Lecture 1: Overview

Hsie-Chia Chang  張錫嘉
E-mail: hcchang@mail.nctu.edu.tw

Fall 2006
✓ Typical DSP Algorithm
  - Convolution, Correlation, Digital Filter, Adaptive Filter, Decimator and Expander, Viterbi Algorithm, Motion Estimation, Discrete Cosine Transform, Vector Quantization, Wavelets and Filter Banks

✓ Representations of DSP Algorithms
  - Block Diagram
  - Signal-Flow Graph
  - Data-Flow Graph
  - Dependence Graph

✓ Iteration Bound
  - Loop Bound and Iteration Bound
  - Algorithms for Computing Iteration Bound
  - Iteration Bond of Multi-rate Data-Flow Graphs

Optimized Application-Specific Integrated Systems
DSP has advantages over analog signal processing
  - Robust w.r.t. temperature, process variation, …
  - Higher precision by increasing wordlength
  - High signal to noise ratio
  - Repeatability and flexibility by algorithms

Algorithm
  - A set of rules for solving a problem in a finite number of steps
  - DSP algorithms can be found in packages and literatures easily

Two features of DSP
  - Real-time throughput requirement
    - No advantage if the processing rate faster than the input sample rate
  - Data-driven property
Typical DSP Algorithm (2/4)

- Convolution

- Correlation
  - The correlation operation can be described as a convolution
Digital Filters
- To modify the frequency properties of the input signal $x(n)$ to meet certain specific design requirements in LTI systems
  - FIR filter

  - IIR filter

- Linear phase FIR filters are attractive as their unit-sample responses are symmetric and require only half the number of multiplications.

Adaptive filters
- The coefficients are updated at each iteration in order to minimize the difference between the filter output and the desired signal
Decimator (compressor or downsampler)
- \( y_D(n) = x(Mn) \), where \( M \) is a positive integer
- Output rate is \( M \) times slower than input

Expander (interpolator or upsampler)
- \( y_E(n) = \begin{cases} x(n/L), & \text{if } n \text{ is integer multiple of } L \\ 0, & \text{otherwise} \end{cases} \)
- Every input sample, inserting \( L-1 \) zeros.

Decimator and Expander are nonlinear operations

Noble identities \( \rightarrow \) Delay elements transfer
✓ **Iteration period**
  - the time required for execution of one iteration of the algorithm

  \[ y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2] + h_3 x[n-3] \]

✓ **Critical path**
  - longest path between any 2 storage elements (delay elements)
  - Minimum feasible clock period

✓ **Sampling rate (throughput)**
  - number of samples processed per second

✓ **Latency**
  - The difference between an output generated and its corresponding input received by the system

✓ The clock rate of a DSP system is not the same as its sampling rate
DSP algorithm can be described by mathematic formations

- **Behavioral description**
  - Applicative language  e.g. Silage
  - Prescriptive language  e.g. C
  - Descriptive language  e.g. Verilog

- **Graphical description**
  - Block diagram
  - Signal-Flow graph
  - Data-Flow graph
  - Dependence graph  -> least structure bias

  **Graphical representations are efficient** for investigating and analyzing data flow properties of DSP algorithm and for exploiting the inherent parallelism
4 possible paths
- Input nodes to delay element
- Input node to output node
- Delay element to delay element
- Delay element to output

Example: 5-tap FIR filter and assume $T_A=4\text{ns}$, $T_M=10\text{ns}$

Critical paths = 26ns
A block diagram
- Consists functional blocks connected with directed edges
- Can be constructed with different levels of abstraction

A system can be represented using various block diagrams
- Data-broadcast structure

\[ y[n] = h_0 x[n] + h_1 x[n - 1] + h_2 x[n - 2] + h_3 x[n - 3] \]
A SFG is a collection of nodes and directed edges

- **Nodes**
  - **source** no entering edge
  - **Sink** only entering edge
  - adder, multiplier, ...

