A Dynamic-Programming Algorithm for Reducing the Energy Consumption of Pipelined System-Level Streaming Applications

N. Liveris, H. Zhou, P. Banerjee*
Northwestern University, Evanston IL, USA
*HP Labs, Palo Alto Ca, USA

ASP-DAC 2008, Seoul, Korea
January 22, 2007
Outline

- Motivation
- Technique
- Theoretical Exploration
- Dynamic Programming Solution
- Experimental Results
- Conclusions
Leakage power consumption is expected to become dominant for future technologies.

Techniques for reducing power consumption are needed that:
- are adaptive to environment changes
- do not require resynthesis of IP cores
Power Gating

- Sleep transistor creates “virtual Vdd” when logic block is idle (sleep = 1)
- Stand-by energy consumption is decreasing
  - $E = T \cdot I_L(Vdd) \cdot Vdd$
Power Gating

- With power gating a module is shut down when it is idle.
- Energy penalty mainly for switching from sleep to active – loading of the nodes back to normal Vdd levels.
- In this work we try to increase the energy savings by reducing the number of switches.
Synchronous Dataflow Graphs

- Each node represents computation process
  - constant production and consumption rate
  - executed a specific number of times during each complete cycle
- Edge represents a channel between two actors
  - FIFO protocol for tokens
  - initial number of tokens on edge (delays)

If for all edges \( p(e) = c(e) = 1 \), the graph is called unirate.

\[
\begin{align*}
q_A &= 3 & q_B &= 2 & q_C &= 3 \\
A &\quad 4 \quad 4 \quad 6 \quad 3 \quad 2 \quad 1 & B &\quad 1 & C &\quad 1
\end{align*}
\]
Chain-Structured Synchronous Data Flow Graphs

- First level of hierarchy, chain-structured Synchronous Data Flow Graph
- Each node $S_i$ reads data produced by $S_{i-1}$ and produces data for $S_{i+1}$
- Each node represents a pipeline stage and can be executed in parallel with other nodes
- Model commonly used for pipelined streaming applications
Second level: Each stage is a graph

Nodes of the graph represent hardware units (processes) that can be independently power gated

The edges of the graph represent data flow
Stage Execution

- The interval $L - l(S_i)$ is the slack time for the stage $S_i$.
- If for a process $v$ of stage $S_i$ power can be saved, the process is put to sleep mode while being idle.
- The number of consecutive executions of each stage is $1$ ($x=1$).
Changing the Stage Execution

- The interval $3L - 3l(S_i)$ is the slack time for the stage $S_i$
- More processes of stage $S_i$ can be put in sleep mode due to the increased idle time
- For some processes the penalty of switching mode is paid only once in $3L$
- The number of consecutive executions of stage $S_i$ is 3 ($x=3$)
Problem Formulation

- Determine the number of consecutive executions for each pipeline stage, so that the energy savings will be maximized
  - keep the average throughput of the application constant
  - take into account the energy penalty caused by the increase in the number of buffers
If for process $v$ of stage $G_s$ the idle time $l(G_s) - l(v)$ is not enough to put $v$ in sleep mode, then $v$ is a type-1 process.

For type-1 processes the energy savings are an increasing function of the number of consecutive executions ($x$) of the stage $G_s$.

An upper bound for the energy savings in $L$ cycles using this technique is derived.
If for process $v$ of stage $G_s$ the idle time $l(G_s) - l(v)$ is enough to put $v$ in sleep mode, then $v$ is a type-2 process.

