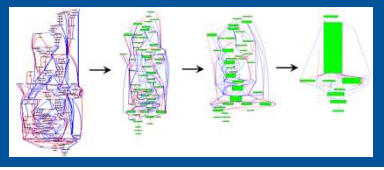
Applications Specification

- Support both sequential C or parallel models (process networks)
- Multi-App Scenarios (Real-Time constraints, interference)



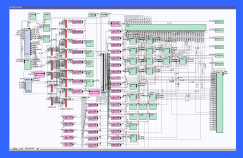
Application Partitioning

- High-level transformations to expose more parallelism
- Semi-automatic Parallelism extraction



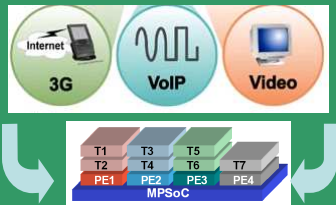
MPSoC Backends

- Code generation for MPSoCs
- Native Hardware/Virtual Platform System Simulation

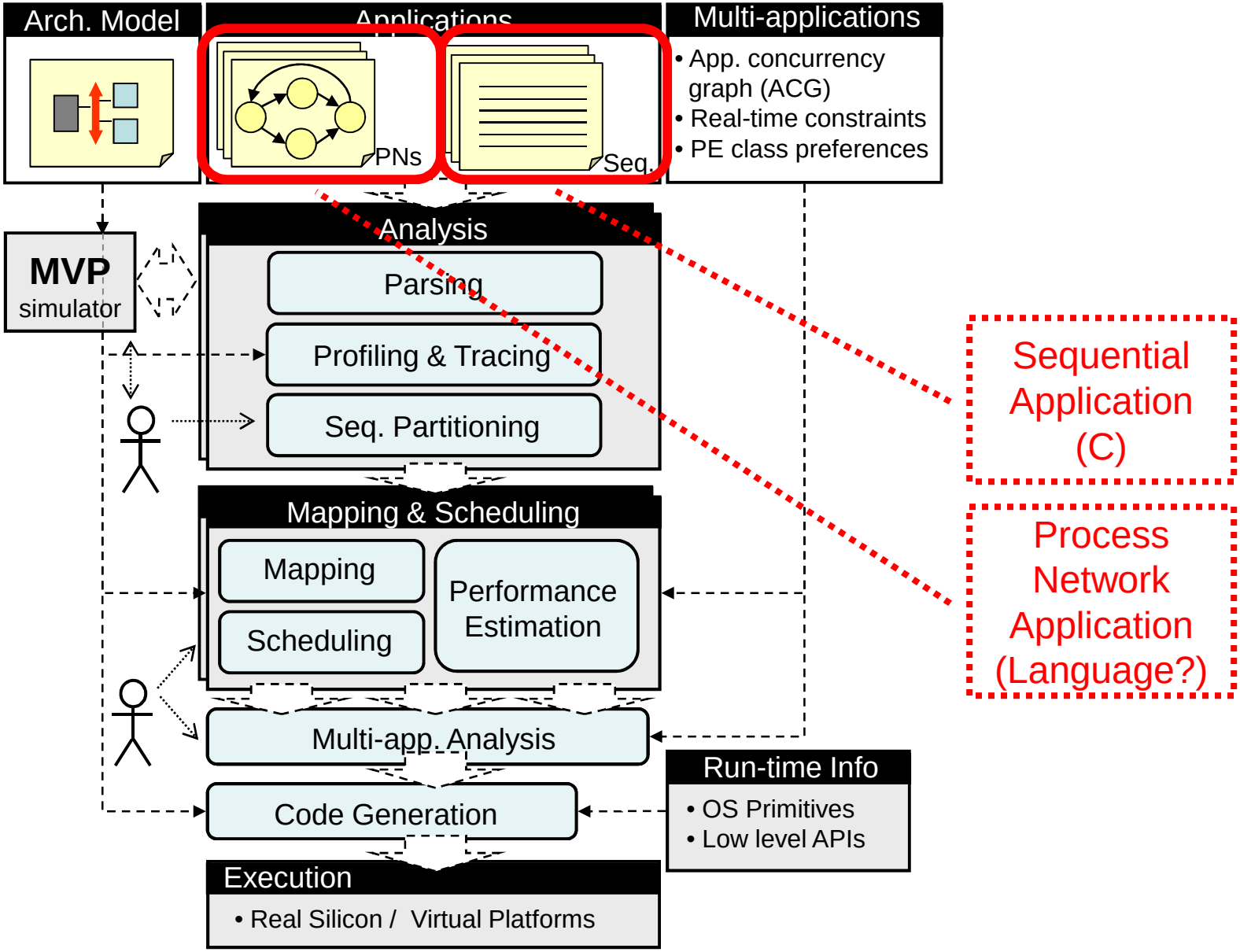


Efficient Mapping/Scheduling

- Spatial/Temporal Mapping
- Efficient/Fast Multi-task Run-time processing



MAPS Tool Flow

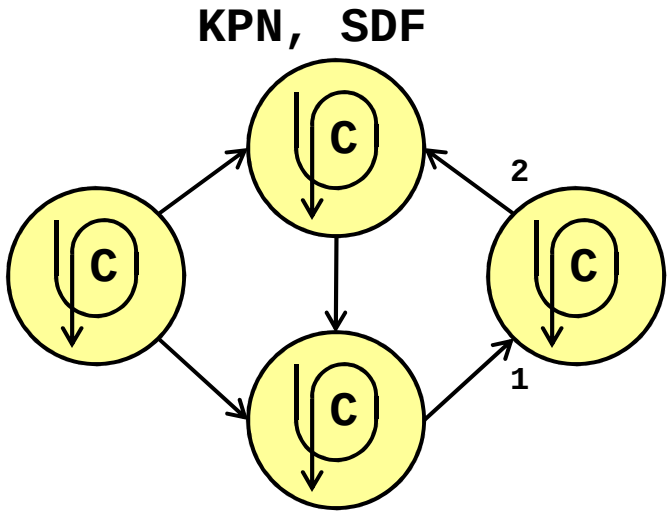


Outline

- **Process Network (PN)**
  - C code inside processes

Merge C + PN

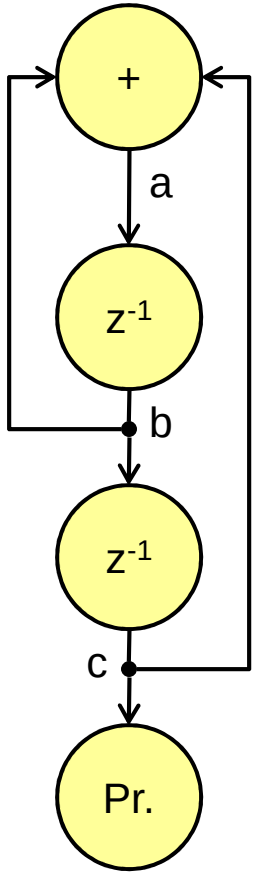
into a single language



- **C for Process Networks (CPN)**
  - Minimal language extension to C
  - New keywords
    - Processes
    - Channels
    - Channel Accesses

CPN Example: Fibonacci Numbers

```
__PNsdf Add __PNin(int u, int v)
            __PNout(int sum) {
  __PNloop { sum = u + v; }
}

__PNkpn Delay __PNin(int i)
              __PNout(int o)
              __PNparam(int first) {
  __PNout(o) { o = first; }
  while (1) {
    __PNin(i) __PNout(o) { o = i; }
  }
}

__PNsdf Print __PNin(int wnd : 2) {
  __PNloop {
    printf("%d %d\n", wnd[0], wnd[1]);
  }
}

__PNchannel int a, b, c;

__PNprocess add = Add __PNin(b, c) __PNout(a);
__PNprocess delay1 = Delay __PNin(a) __PNout(b) __PNparam(1);
__PNprocess delay2 = Delay __PNin(b) __PNout(c) __PNparam(0);
__PNprocess print = Print __PNin(c);
```



