

Benchmark Applications

Todor Stefanov and Ed Deprettere

Leiden Embedded Research Center,
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands



Universiteit Leiden

Agenda

- Introduction to the current benchmark suite
- Brief reporting on the experiences with the benchmark applications
 - Iuliana Bacivarov (10 min)
 - Todor Stefanov (10 min)
 - Bastian Ristau (10 min)
- Discussion

Benchmark Suite

- Common set of benchmark applications is missing in the MPSoC community
 - The issue was discussed at the Map2MPSoC working meeting on November 27-28, 2008 in Dusseldorf
 - Many applications are available BUT
 - They are in sequential form
 - The MPSoC community needs parallel application specifications
- Decision made at the meeting
 - To initiate the creation of an application benchmark suit for MPSoC
 - To create a common repository where interested parties can upload applications for the benchmark suite
 - To make the benchmark suite available on Internet

Current Content of Benchmark Suite

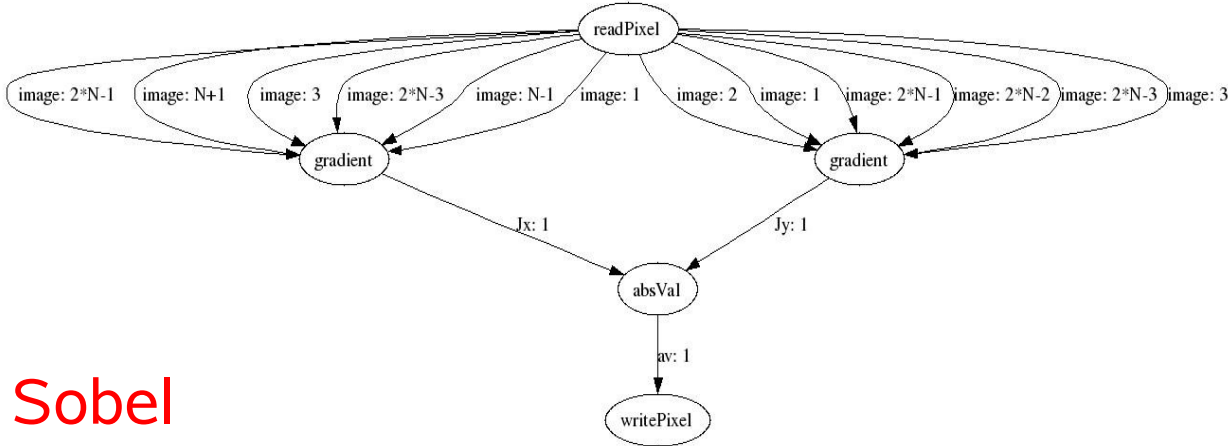
- The benchmark suit is available on Internet at <http://www.artist-embedded.org/artist/Benchmarks.html>
- It consists of 8 applications contributed by 3 universities
 - U Leiden (Todor Stefanov)
 - Motion JPEG encoder
 - JPEG 2000 encoder
 - Sobel
 - TU Eindhoven (Marc Geilen)
 - MP3
 - H.263 encoder
 - H.263 decoder
 - ETH Zurich (Iuliana Bacivarov)
 - MPEG-2 decoder
 - MJPEG decoder

Features of the applications

Application	Sequential Spec		Parallel Spec			License
	format	fully func.	format	fully func.	MoC	
Motion JPEG encoder	C	Yes	C++ for YAPI XML for Daedalus	Yes Yes	KPN KPN	CPL CPL
JPEG 2000 encoder	C	Yes	C++ YAPI XML for Daedalus	Yes Yes	KPN KPN	CPL CPL
Sobel	C	Yes	C++ YAPI XML for Daedalus	Yes Yes	KPN KPN	CPL CPL
MP3	C	Yes	XML for SDF3	No	SDF	GPL
H.263 encoder	C	Yes	XML for SDF3	No	SDF	GPL
H.263 decoder	C	Yes	XML for SDF3	No	SDF	GPL
MPEG 2 decoder	C	Yes	XML and C for DOL	Yes	KPN	ETHZ
MJPEG decoder	C	Yes	XML and C for DOL	Yes	KPN	ETHZ