For type-2 processes the energy savings in $L$ cycles are independent of the number of consecutive executions of $G_s$. 
Energy Penalty on Channels

- The energy penalty of an edge is an increasing, non-linear function of the number of buffers.
- Number of buffers is increasing with respect to:
  - the x value of the tail and head node (case 1)
  - the lcm of the x values of tail and head node (case 2 – unirate case)
  - the lcm of the x and q values of the tail and head node (case 3 – multirate case)
- Table that returns the energy penalty based on the x values of the tail and head node of the channel is input to the algorithm.
Bound on x – quality metric

- Given a quality metric $p$ as input to the algorithm, a bound $x_{max}$ can be derived for the x values from the parameters of the graph (case 1 and case 2).
- In solution space $[1, x_{max}]^{[S]}$ there is at least one solution for which $E_{sav} > (1-p) \ E_{max}$.
- For example if input $p=0.02$, there will be solution with $E_{sav} > 0.98 \ E_{max}$ in the solution space.
- The bound $x_{max}$ is derived from the effect of an increase in x can have on the energy savings of type-1 processes.
Bound on x – energy penalty

- Bound will be determined by energy penalty on channels between pipeline stages
- Theoretical limit on savings for type-1 processes
- Bound is derived when the penalty on a single edge exceeds all savings
- Bound is applicable to all three cases (increasing function, unirate graph, multirate graph)
Dynamic Programming Algorithm - Intuition

- Even after determining $x_{max}$, the solution space is still large ($x_{max}^{|S_i|}$)
- For the $x$ value of pipeline stage $S_i$, only the $x$ values of stages $S_{i-1}$ and $S_{i+1}$ need to be taken into account
- Algorithm finds best solution for each stage, for any $x$ values of the neighbor stages
- Then the algorithm combines the best solutions to solve the problem for each subchain, for any $x$ values of the neighbor stages
- The solution returned by the algorithm maximizes the energy savings for the whole chain-structured graph under the restriction that for all stages $x$ belongs to $[1, x_{max}]$
Dynamic Programming Algorithm
## Experimental Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Input Rate</th>
<th>Alg. Exec. Time (sec)</th>
<th>$xmax$</th>
<th>Increase in En. Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-to-DAT (multirate -3 stages)</td>
<td>50%</td>
<td>2.97</td>
<td>71</td>
<td>15.17%</td>
</tr>
<tr>
<td></td>
<td>25%</td>
<td>2.97</td>
<td>71</td>
<td>5.25%</td>
</tr>
<tr>
<td></td>
<td>12.50%</td>
<td>2.98</td>
<td>71</td>
<td>2.27%</td>
</tr>
<tr>
<td>K-means clustering (unirate - 10 stages)</td>
<td>11.10%</td>
<td>144.39</td>
<td>169</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>8.33%</td>
<td>29.26</td>
<td>100</td>
<td>107.31%</td>
</tr>
<tr>
<td></td>
<td>6.67%</td>
<td>3.37</td>
<td>49</td>
<td>6.71%</td>
</tr>
<tr>
<td>K-means clustering (unirate - 3 stages)</td>
<td>33.33%</td>
<td>5.79</td>
<td>100</td>
<td>900.38%</td>
</tr>
<tr>
<td></td>
<td>25.00%</td>
<td>0.7</td>
<td>49</td>
<td>24.25%</td>
</tr>
<tr>
<td></td>
<td>12.50%</td>
<td>0.06</td>
<td>16</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>10-stage K-means</th>
<th>3-stage K-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>0.9</td>
<td>0.95</td>
</tr>
<tr>
<td>Input Rate</td>
<td>6.67%</td>
<td>6.67%</td>
</tr>
<tr>
<td>Alg. Exec. Time (sec)</td>
<td>3.37</td>
<td>351</td>
</tr>
<tr>
<td>$xmax$</td>
<td>49</td>
<td>225</td>
</tr>
<tr>
<td>Increase in En. Savings</td>
<td>6.71%</td>
<td>6.71%</td>
</tr>
</tbody>
</table>
Conclusions

- Presented an algorithm to increase energy savings of pipelined streaming applications
- Algorithm increases energy savings obtained from power gating by finding the number of consecutive executions of each pipeline stage
- Energy savings are larger when the slack is not enough for all hardware units to be put into sleep mode
Thank you!