Tools and dataflow-based programming models for heterogeneous MPSoCs

Jeronimo Castrillon
Chair for Compiler Construction
TU Dresden
jeronimo.castrillon@tu-dresden.de

PEGPUM Workshop. HiPEAC Conference
Amsterdam, January 21st 2015

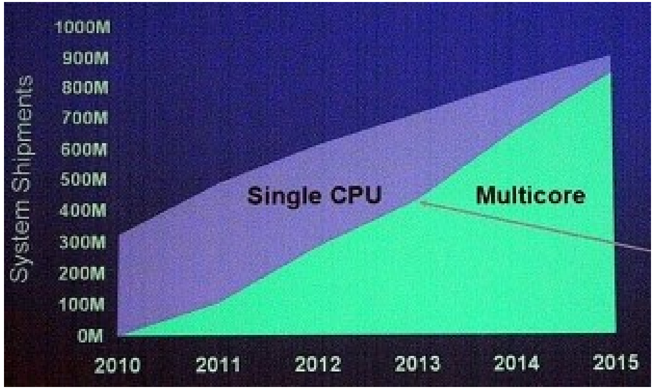
Acknowledgements

- German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)
- Collaborative research center (CRC): Highly Adaptive Energy-Efficient Computing (HAEC)
- Silexica Software Solutions GmbH

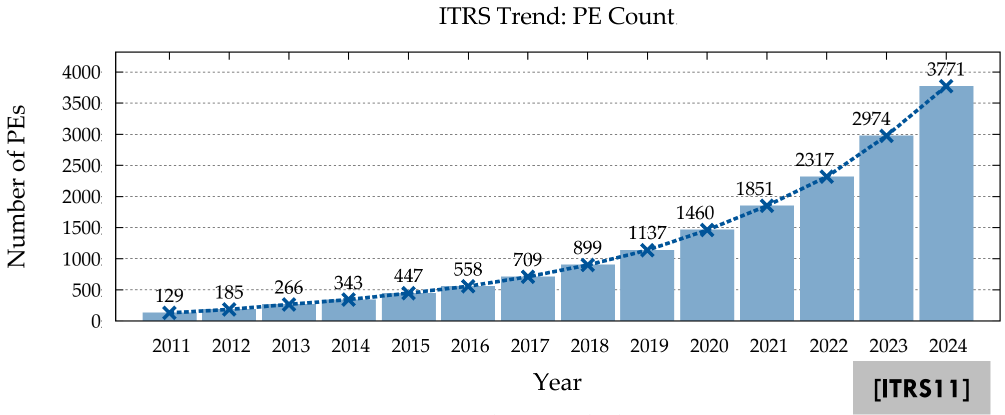
CRC 912: Highly Adaptive Energy-Efficient Computing

Multi-Processor on Systems on Chip (MPSoCs)

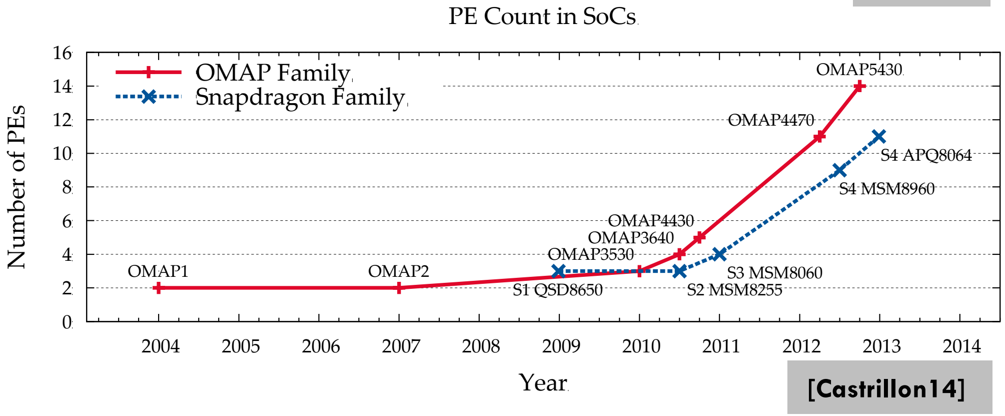
- ❑ **HW complexity**
  - ❑ Increasing number of cores
  - ❑ Increasing heterogeneity
- ❑ Multi-cores everywhere
  - ❑ Ex.: Smartphones, tablets and e-readers



[EETimes11]



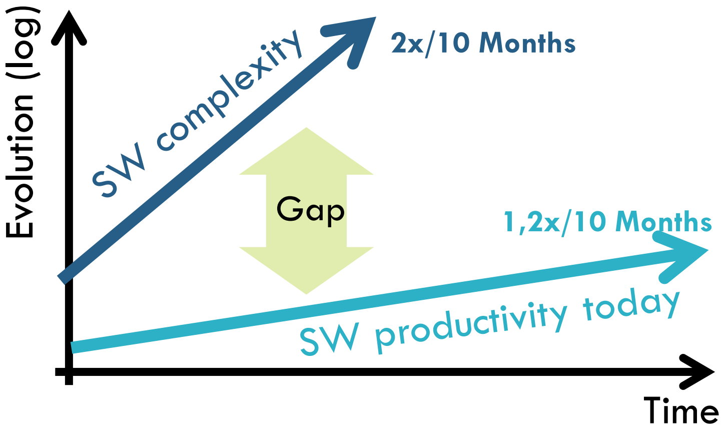
[ITRS11]



[Castrillon14]

MPSoCs: SW productivity gap

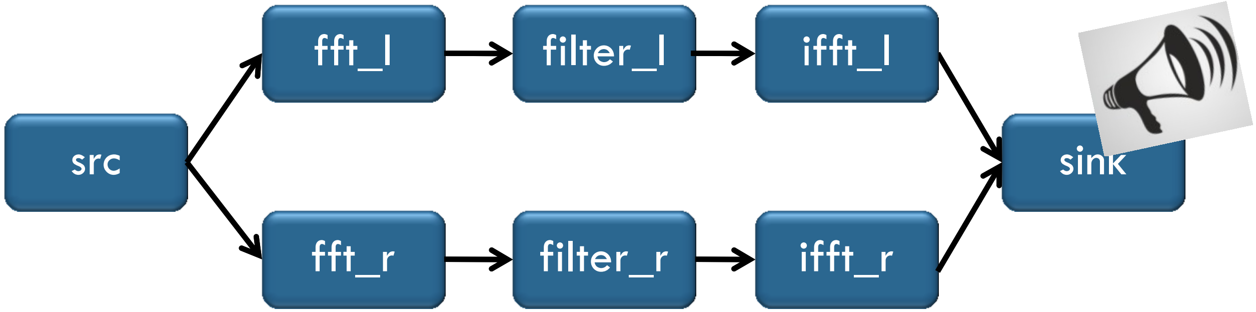
- ❑ SW-productivity gap: complex SW for ever-increasing complex HW
  - ➔ Cannot keep pace with requirements
  - ➔ Cannot leverage available parallelism
- ❑ Difficult to reason about time constraints
  - ❑ Even more difficult about energy consumption