Outline

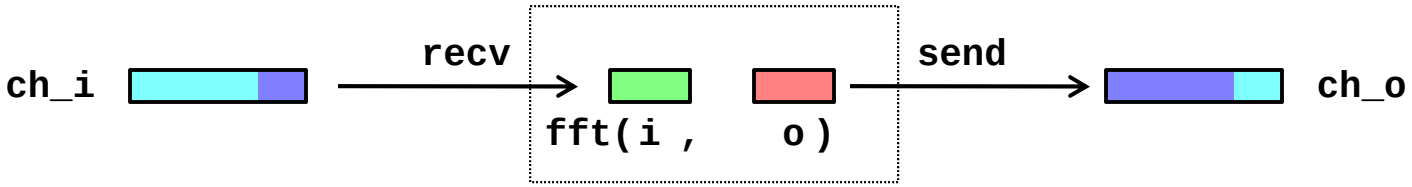
Translation of CPN to Message Passing

CPN Code

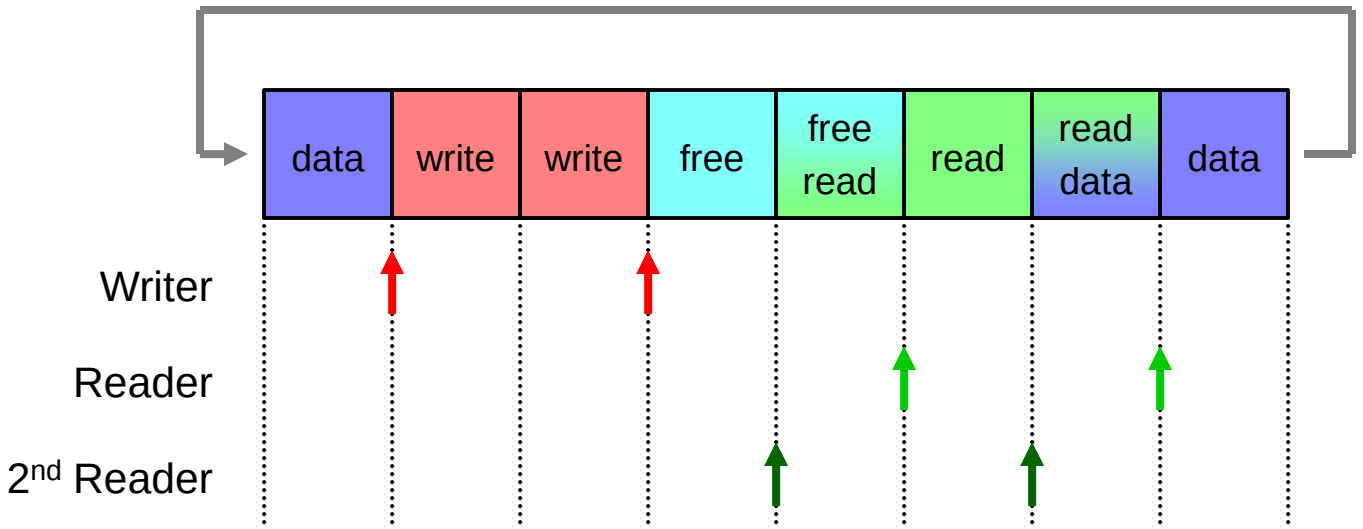
```
void fft(int i[64], int o[64]);

__PNkpn FFT __PNin(int i : 64)
            __PNout(int o : 64) {
  …
  __PNin(i) __PNout(o) {

    fft(i, o);

  }
  …
}
```

C Code for Message Passing

```
…
{
  int i[64], o[64];
  recv(ch_i, i, 64);

  fft(i, o);

  send(ch_o, o, 64);
}
…
```



- **FIFO Channel in global Shared Memory**
  - Cyclic buffer
    - Part filled with data, free part
  - Windowed Access (WA)
    - Parts being written and read
  - Multiple Readers (MR)
    - Replication of reader information



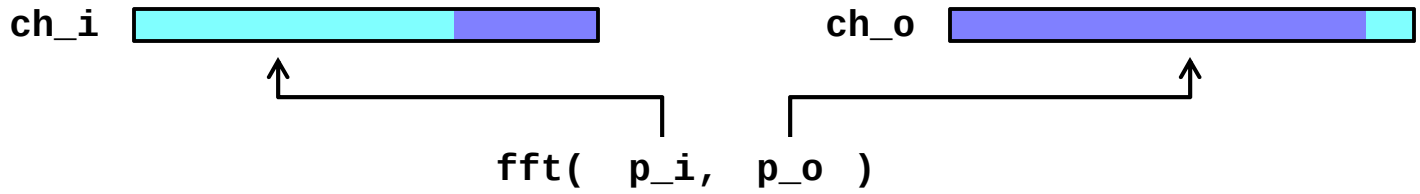
Translation of CPN for Shared Memory Systems

