Compaan: Deriving Process Networks from Matlab for Embedded Signal Processing Architectures

Bart Kienhuis¹,
Edwin Rypkema², and Ed Deprettere²
1: UC Berkeley, USA
2: Leiden University, The Netherlands

by Bart Kienhuis

At the 8th international Workshop on Hardware/Software Codesign.
San Diego, May 3-5, 2000
Problem

Application

for k = 1:1:K,
    for j = 1:1:N,
        \[ r(j,j, x(k,j), t) = \text{Vec}(r(j,j), x(k,j)) ; \]
        for i = j+1:1:N,
            \[ r(j,i, x(k,i), t) = \text{Rot}(r(j,i), x(k,i), t) ; \]
        end
    end
end

Mapping?

Programmable Interconnect Network

MatParser

Polyhedra Reduced Dependence Graph

DGPParser

Panda

Architecture

Micro Processor

Memory

PE0 PE1 PE2 \cdots PE\text{n}

Process Network
Step 1: MatParser

Array Dataflow Analysis

for \( k = 1 : 1 : K \),
for \( j = 1 : 1 : N \),
\[
[r(j,j), x(k,j), t] = \text{Vec}(r(j,j), x(k,j));
\]
for \( i = j + 1 : 1 : N \),
\[
[r(j,i), x(k,i), t] = \text{Rot}(r(j,i), x(k,i), t);
\]
end
end

for \( k = 1 : 1 : K \),
for \( j = 1 : 1 : N \),
if \( k - 2 \geq 0 \),
\[
[\text{in}_0] = \text{ipd}(r_2(k-1, j));
\]
else
\[
[\text{in}_0] = \text{ipd}(r_1(j, j));
\]
end

\[
[\text{out}_0, \text{out}_1, \text{out}_2] = \text{Vec}(\text{in}_0, \text{in}_1);
\]
\[
[r_1(k, j)] = \text{opd}(\text{out}_0);
\]
\[
[x_1(k, j)] = \text{opd}(\text{out}_1);
\]
\[
[t_1(k, j)] = \text{opd}(\text{out}_2);
\]
for \( i = j + 1 : 1 : N \),

Bart Kienhuis (Codes2000)
for k = 1:1:K,
    for j = 1:1:N,
        \[ r(j, j), x(k, j), t \] = Vec( r(j, j), x(k, j) );
        for i = j + 1:1:N,
            \[ r(j, i), x(k, i), t \] = Rot( r(j, i), x(k, i), t );
        end
    end
end
Step 2: DgParser

Polyhedral Reduced Dependence Graph

Bart Kienhuis (Codes2000)
Example of Node and Port Domains
SBF Object and Network Generation

SBF Network (Parallel)

SBF Object (Sequential)
Elements of an SBF Object

- Function Repertoire $F = \{ f_1, f_2, \ldots, f_{|F|} \}$
- Binding Function $\mu: C \rightarrow F$
- Transition Function $\omega: C \rightarrow C$

Bart Kienhuis (Codes2000)
Step 3: PANDA

- Domain Reconstruction
- Domain Scanning
- Linearization
/** fire the actor. */
public void fire() throws IllegalActionException {

    for ( int k = 1; k <= 1*K; k += 1 ) {
        for ( int j = 1; j <= 1*N; j += 1 ) {

            if ( k - 2 >= 0 ) { in_0 = RP_1.get(0); }  
            if ( k - 1 == 0 ) { in_0 = RP_2.get(0); }  
            if ( j - 2 >= 0 ) { in_1 = RP_3.get(0); }  
            if ( j - 1 == 0 ) { in_1 = RP_4.get(0); }  

            // Execute the function
            [out_0, out_1, out_2] = F.Vectorize(in_0, in_1);

            if ( K - k - 1 >= 0 ) { WP_1.send( out_0 ); }  
            if ( -K + k == 0 ) { WP_11.send( out_0 ); }  
            if ( N - j - 1 >= 0 ) { WP_10.send( out_2 ); }  

        }
    }
}
Ptolemy II, simulation
Workload Analysis

- Separation Communication/Computation
- Higher levels of granularity

Bart Kienhuis (Codes2000)
Possible Architecture

MicroProcessor/Coprocessor Bus

Bart Kienhuis (Codes2000)
Y-chart

• Workload Analysis
• Early integration
• High-level abstraction
Conclusions

- Described the Compaan Tool:
  - Polyhedral Reduced Dependence Graph
    - Good Mathematical model
  - SBF Model of Computation
    - Possible candidates for coprocessors
- Implemented a 3-step approach
  - MatParser/DgParser/Panda
- Why did we do it?
  - Process Networks are better match
    - Expresses parallelism (fine grained/coarse grained)
    - Easier mapping

Http://gigascale.org/compaan