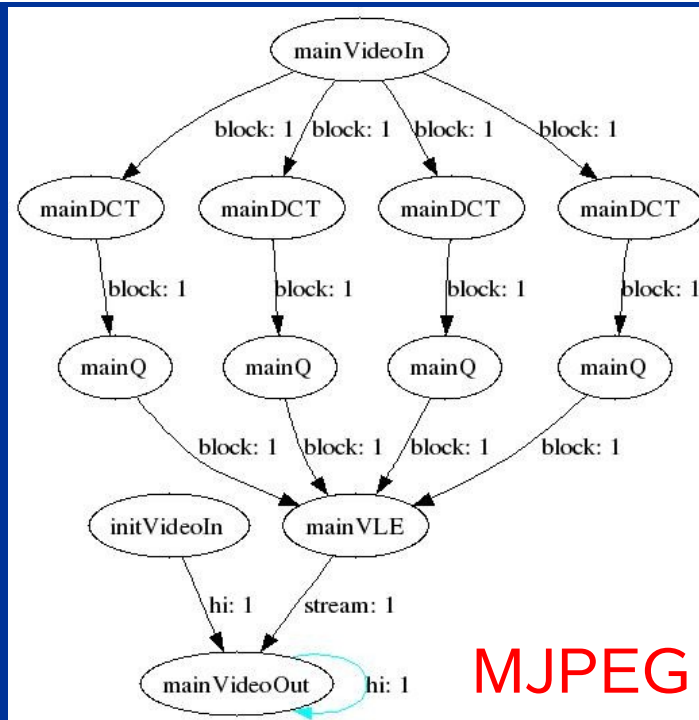
- Some observations:
 - All applications are streaming (data-flow) applications
 - All parallel specs use well known data-flow models of computation (MoC)
 - All parallel specs are in different XML formats
 - Not all parallel specs are fully functional
 - ...

Applications contributed by U Leiden

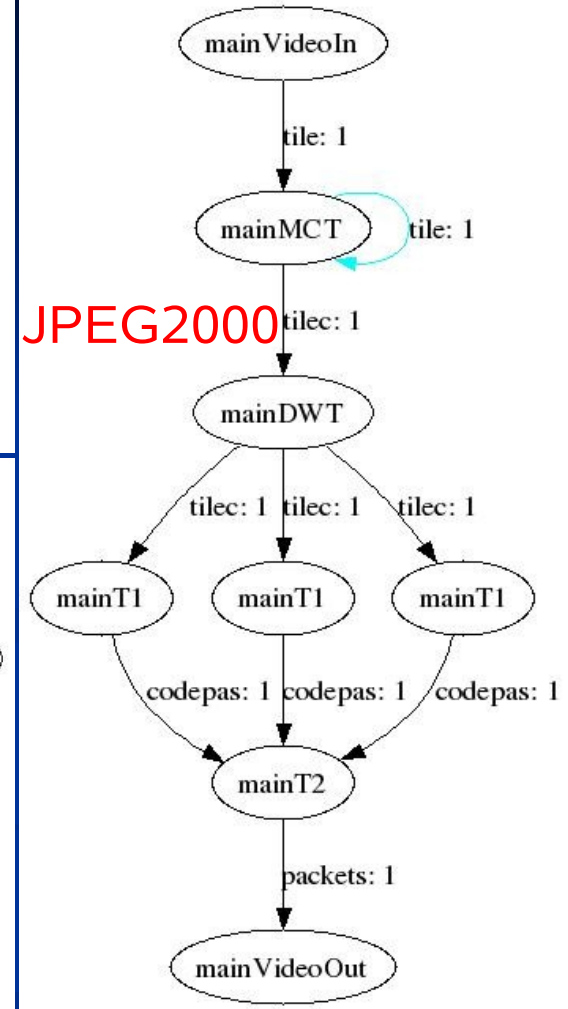


Sobel

- Sobel (edge detection)
 - only task-level parallelism
- JPEG2000 (encoder)
 - mainly task-level parallelism
- Motion JPEG (encoder)
 - task-level and data-level parallelism