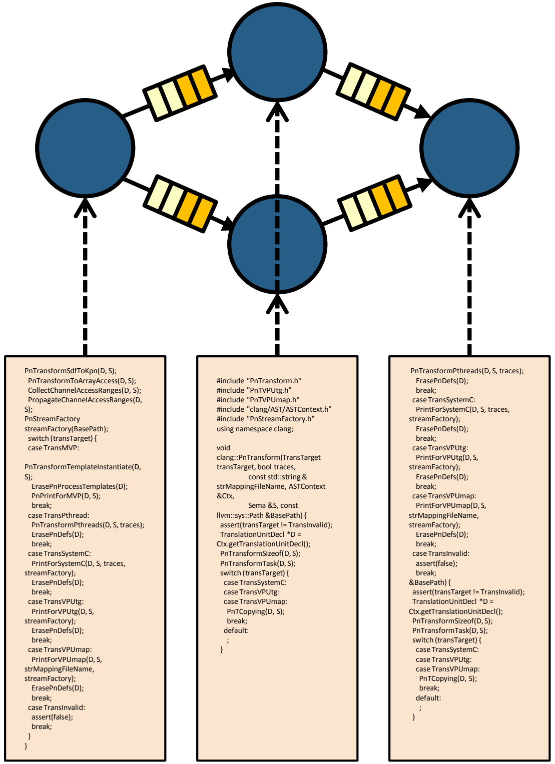


- ❑ Need domain-specific programming tools and methodologies!
  - ➔ In this presentation: **Dataflow/process networks for signal processing and multimedia**

Dataflow and process networks

- Graph representation of applications
  - Implicit repetitive execution of tasks
  - Good model for streaming applications
  - Good match for signal processing & multi-media applications
- Stereo digital audio filter





```
PnTransformSdfToKpn(D, S);
  PnTransformToArrayAccess(D, S);
  CollectChannelAccessRanges(D, S);
  PropagateChannelAccessRanges(D,
S);
PnStreamFactory
streamFactory(BasePath);
  switch (transTarget) {
  case TransMVP:

PnTransformTemplateInstantiate(D,
S);
    ErasePnProcessTemplates(D);
    PnPrintForMVP(D, S);
    break;
  case TransPthread:
    PnTransformPthreads(D, S, traces);
    ErasePnDefs(D);
    break;
  case TransSystemC:
    PrintForSystemC(D, S, traces,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransVPUtg:
    PrintForVPUtg(D, S,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransVPUmap:
    PrintForVPUmap(D, S,
strMappingFileName,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransInvalid:
    assert(false);
    break;
  }
}
```

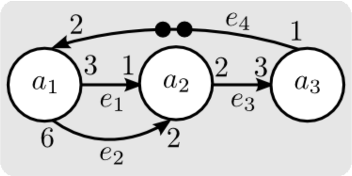
```
#include "PnTransform.h"
#include "PnTVPUtg.h"
#include "PnTVPUmap.h"
#include "clang/AST/ASTContext.h"
#include "PnStreamFactory.h"
using namespace clang;

void
clang::PnTransform(TransTarget
transTarget, bool traces,
          const std::string &
strMappingFileName, ASTContext
&Ctx,
          Sema &S, const
llvm::sys::Path &BasePath) {
  assert(transTarget != TransInvalid);
  TranslationUnitDecl  *D =
Ctx.getTranslationUnitDecl();
  PnTransformSizeof(D, S);
  PnTransformTask(D, S);
  switch (transTarget) {
    case TransSystemC:
    case TransVPUtg:
    case TransVPUmap:
      PnTCopying(D, S);
      break;
    default:
      ;
  }
```

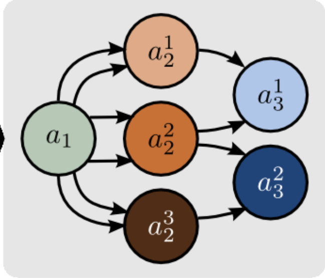
```
 PnTransformPthreads(D, S, traces);
    ErasePnDefs(D);
    break;
  case TransSystemC:
    PrintForSystemC(D, S, traces,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransVPUtg:
    PrintForVPUtg(D, S,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransVPUmap:
    PrintForVPUmap(D, S,
strMappingFileName,
streamFactory);
    ErasePnDefs(D);
    break;
  case TransInvalid:
    assert(false);
    break;
&BasePath) {
  assert(transTarget != TransInvalid);
  TranslationUnitDecl  *D =
Ctx.getTranslationUnitDecl();
  PnTransformSizeof(D, S);
  PnTransformTask(D, S);
  switch (transTarget) {
    case TransSystemC:
    case TransVPUtg:
    case TransVPUmap:
      PnTCopying(D, S);
      break;
    default:
      ;
  }
```

Dataflow models: static vs. dynamic

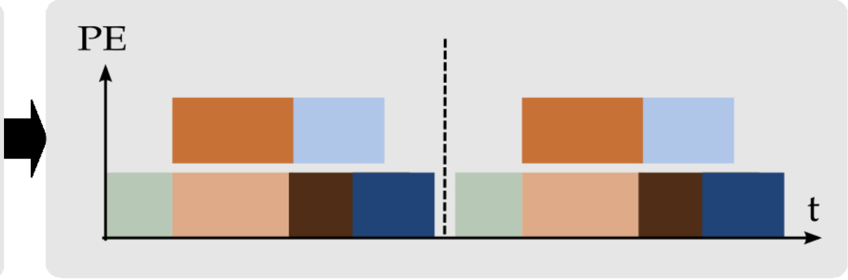
- Static: Synchronous dataflow models



Synchronous Dataflow (SDF) spec.