CPN Code

```
void fft(int i[64], int o[64]);

__PNkpn FFT __PNin(int i : 64)
            __PNout(int o : 64) {
  …
  __PNin(i) __PNout(o) {

    fft(i, o);

  }
  …
}
```

C Code for Shared Memory FIFO

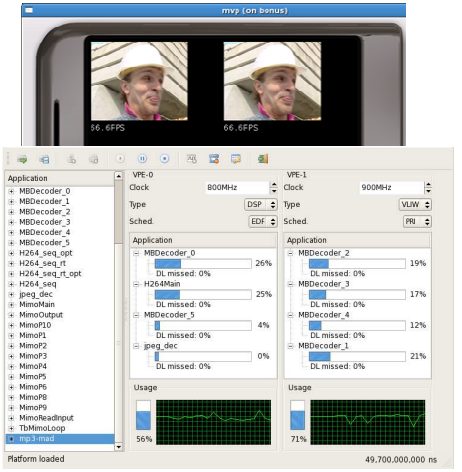
```
…
{
  int * p_i = rd_begin(ch_i, 64);
  int * p_o = wr_begin(ch_o, 64);

  fft(p_i, p_o);

  wr_end(ch_o, 64);
  rd_end(ch_i, 64);
}
…
```



Outline

Targets

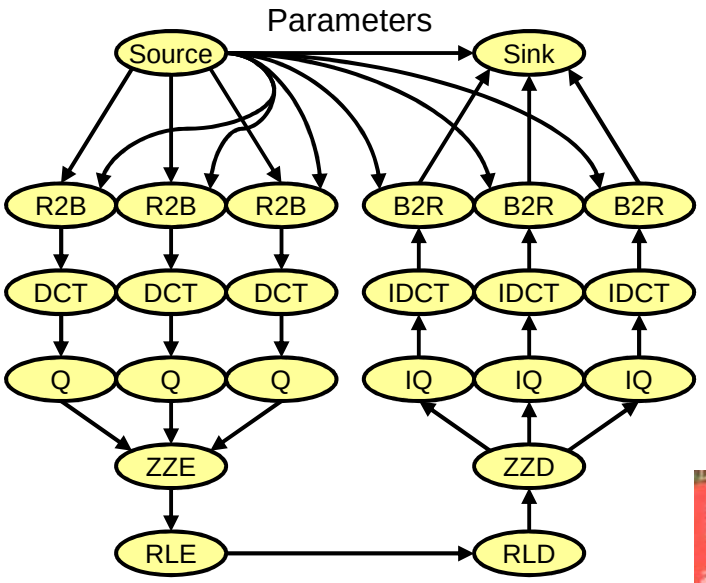
- **Pthreads**
  - Execution of application on host
- **MAPS Virtual Platform (MVP)**
  - Early Exploration of Mapping & Scheduling
- **CoWare/Synopsys ATM**
  - Abstract Virtual Platform
  - Based on Virtual Processing Unit (VPU)



Target	Shared Memory	Process	Channel
Pthreads	Yes	Host Thread	Sh. Mem. (WA + MR)
MVP	Yes	MVP Task	Sh. Mem. (WA + MR)
ATM	No	VPU Task	System C FIFO

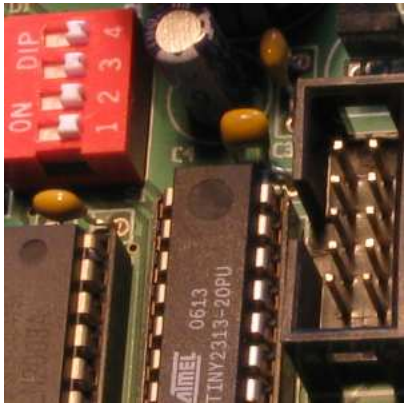
JEPG Encoder / Decoder

Task Graph

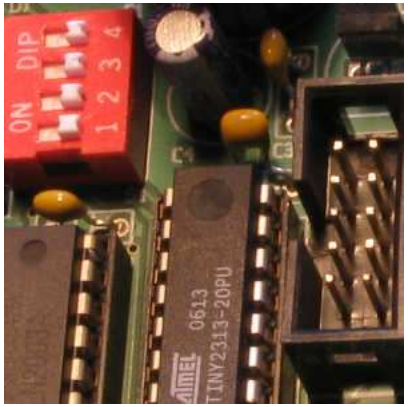


Difference
(amplified 16x)

