Research Status on the Methodology of System Design

Jinian Bian, Hongxi Xue, Ming Zhu, Qiang Wu, Yunfeng Wang, Zhengsheng Xu, Jianzhou Zhao, Heng Hu

DA Lab., Dept. of Computer, Tsinghua University, Beijing, China
Contents

- System Level Design Flow
- System Specification
  - C2VHDL, Extension to HDM & CDFG
- System Partitioning
  - Applying the Search Space Smoothing Technique to Hardware/Software Partitioning
- System Synthesis
  - HSM, Delay-driven Post-layout Re-synthesis
- Future Work
System Level Design Flow

- C
- VHDL
- C2VHDL
- HDM(IIR)
- HCDFG
- Hardware/Software Partitioning
- High Level Synthesis
- Software Compiler
- System Integration
System Specification

- **C2VHDL**
  - Packages
    - FPU32, FPU64, CtoVHDL (synthesizable)
  - Experiments
    - FFT, FWT, FIR, Lowpass
- We run C codes with some input patterns, and run the VHDL simulator with the corresponding waveforms, the output of C and VHDL codes are identical.
struct ref_func
{
    double element[512];
};
struct ref_func main()
{
    struct ref_func triang(int,struct ref_func,int,int);
    struct ref_func hamming(int,struct ref_func,int,int,double,double);
    struct ref_func kaiser(int,n,struct ref_func w,int n,int ied,double beta);
    double g[512],pi,twopi,c1,xn,c3,beta;
    int i,ied,n,il;
    struct ref_func w;
    double rmin,rmax;
    int j,k,l,index,imd;
    int nf,itype,jtype;
    double fc,fl,fh,alpha,att,powofatt;
    nf=25;

    USE work.FPU64.all;
    PACKAGE maintypedefine IS
        TYPE zjarray0 IS ARRAY (0 TO 512-1) OF ZREAL;
        TYPE ref_func IS RECORD
            element : zjarray0;
        END RECORD;
    END maintypedefine;
    USE maintypedefine.all;
    USE work.CtoVHDL.all;
    USE work.FPU64.all;
    LIBRARY IEEE;
    USE ieee.std_logic_1164.all;
    USE ieee.std_logic_arith.all;
    ENTITY zjzzjz IS
        port {
            zjzzjzout : OUT ref_func
        };
    END zjzzjz;
    ARCHITECTURE zjzzjz OF zjzzjz IS
    BEGIN
System Specification

- Hierarchical CDFG
  - Transport Node: Modeling the communication resources
  - Hierarchy: A node can recursively contain a CDFG
**System Specification**

- Hierarchical CDFG
- Example

```c
void swap(int & x, int & y) {
    int t;
    t = x;
    x = y;
    y = t;
}
void main() {
    int a[10];
    for(int i=0; i<5; i++)
        if(a[i]>a[9-i]) swap(a[i], a[9-i]);
    }
```
System Partitioning

- **Partitioning Model**
  - Partitioning Problem
    - A constrained optimization problem
  - **Target Architecture**
    - One CPU+RAM+BUS, several ASIC blocks

- **System Function Representation**
  - **Process Graph**
    - Coarse granularity: each node represents a process
    - Directed Acyclic graph
    - Weights of nodes and edges denote computation load, area, communication density and delay…
System Partitioning

- **Object Function & Constraints**

  - **Object Function**

    \[
    C(H_W, S_W) = Q1 \times \sum_{(ij) \in \mathcal{M}} W_{ij}^E + Q2 \times \frac{\sum_{i \in H_W} W_i^N}{N_H} - Q3 \times \left( \frac{\sum_{i \in H_W} W_{2i}^N}{N_H} + \frac{\sum_{i \in S_W} W_{2i}^N}{N_S} \right)
    \]

  - **Constraints**

    \[
    \sum_{(i) \in H_W} H_{-cost_i} \leq Max^H;
    \sum_{(i) \in S_W} S_{-cost_i} \leq Max^S
    \]