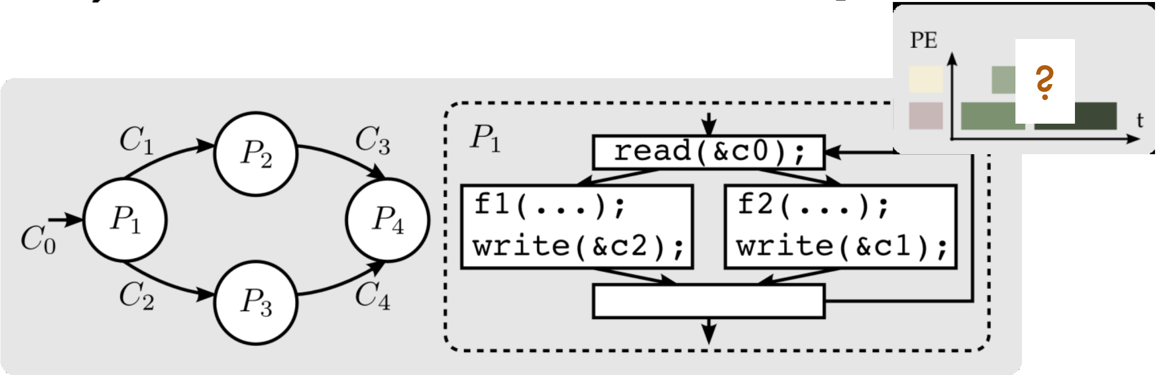


Unfolded cycle



"Blocked" DAG scheduling

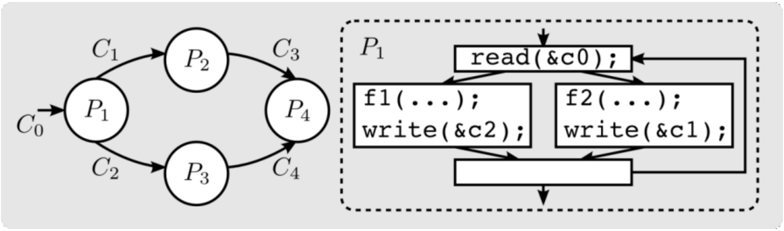
- KPNs (& DDFs)
  - No “hardcoded” rates
  - More expressiveness
  - More difficult to analyze



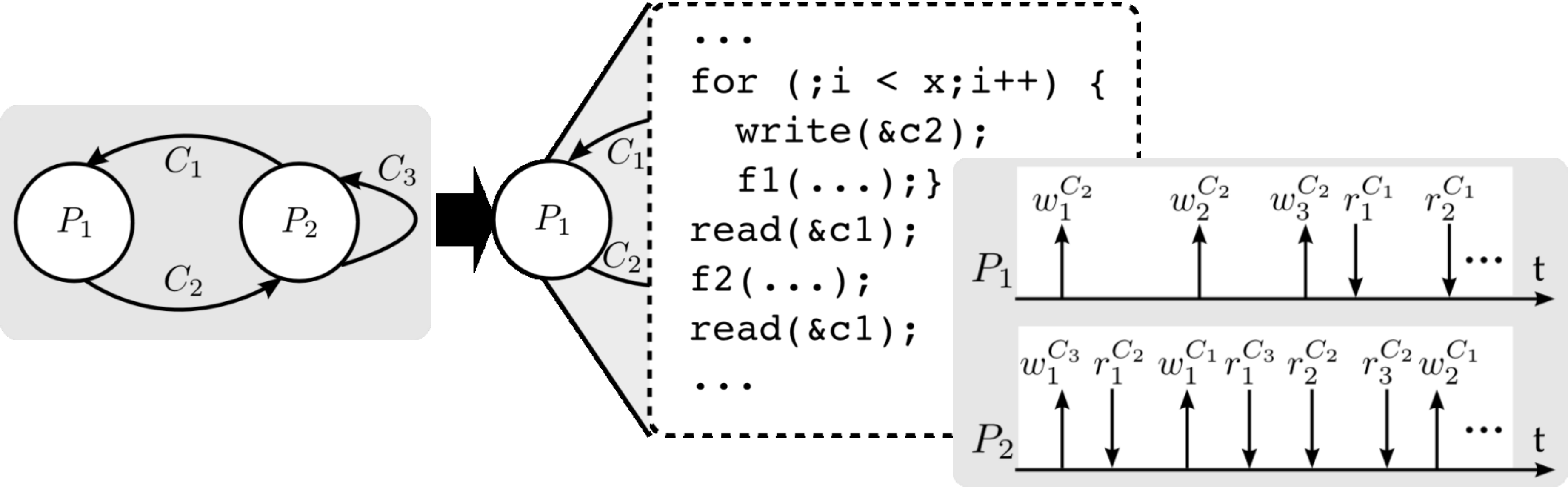
```
read(&c0);
f1(...);
write(&c2);
f2(...);
write(&c1);
```

Dealing with dynamic behavior: tracing

- ❑ **White model of processes:** source code analysis and tracing

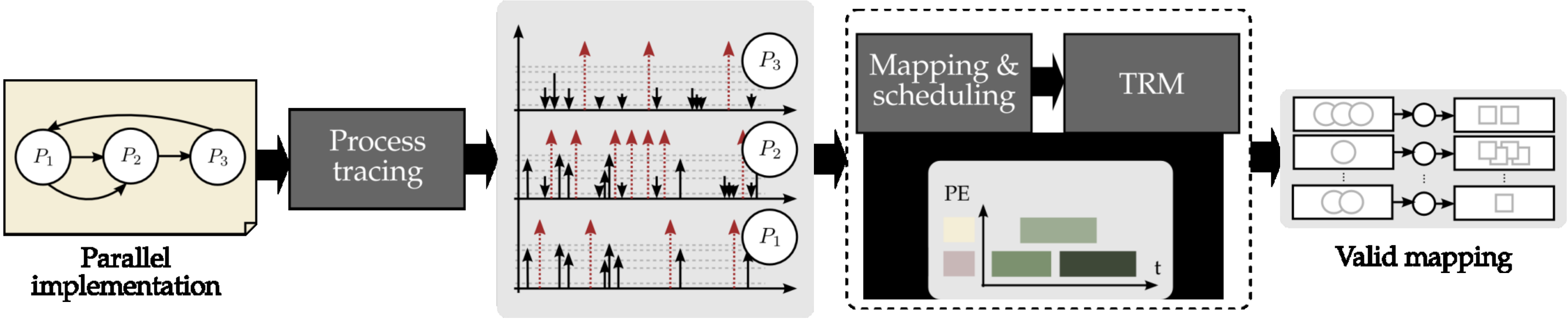


```
read(&c0);
f1(...);
write(&c2);
f2(...);
write(&c1);
```


P_1
P_2
C_1
C_2
C_3
P_1
C_1
C_2
$w_1^{C_2}$ $w_2^{C_2}$ $w_3^{C_2}$ $r_1^{C_1}$ $r_2^{C_1}$
P_1
...
t
$w_1^{C_3}$ $r_1^{C_2}$ $w_1^{C_1}$ $r_1^{C_3}$ $r_2^{C_2}$ $r_3^{C_2}$ $w_2^{C_1}$
P_2
...
t

```
...
for (;i < x;i++) {
  write(&c2);
  f1(...);}
read(&c1);
f2(...);
read(&c1);
...
```