Same Results on all 3 Targets



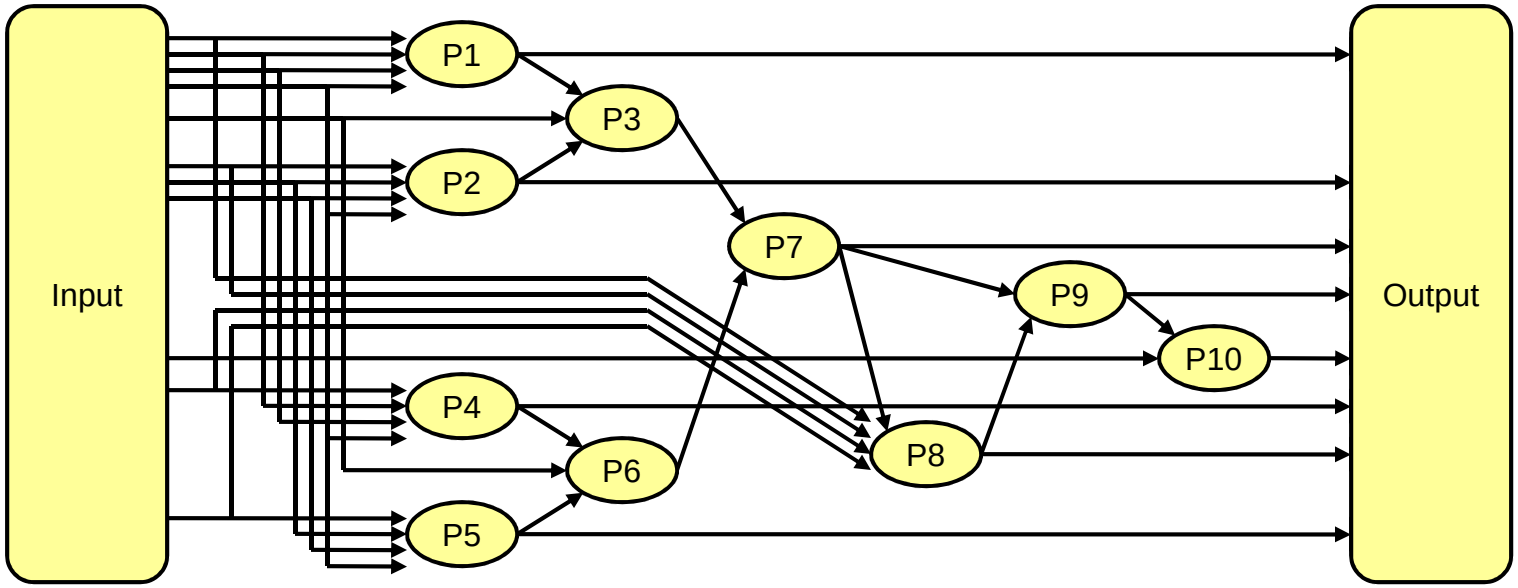
Input



Output

MMSE Processing in MIMO Receiver

Task Graph



Input

Test Vectors

Reference

Expected Results

=*

Output

Results

*) on all 3 Targets

Outline

- Introduction
- MPSoC Application Programming Studio
- C for Process Networks
- Generated Code
- Targets & Applications
- Summary & Outlook

Summary & Outlook

- **MPSoC programming requires tool support**
  - MAPS: MPSoC Application Programming Studio
- **CPN to write PN applications**
  - Minimal extension to C
  - Concise and expressive
- **Code Generation**
  - Portable to different targets
    - Use special features of target → Efficiency
- **Outlook:**
  - Code generation for real hardware platforms
  - Evaluate additional applications

Thank You!

Questions?