MJPEG

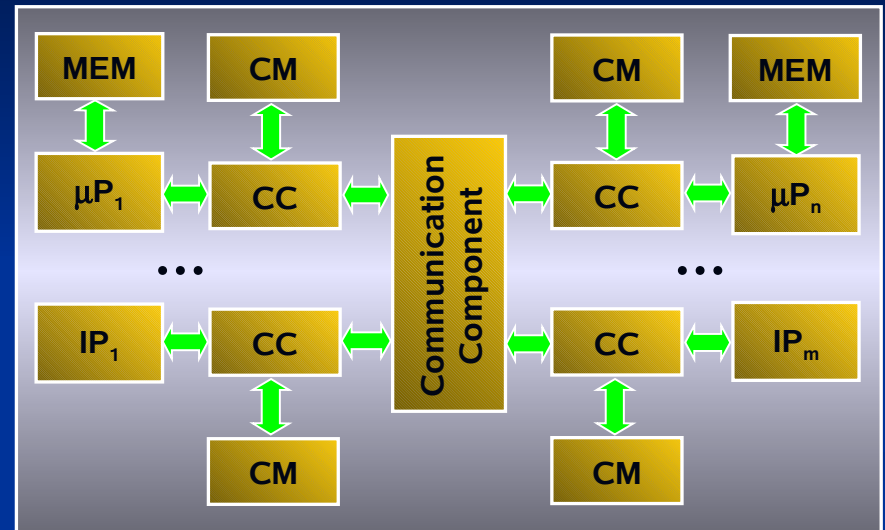


JPEG2000

MP-SoC platform considered

Platform \Leftrightarrow library of components + well defined rules how to connect and synchronize

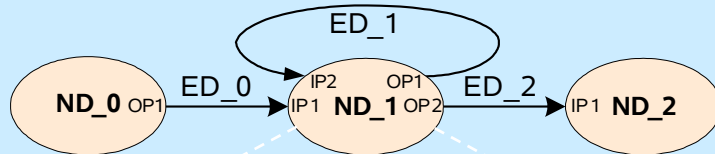
- Processing Components:
 - Programmable processors
 - Hardware IP Cores
- Memory Components:
 - Program, Data (on-chip and external) Memory (MEM)
 - Communication Memory (CM)
- Communication Components:
 - Point-to-point network
 - Crossbar switch
 - Shared bus with Round-Robin, Fixed Priority, or TDMA arbitration
- Communication Controller (CC) – interface between processing, memory, and communication components



Many alternative platform instances can be constructed fast and easily by instantiating different type/number of components and setting their parameters.

Parallel Specification in C++YAPI format

- YAPI simulation environment is available at <http://sourceforge.net/projects/y-api/>



```

1 // process ND_1
2 void main( ) {
3   for( int i=2; i<=M; i++ )
4     for( int j=2; j<=N; j++ ) {          CONTROL
5       if( j-2 == 0 )
6         read( IP1, in_0 );
7       if( j-3 >= 0 )
8         read( IP2, in_0 );              READ
9       Transformer( in_0, out_0 );      EXECUTE
10      if( -j+N-1 >= 0 )
11        write( OP1, out_0 );
12      if( j-N == 0 ) {
13        write( OP2, out_0 );
14      } // for j
15 } // main
  
```

```

#ifndef simple_KPN_H
#define simple_KPN_H

#include "yapi.h"

#include "ND_0.h"
#include "ND_1.h"
#include "ND_2.h"

class simple : public ProcessNetwork {
private:
    // Channels
    Fifo<tCH_1> ED_0;
    Fifo<tCH_2> ED_1;
    Fifo<tCH_3> ED_2;

    // processes
    ND_0 ND_0_instance;
    ND_1 ND_1_instance;
    ND_2 ND_2_instance;

public:
    simple(
        Id n, int parm_N, int parm_M
    ) :
        ProcessNetwork(n),
        ED_0(id("ED_0"), 1, 1),
        ED_1(id("ED_1"), 1, 1),
        ED_2(id("ED_2"), 1, 1),
        ND_0_instance(id("ND_0"), ED_0, parm_N, parm_M),
        ND_1_instance(id("ND_1"), ED_0, ED_1, ED_1, ED_2, parm_N, parm_M),
        ND_2_instance(id("ND_2"), ED_2, parm_N, parm_M)
    {
    };

    const char* type() const { return "simple"; };
};

#endif /* simple_H */
  