Trace-based mapping and scheduling



Parallel implementation

Process tracing

Mapping & scheduling

TRM

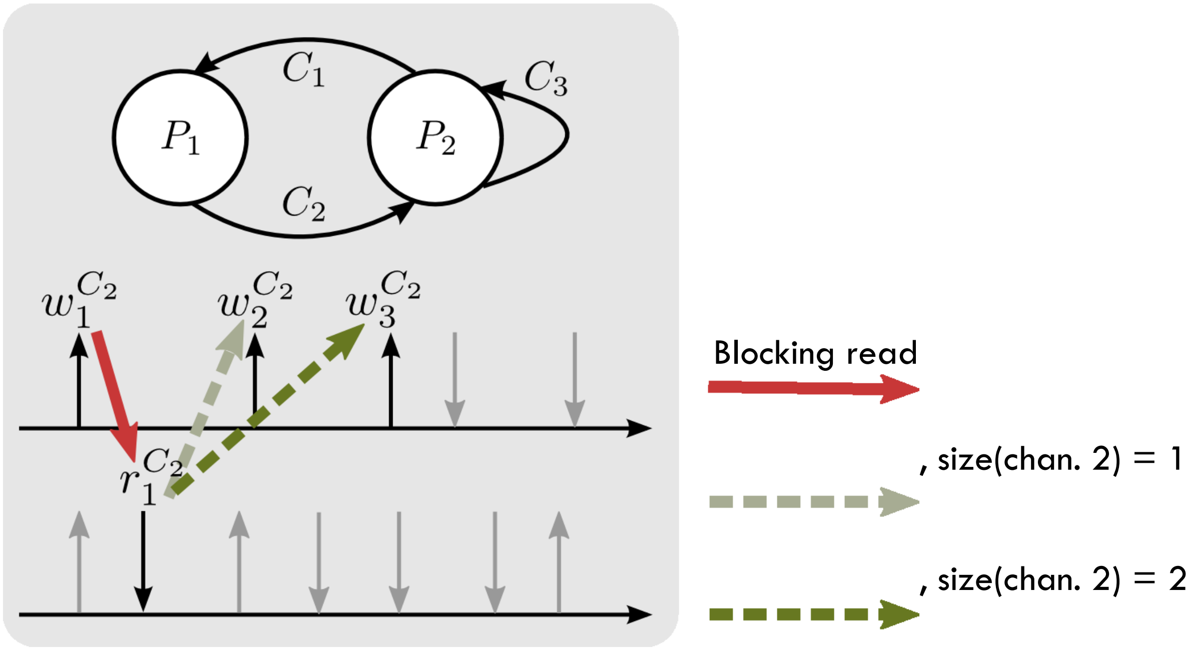
Valid mapping

- ❑ Mapping: Trace-based heuristics
  - ❑ Mapping & scheduling: Analyze traces and propose mapping
  - ❑ Iterate: Improve mapping (if required)

[DATE10, IEEE-TII13]

Trace-based algorithms

- Event traces can be represented as large dependence graphs



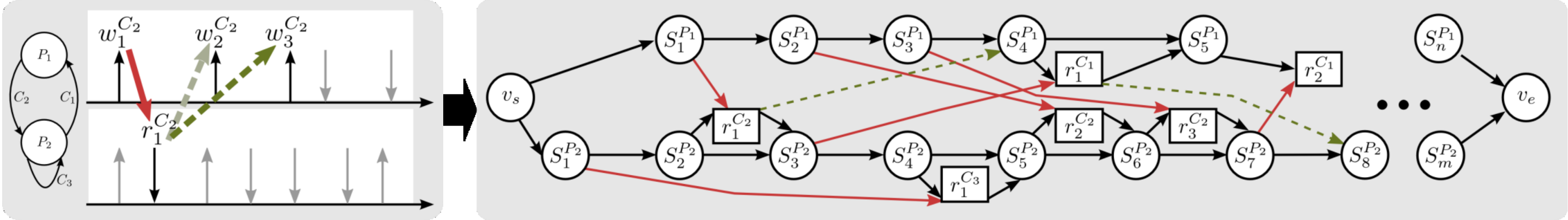
Blocking read

, size(chan. 2) = 1

, size(chan. 2) = 2

Trace-based algorithms (2)

- Sample trace graph



size(chan. 2) = 2, size(chan. 1,3) = 1

- Possible to reason about
  - Channel sizes and memory allocation
  - Mapping and scheduling onto heterogeneous processors

Dealing with heterogeneity: group-based mapping (GBM)