  The object function guide the partitioning procedure to select those nodes that have more computation load, less communication for Hardware implementation.
After smoothing, the number of local extremal points is decreased. So the search procedure leads to global optimum easier.
Partitioning Algorithm: Search Space Smoothing (2)

Serial Search Space Smoothing: use a series of smoothed Search Space to guide the search procedure heading to global optimum.
Partitioning Algorithm: Search Space Smoothing (3)

- **Smoothing Method**
  \[ w_i(\alpha) = \begin{cases} 
  w + (w_i - \bar{w})^\alpha & w_i \geq \bar{w} \\
  w - (w - w_i)^\alpha & w_i < \bar{w} 
  \end{cases} \]

- **Local Search Method**
  - Simulated Annealing, simple move (SM)

- **Experiments**
  - Tabu Search, Search Space Smoothing algorithms & two types of Simulated Annealing are implemented for comparison
  - PG instances of 10, 20, 40, 100, 200, 300, 400, 500 nodes are generated & tested
Solution Quality Analysis

TS found the best solution in all runs. SSS found the best solution in half of the runs. Two types of SA failed to find the best solution at all.

SSS technique greatly improved the performance of the SA (SM).
Run Time Analysis (1)

TS climbs far more quickly than SSS. SAs runs faster than the former two.
SSS outperforms TS in cases of large PGs with a node number greater than 200.
Conclusion of Experiments

- SSS can find best solution in a higher probability than SA, while TS is more stable.
- SSS runs faster than TS in cases of large node num. (>=200)

<table>
<thead>
<tr>
<th>Node Num.</th>
<th>SA</th>
<th>TS</th>
<th>SSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>small(&lt;40)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>middle(40-400)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>large(&gt;400)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Note: The first column represents solution quality, the second column run time.
System Synthesis

- Interconnect-driven High Level Synthesis
  - Data-path synthesis
    - Use grid to represent resource allocation and scheduling of operations
    - Use CBL to represent the layout of blocks
    - Use SA and SSS as the optimization algorithm
  - Control synthesis
    - Hierarchical State Minimization (HSM)
## Experimental Results for Hierarchical State Minimization (HSM)

<table>
<thead>
<tr>
<th>FSMs</th>
<th>In/Out</th>
<th>Initial state</th>
<th>STAMINA</th>
<th>VOID</th>
<th>HSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cover</td>
<td>Time (s)</td>
<td>Cover</td>
</tr>
<tr>
<td>Bbtas_bbara</td>
<td>6/4</td>
<td>60</td>
<td>42</td>
<td>0.68</td>
<td>42</td>
</tr>
<tr>
<td>Bbtas'beecount</td>
<td>5/6</td>
<td>42</td>
<td>24</td>
<td>0.25</td>
<td>24</td>
</tr>
<tr>
<td>Bbtas's8</td>
<td>6/3</td>
<td>30</td>
<td>6</td>
<td>0.14</td>
<td>6</td>
</tr>
<tr>
<td>Beecount'bbara</td>
<td>7/6</td>
<td>70</td>
<td>28</td>
<td>63.75</td>
<td>28</td>
</tr>
<tr>
<td>Beecount'dk14</td>
<td>6/9</td>
<td>49</td>
<td>28</td>
<td>0.41</td>
<td>28</td>
</tr>
<tr>
<td>Beecount'dk17</td>
<td>5/7</td>
<td>56</td>
<td>32</td>
<td>0.2</td>
<td>32</td>
</tr>
<tr>
<td>Beecount's27</td>
<td>7/5</td>
<td>42</td>
<td>20</td>
<td>0.38</td>
<td>20</td>
</tr>
<tr>
<td>Dk14's8</td>
<td>7/6</td>
<td>35</td>
<td>7</td>
<td>0.21</td>
<td>7</td>
</tr>
<tr>
<td>Dk15'beecount</td>
<td>6/9</td>
<td>28</td>
<td>16</td>
<td>0.19</td>
<td>16</td>
</tr>
<tr>
<td>Dk17's8</td>
<td>6/4</td>
<td>40</td>
<td>8</td>
<td>0.09</td>
<td>8</td>
</tr>
<tr>
<td>Dk27's27</td>
<td>5/3</td>
<td>42</td>
<td>35</td>
<td>0.04</td>
<td>35</td>
</tr>
<tr>
<td>Dk27's8</td>
<td>5/3</td>
<td>35</td>
<td>7</td>
<td>0.06</td>
<td>7</td>
</tr>
<tr>
<td>Lion's8</td>
<td>6/2</td>
<td>20</td>
<td>4</td>
<td>0.03</td>
<td>4</td>
</tr>
<tr>
<td>Train4's8</td>
<td>6/2</td>
<td>20</td>
<td>4</td>
<td>0.06</td>
<td>4</td>
</tr>
<tr>
<td>S298</td>
<td>3/6</td>
<td>218</td>
<td>-</td>
<td>-</td>
<td>135</td>
</tr>
<tr>
<td>S1494</td>
<td>8/19</td>
<td>48</td>
<td>48</td>
<td>-</td>
<td>48</td>
</tr>
<tr>
<td>Tbk</td>
<td>6/3</td>
<td>32</td>
<td>16</td>
<td>-</td>
<td>16</td>
</tr>
<tr>
<td>Comp. with STAMINA</td>
<td></td>
<td></td>
<td>100%</td>
<td></td>
<td>84.5%</td>
</tr>
</tbody>
</table>
System Synthesis