```


Parallel Spec in XML Daedalus format

- Daedalus design framework is available at <http://daedalus.liacs.nl>

Network Topology Specification in XML

```
<sadg>
  <adg name="simple" levelUpNode="">
    <parameter name="N" lb="450" ub="1000" value="450"/>
    <parameter name="M" lb="275" ub="1000" value="275"/>

    <node name="ND_0" levelUpNode="">
      <outport name="OP1" node="ND_0" edge="ED_0">
      </outport>
    </node>

    <node name="ND_1" levelUpNode="">
      <inport name="IP1" node="ND_1" edge="ED_0">
      </inport>
      <inport name="IP2" node="ND_1" edge="ED_1">
      </inport>
      <outport name="OP1" node="ND_1" edge="ED_1">
      </outport>
      <outport name="OP2" node="ND_1" edge="ED_2">
      </outport>
    </node>

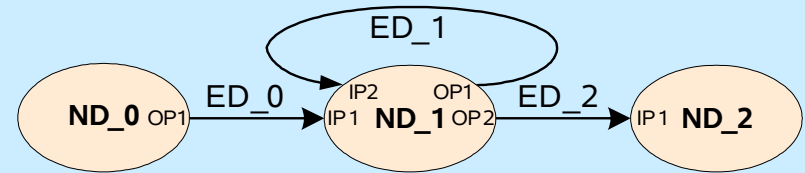
    <node name="ND_2" levelUpNode="">
      <inport name="IP1" node="ND_2" edge="ED_2">
      </inport>
    </node>

    <edge name="ED_0" fromPort="OP1" fromNode="ND_0" toPort="IP1" toNode="ND_1" size="1">
    </edge>

    <edge name="ED_1" fromPort="OP1" fromNode="ND_1" toPort="IP2" toNode="ND_1" size="1">
    </edge>

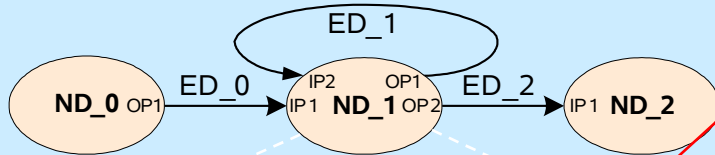
    <edge name="ED_2" fromPort="OP2" fromNode="ND_1" toPort="IP1" toNode="ND_2" size="1">
    </edge>

  </adg>
</sadg>
```



Parallel Spec in XML Daedalus format

Process Control Code Specification in XML (1)



$$\begin{matrix} 2 \leq i \leq M, \\ 2 \leq j \leq N, \\ j - 2 = 0 \end{matrix}$$

$$\begin{matrix} 2 \leq i \leq M, \\ j - 2 = 0 \end{matrix}$$

$$\begin{matrix} i - 2 & 0, \\ -i + M & 0, \\ j - 2 = 0 \end{matrix}$$

```

1 // process ND_1
2 void main( ) {
3   for( int i=2; i<=M; i++ )
4     for( int j=2; j<=N; j++ ){
5       if( j-2 == 0 )
6         read( IP1, in_0 );
7       if( j-3 >= 0 )
8         read( IP2, in_0 );
9       Transformer( in_0, out_0 );
10      if( -j+N-1 >= 0 )
11        write( OP1, out_0 );
12      if( j-N == 0 ) {
13        write( OP2, out_0 );
14      } // for j
15 } // main
  
```

$$\begin{matrix} 1*i + 0*j + 0*N + 0*M - 2 & 0, \\ -1*i + 0*j + 0*N + 1*M + 0 & 0, \\ 0*i + 1*j + 0*N + 0*M - 2 & = 0 \end{matrix}$$