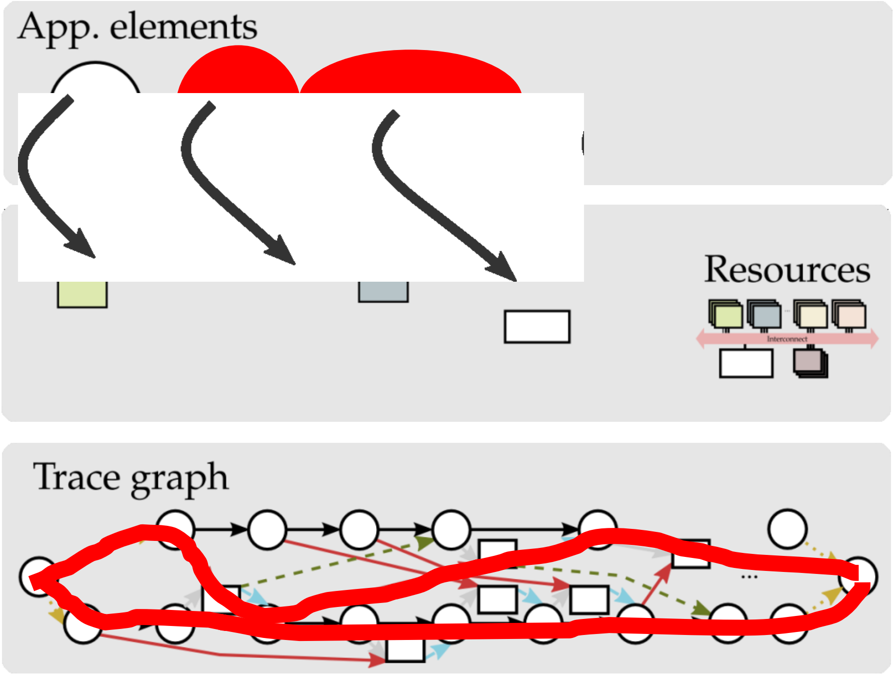
1) Initialize: All to all

2) Select element: Trace graph critical path

3) Reduce group

4) Assess & propagate

5) Quasi-homogeneous

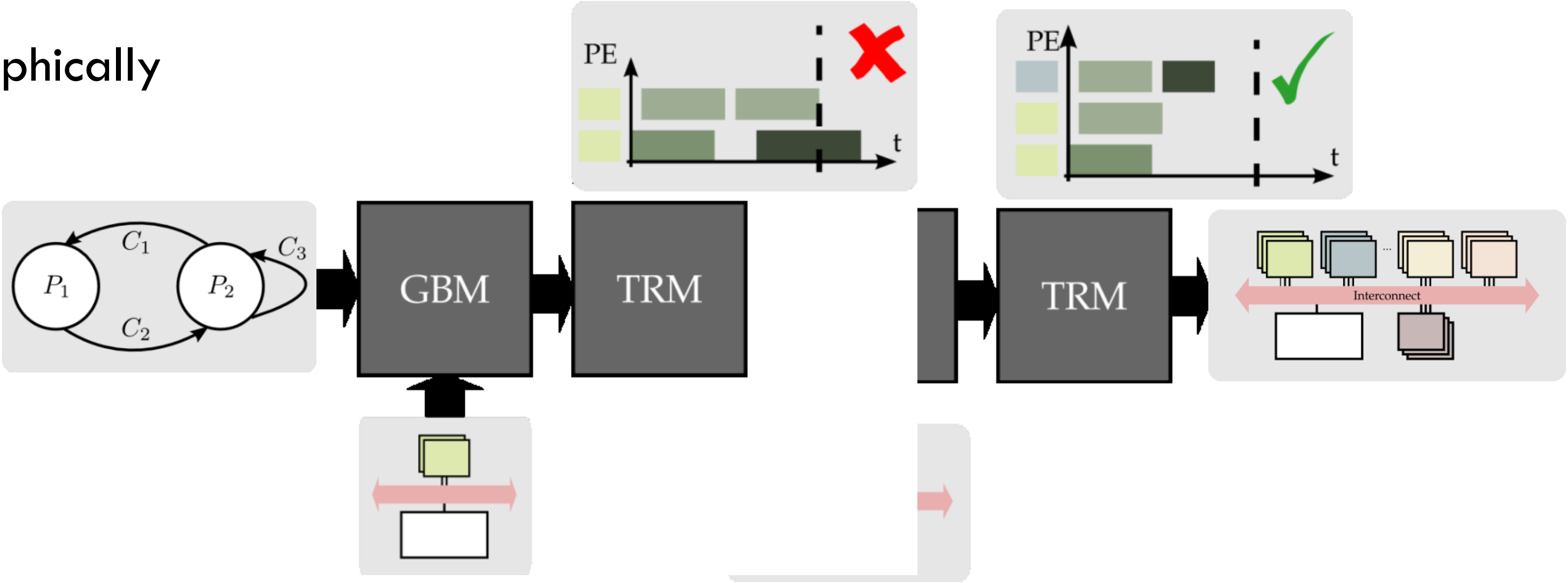


[DAC12]

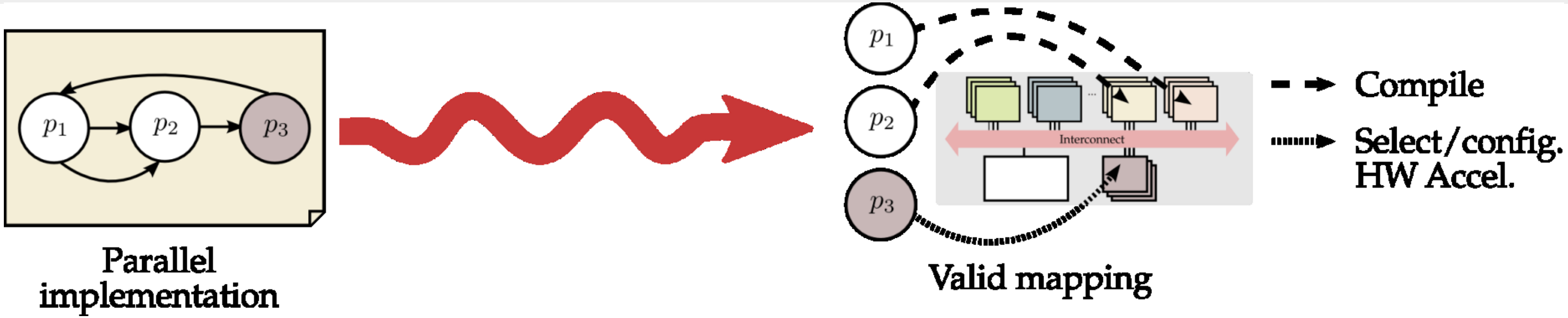
Dealing with real-time applications

- Idea: intelligently add resources until a valid mapping is found
  - Resources: memories and processors

- Graphically



Dealing with SW/HW acceleration



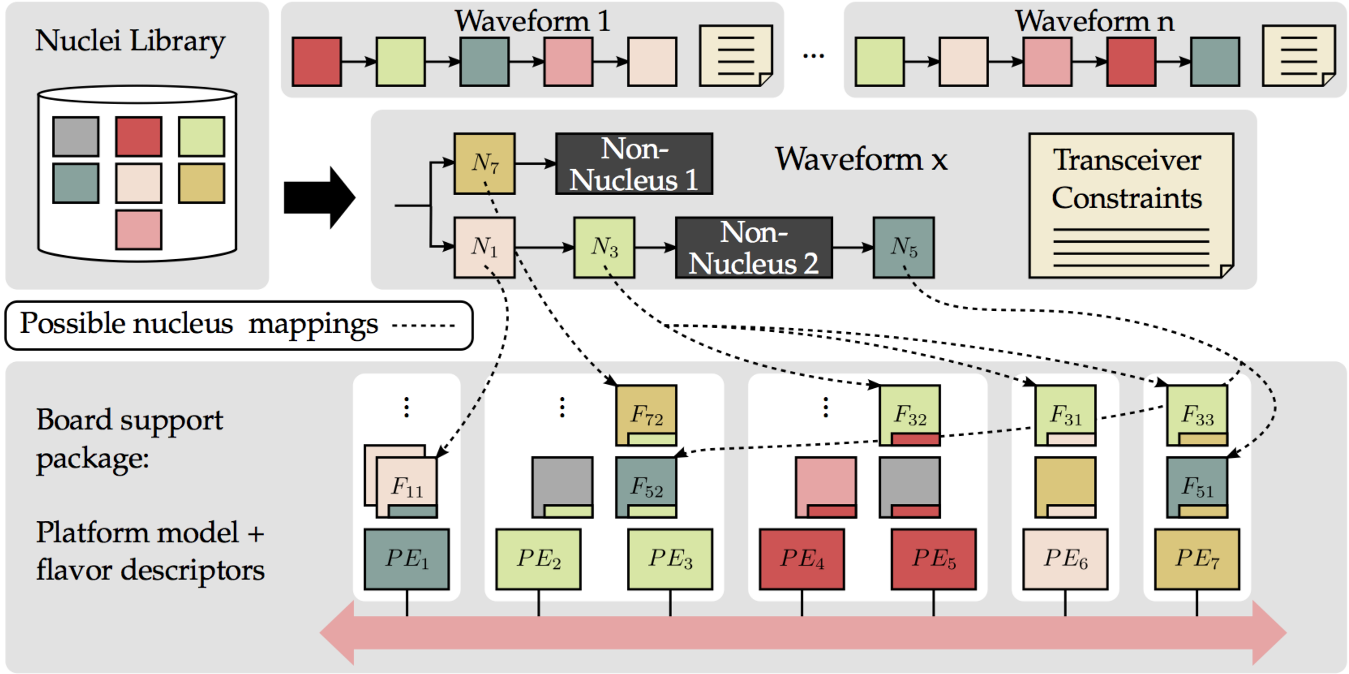
[SDR10, ALOG11]