- Delay-driven post-layout re-synthesis
- GBAW: Graph Based Alternative Wiring
- ALEB: Alternative Logic Equivalent Block

Example of ALEB
## Experimental Results for Re-synthesis

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Original circuit</th>
<th>Re-synthesized Based GBAW</th>
<th>Re-synthesized Based ALEB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gates number</td>
<td>Max delay</td>
<td>Gates number</td>
</tr>
<tr>
<td>C3540</td>
<td>1026</td>
<td>7.035</td>
<td>1079</td>
</tr>
<tr>
<td>C5315</td>
<td>1616</td>
<td>12.136</td>
<td>1640</td>
</tr>
<tr>
<td>C6288</td>
<td>2398</td>
<td>16.170</td>
<td>2495</td>
</tr>
<tr>
<td>C7552</td>
<td>2512</td>
<td>34.853</td>
<td>2527</td>
</tr>
<tr>
<td>Duke2</td>
<td>411</td>
<td>1.597</td>
<td>-</td>
</tr>
<tr>
<td>Dalu</td>
<td>1257</td>
<td>4.052</td>
<td>-</td>
</tr>
<tr>
<td>Alu2</td>
<td>325</td>
<td>4.134</td>
<td>352</td>
</tr>
<tr>
<td>Alu4</td>
<td>655</td>
<td>5.597</td>
<td>705</td>
</tr>
<tr>
<td>Apex6</td>
<td>676</td>
<td>1.766</td>
<td>687</td>
</tr>
<tr>
<td>Misex3</td>
<td>636</td>
<td>10.924</td>
<td>636</td>
</tr>
<tr>
<td>Adder</td>
<td>206</td>
<td>3.132</td>
<td>223</td>
</tr>
<tr>
<td>Rot</td>
<td>546</td>
<td>3.382</td>
<td>570</td>
</tr>
<tr>
<td>Sao2</td>
<td>285</td>
<td>5.030</td>
<td>332</td>
</tr>
<tr>
<td>Term1</td>
<td>338</td>
<td>2.408</td>
<td>355</td>
</tr>
<tr>
<td>Ttt2</td>
<td>189</td>
<td>1.398</td>
<td>202</td>
</tr>
<tr>
<td>X3</td>
<td>648</td>
<td>7.670</td>
<td>648</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td>100%</td>
<td>105.0%</td>
</tr>
</tbody>
</table>
Future Work

- Implement the simulation and evaluation tools for HCDFG
- Find suitable local search method to cooperate with SSS
- Extend the system partitioning algorithm to deal with multi-way partitioning
- Continue developing HLS tool and software compiler to synthesize hardware and software parts
Thank You!

Email:
wuqiang2000@mails.tsinghua.edu.cn