sign	i	j	N	M	const
1	1	0	0	0	-2
1	-1	0	0	1	0
0	0	1	0	0	-2

Parallel Spec in XML Daedalus format

Process Control Code Specification in XML (2)

```
<sadg>
<adg name="simple" levelUpNode="">
  <parameter name="N" lb="450" ub="1000" value="450"/>
  <parameter name="M" lb="275" ub="1000" value="275"/>

  <node name="ND_1" levelUpNode="">

    <import name="IP1" node="ND_1" edge="ED_0">
      <bindvariable name="in_0" dataType="int"/>
      <domain type="LBS">
        <linearbound index="i, j" staticControl="" dynamicControl="" parameter="N, M">
          <constraint matrix="[0, 0, 1, 0, 0, -2;
                               1, 1, 0, 0, 0, -2;
                               1, -1, 0, 0, 1, 0]"/>


        </linearbound>
      </domain>
    </import>

    <import name="IP2" node="ND_1" edge="ED_1">
    </import>

    <outputport name="OP2" node="ND_1" edge="ED_1">
    </outputport>
    <outputport name="OP2" node="ND_1" edge="ED_2">
    </outputport>

    <function name="Transformer">
      <inargument name="in_0" dataType="int"/>
      <outargument name="out_0" dataType="int"/>
    </function>
    <domain type="LBS">
      <linearbound index="i, j" staticControl="" dynamicControl="" parameter="N, M">
        <constraint matrix="[1, 1, 0, 0, 0, -2;
                             1, -1, 0, 0, 1, 0;
                             1, 0, 1, 0, 0, -2;
                             1, 0, -1, 1, 0, 0]"/>

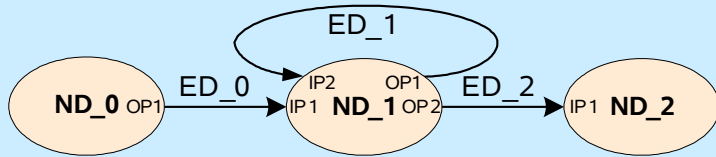
      </linearbound>
    </domain>
  </node>
</adg>
</sadg>
```



<i>sign</i>	<i>i</i>	<i>j</i>	<i>N</i>	<i>M</i>	<i>const</i>
1	1	0	0	0	-2
1	-1	0	0	1	0
0	0	1	0	0	-2

Parallel Spec in XML Daedalus format

Schedule Specification in XML



```
<?xml version="1.0"?>
<sadg>

  <adg name="simple" levelUpNode="">
    ...
  </adg>

  <ast>

    <for iterator="c0" LB="2" UB="1*M+1" stride="1">
      <for iterator="c1" LB="2" UB="1*N+1" stride="1">

        <stmt node="ND_0"/>

        <if LHS="1*c0" RHS="1*M" sign="-1">
          <if LHS="1*c1" RHS="1*N" sign="-1">
            <stmt node="ND_1"/>
          </if>

          <if LHS="1*c1" RHS="1*N" sign="0">
            <stmt node="ND_2"/>
          </if>

        </if>

      </for>
    </for>

  </ast>

</sadg>
```

- Schedule gives a deadlock free execution order of the processes
 - Either with the absolute minimum FIFO buffer sizes that guaranty deadlock free execution
 - Or with the minimum FIFO buffer sizes that guarantee maximum performance
- Schedule represented as an abstract syntax tree in XML -- <ast> tag
- The <ast> can be converted to a control program implementing the schedule

Discussion (1)

- Benchmark application suite
 - Do we need more applications?
 - Currently, we have 8 applications
 - Do we need more diverse applications?
 - Currently, we have only streaming (data-flow) applications where the parallel specs use the KPN or SDF MoC
 - What about control oriented applications?
 - What about more dynamic/adaptive applications?
 - What about applications in other MoCs?
 - Do we accept in the benchmark suit non fully functional application specs and non available simulation/execution engine for the parallel specs?
 - Do we need a common format for the applications?
 - Currently we have different XML and C/C++ formats
 - How difficult is to port an application to your specific format?

Discussion (2)

- How to use the benchmark application suite to
 - Understand better the design flows and tools by others
 - Port the benchmark suite to your tool specific format
 - Make available your tools with the ported benchmark suit to others
 - Present experimental results
 - Make available your tools and experimental setup such that others can reproduce the experimental results
 - Compare qualitatively design flows and tools
 - Use the suite, map it onto your MPSoC and present the best results you can get
 - The results will depend on the MPSoC and the quality of the tools
 - We define/agree on a common platform and map the suite onto this platform
 - The result will depend on the quality of the tools

Thank you

All benchmark applications can be found at:

<http://www.artist-embedded.org/artist/Benchmarks.html>