- ❑ Specialized platforms: HW/SW acceleration
  - ❑ Compilation and source-based performance estimation **not applicable**
- ❑ Approach
  - ❑ Framework for marking accelerated routines
  - ❑ Tool flow to select and configure accelerators

Dealing with SW/HW acceleration (2)

- "Nucleus" tool flow

[MILCOM09, Castrillon14]



Implementation: C extension for KPNs

- ❑ FIFO Channels

```
typedef struct { int i; double d; } my_struct_t;
__PNchannel my_struct_t C;
__PNchannel int A = {1, 2, 3}; /* Initialization */
__PNchannel short C[2], D[2], F[2], G[2];
```

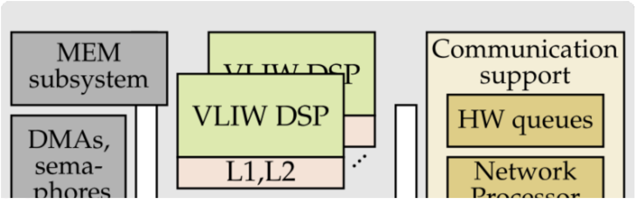
- ❑ Processes & networks

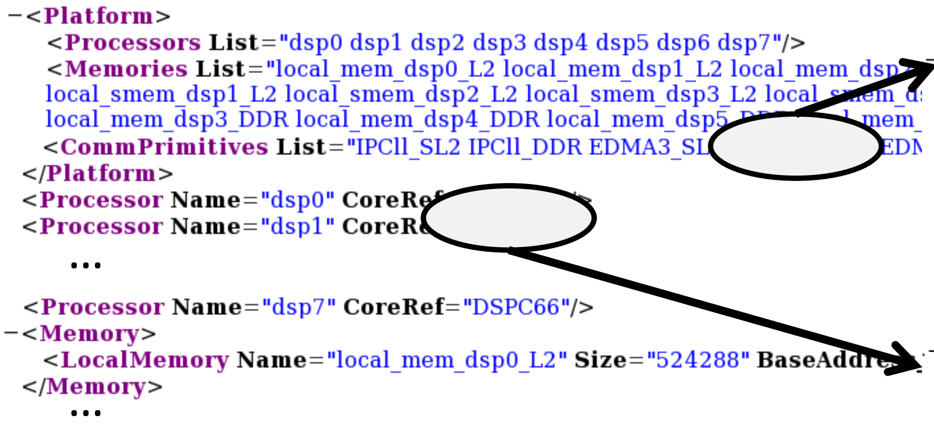
```
__PNkpn AudioAmp __PNin(short A[2]) __PNout(short B[2])
                 __PNparam(short boost){
  while (1)
    __PNin(A) __PNout(B) {
      for (int i = 0; i < 2; i++)
        B[i] = A[i]*boost;
    }}
__PNprocess Amp1 = AudioAmp __PNin(C) __PNout(F) __PNparam(3);
__PNprocess Amp2 = AudioAmp __PNin(D) __PNout(G) __PNparam(10);
```

[PARCO'14]

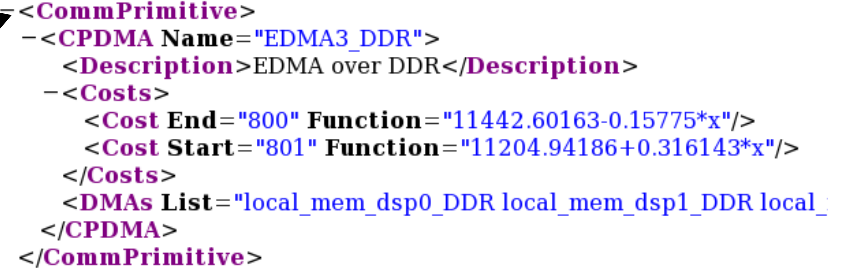
Implementation: platform model

- Example: Texas Instruments Keystone

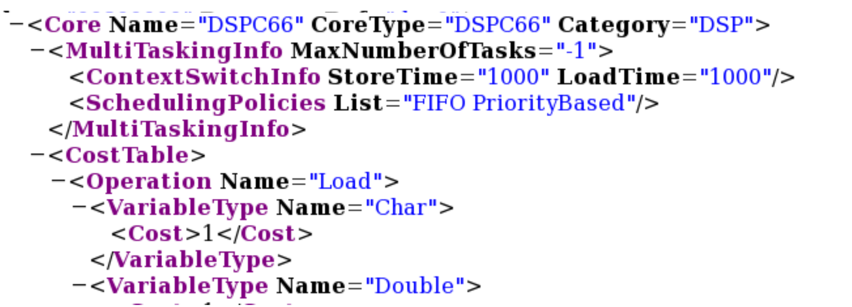




```
-<Platform>
   <Processors List="dsp0 dsp1 dsp2 dsp3 dsp4 dsp5 dsp6 dsp7"/>
   <Memories List="local_mem_dsp0_L2 local_mem_dsp1_L2 local_mem_dsp
   local_smem_dsp1_L2 local_smem_dsp2_L2 local_smem_dsp3_L2 local_smem_d
   local_mem_dsp3_DDR local_mem_dsp4_DDR local_mem_dsp5_DDR local_mem_
   <CommPrimitives List="IPCll_SL2 IPCll_DDR EDMA3_SL                EDN
 </Platform>
 <Processor Name="dsp0" CoreRef
 <Processor Name="dsp1" CoreRe
     ...
 <Processor Name="dsp7" CoreRef="DSPC66"/>
-<Memory>
   <LocalMemory Name="local_mem_dsp0_L2" Size="524288" BaseAddre
 </Memory>
     ...
```



```
-<CommPrimitive>
 -<CPDMA Name="EDMA3_DDR">
    <Description>EDMA over DDR</Description>
   -<Costs>
      <Cost End="800" Function="11442.60163-0.15775*x"/>
      <Cost Start="801" Function="11204.94186+0.316143*x"/>
    </Costs>
    <DMAs List="local_mem_dsp0_DDR local_mem_dsp1_DDR local_
  </CPDMA>
</CommPrimitive>
```

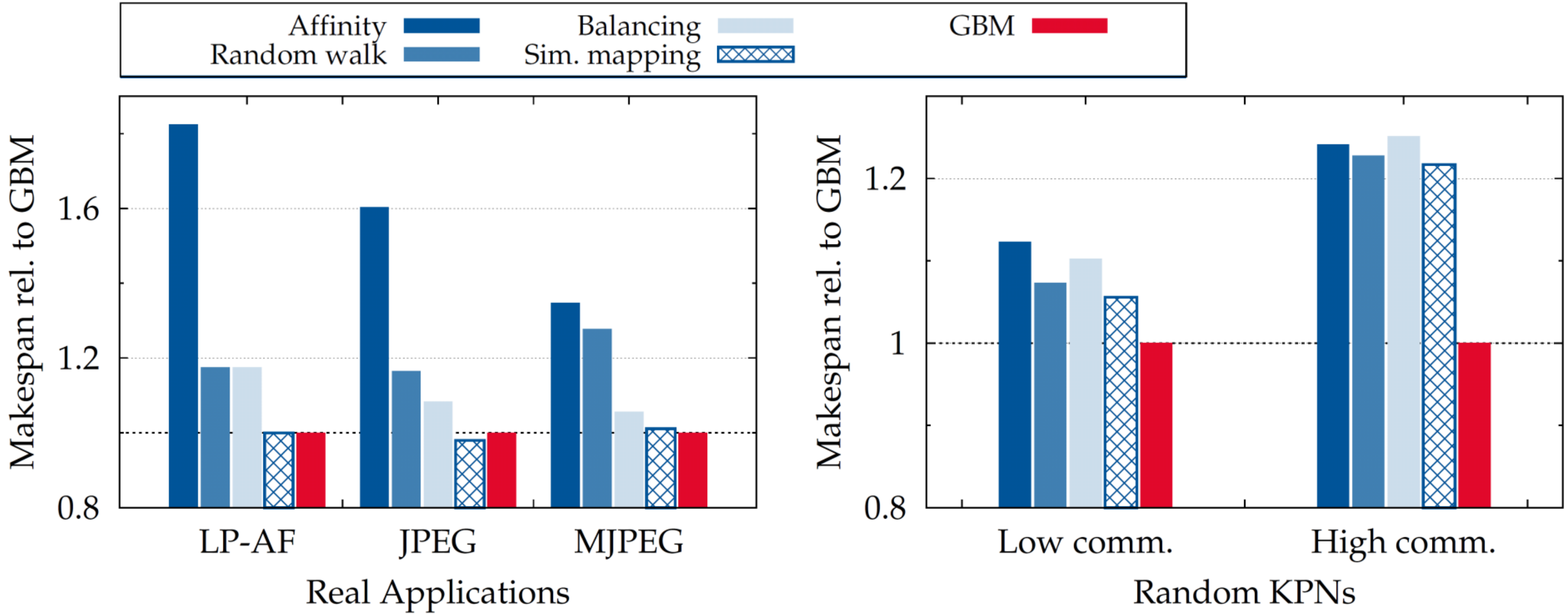


```
-<Core Name="DSPC66" CoreType="DSPC66" Category="DSP">
  -<MultiTaskingInfo MaxNumberOfTasks="-1">
     <ContextSwitchInfo StoreTime="1000" LoadTime="1000"/>
     <SchedulingPolicies List="FIFO PriorityBased"/>
   </MultiTaskingInfo>
  -<CostTable>
    -<Operation Name="Load">
      -<VariableType Name="Char">
         <Cost>1</Cost>
       </VariableType>
      -<VariableType Name="Double">
```

[SOC13]

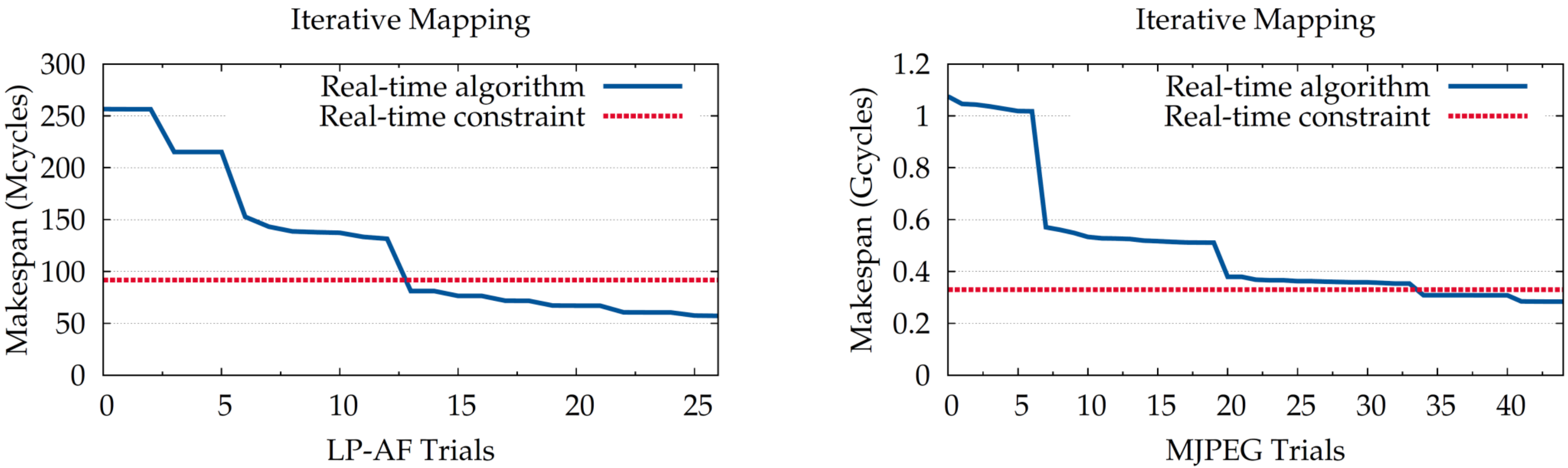
Example: multi-media applications

- Platform: 2 RISCs, 4 VLIW, 7 Memories



Example: multi-media applications (2)

- Dealing with real-time constraints

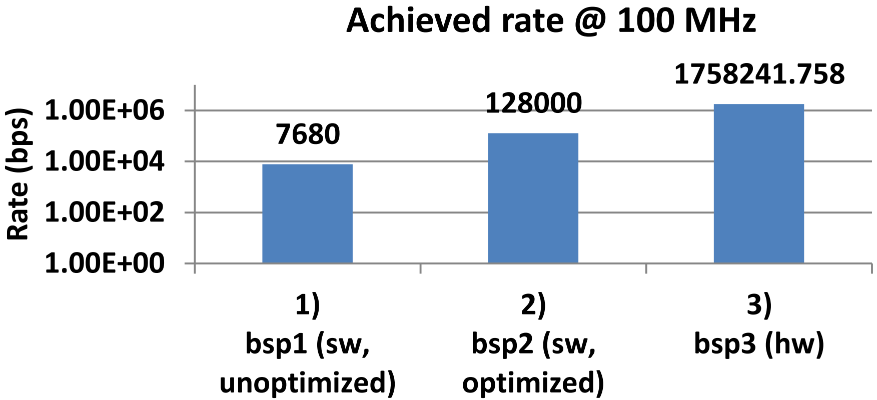


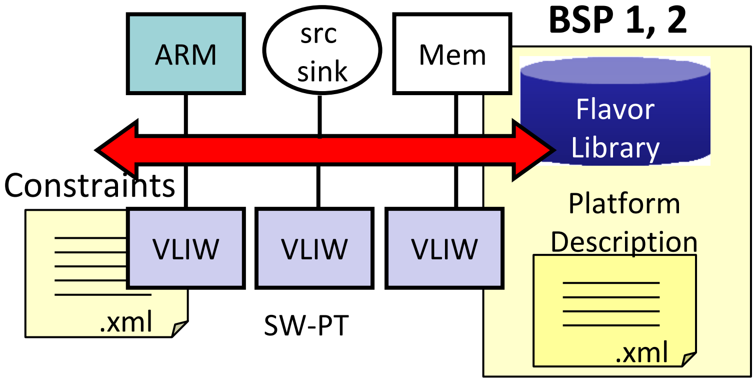
Tool: ~1 min. for LP-AF, ~7 min. for MJPEG ~10 min.

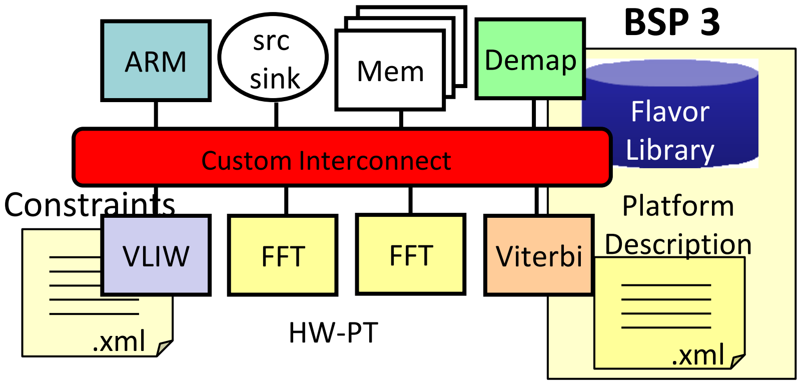
Sim.: ~6 days for LP-AF, ~24 for MJPEG ~10 days (~x10^3)

SDR results: portable performance

- ❑ Application: MIMO OFDM reeiver
- ❑ Hardware
  - ❑ Platform 1: Baseline software
  - ❑ Platform 2: Optimized software
  - ❑ Platform 3: Optimized SW + HW







Summary

- Programming methodology for dataflow applications
  - Programming model adapted to application domain
- Energy-efficiency
  - Aware of **heterogeneous** platforms with hardware acceleration
  - Do not use more resources than needed
- Outlook – Adaptability within HAEC CRC
  - Adapt HW to SW needs (e.g., wireless on-chip interconnect)
  - Adapt parallelism (with implicit parallel constructs) for scalability
  - More abstract application-specific software synthesis

References

[**ITRS11**] International Technology Roadmap for Semiconductors (ITRS), “System Drivers,” 2011. [Online]. Available: http://www.itrs.net/

[**Castrillon14**] J. Castrillon and R. Leupers, Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap. Springer, 2014

[**EETimes11**] http://www.eetimes.com/document.asp?doc_id=1259446

[**PARCO14**] W. Sheng, S. Schürmans, M. Odendahl, M. Bertsch, V. Volevach, R. Leupers, and G. Ascheid, “A compiler infrastructure for embedded heterogeneous MPSoCs”, Parallel Comput. 40, 2 (February 2014), 51-68

[**DATE10**] J. Castrillon, R. Velasquez, A. Stulova, W. Sheng, J. Ceng, R. Leupers, G. Ascheid, H. Meyr, “Trace-based KPN composability analysis for mapping simultaneous applications to MPSoC platforms”, Proceedings of the Conference on Design, Automation and Test in Europe, pp. 753-758, 2010

[**IEEE-TII13**] J. Castrillon, R. Leupers, and G. Ascheid, “MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs,” IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 527–545, 2013

[**DAC12**] J. Castrillon, A. Tretter, R. Leupers, and G. Ascheid, “Communication-aware mapping of KPN applications onto heterogeneous MPSoCs,” in Proceedings of the Design Automation Conference, 2012

[**SDR10**] J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, R. Leupers, G. Ascheid, and H. Meyr, “Component-based waveform development: The nucleus tool flow for efficient and portable SDR,” Wireless Innovation Conference and Product Exposition (SDR), 2010

References (2)

[**ALOG11**] J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, R. Leupers, G. Ascheid, and H. Meyr, “Component-based waveform development: The nucleus tool flow for efficient and portable software defined radio”, Analog Integrated Circuits and Signal Processing, vol. 69, no. 2–3, pp. 173–190, 2011

[**MILCOM09**] V. Ramakrishnan, E. M. Witte, T. Kempf, D. Kammler, G. Ascheid, H. Meyr, M. Adrat and M. Antweiler, “Efficient and Portable SDR Waveform Development: The Nucleus Concept”, in *IEEE Military Communications Conference (MILCOM)*, 2009

[**SOC13**] M. Odendahl, J. Castrillon, V. Volevach, R. Leupers and G. Ascheid, “Split-cost communication model for improved MPSoC application mapping”, In International Symposium on System on Chip (SoC) pp. 1-8, 2013

Thanks!

Questions?