- **Directed edge (j,k)**
  - constant gain multipliers
  - delay elements

\[
y[n] = h_0 x[n] + h_1 x[n - 1] + h_2 x[n - 2]
\]
Transposition of SFG is applicable to linear SISO systems
- Reserve the direction of all edges
- Exchange input and output

\[ y[n] = h_0 x[n] + h_1 x[n - 1] + h_2 x[n - 2] \]

Transpose operations are also applicable to MIMO systems described by symmetric transformation matrices.
In DFG representations,

- **Each node** associate an execution time
  - computations, functions, or tasks
- **Each edge** may have a nonnegative number of delays
Data-Flow Graph (DFG)

✓ **Data-driven property** can be captured by the DFG
  - Node fire

  - Many nodes can be fired simultaneously -> Concurrency
  - Each directed edge -> Precedence constraint

✓ **Intra-iteration precedence constraint**
  - The edge has zero delays

✓ **Inter-iteration precedence constraint**
  - One or more delays
✓ **Synchronous Data-Flow-Graph (SDFG)**
  - A special case of DFG where the number of data samples produced or consumed by each node in each execution is specified a priori.

✓ **Single rate system**

✓ **Multi-rate SDFG**

3f_A = 5f_B  
2f_B = 3f_C

---

**Optimized Application-Specific Integrated Systems**
A DG is a directed graph to show the dependence of the computation in an algorithm
- **Nodes**: computation
- **Edge**: precedence constraint

DGs are widely used in systolic array designs
- SFGs can be derived by DGs
Summary

✓ **Block diagram**

✓ **SFG**
  - It provides an abstract flowgraph representation of linear networks and have been extensively used in digital filter structure design and analysis of finite wordlength effects.

✓ **DFG**
  - It’s generally used for high-level synthesis to derive concurrent implementation of DSP applications onto parallel hardware, where subtask scheduling and resource allocation are of major concern.

✓ **DG**
  - It’s widely used in systolic array designs.
**Iteration Period**

- **Iteration**
  - For a node, it’s the execution of the node exactly once.
  - For a DFG, it’s the execution of **each node** in the DFG exactly once.

- **Iteration period**
  - the time required for execution of one iteration

- **Iteration rate**
  - the number of iterations executed per second

\[ A_k \Rightarrow B_k \]
\[ B_k \Rightarrow A_{k+1} \]

\[ y(n) = a \cdot y(n-1) + x(n) \]
\[ H(z) = \frac{1}{1 - a \cdot z^{-1}} \]
Loop Bound

✓ Loop (cycle)
  - a directed path that begins and ends at the same node

✓ Loop bound of the loop j \( T_j / W_j \)
  - \( T_j \) is the loop computation time
  - \( W_j \) is the number of delays in the loop
  - Critical loop is the loop with the maximum loop bound

Examples:

\[
y(1) = x(1) + a \ y(-1) \quad \text{Only odd input samples!}
\]

\[
y(2) = x(2) + a \ y(0) \quad \text{Only even input samples!}
\]

\[
y(3) = x(3) + a \ y(1)
\]

- The loop bound = 3

\[
y(n)=ay(n-2)+x(n)
\]
Many DSP Algorithms contain feedback loops

Iteration bound

- An inherent lower bound on the iteration (or sample period)
  - It’s not possible to achieve iteration period lower than iteration bound even with infinite processing elements
- The loop bound of the critical loop

\[ T_\infty = \max_{j \in L} \left\{ \frac{T_j}{W_j} \right\} \]

where \( L \) is the set of loops in the DSP system, \( T_j \) is the computation time of the loop \( j \) and \( W_j \) is the number of delays in the loop \( j \)

\[
T_{L1} = \frac{10+2}{1} = 12\text{ns} \\
T_{L2} = \frac{2+3+5}{2} = 5\text{ns} \\
T_{L3} = \frac{10+2+3}{2} = 7.5\text{ns} \\
T_\infty = \max\{12, 5, 7.5\}
\]
Optimized Application-Specific Integrated Systems

Remarks

Critical path = 6
Loop bound = 6

\[ A_0 \rightarrow B_0 \Rightarrow A_1 \]

Critical path = 6
Loop bound = 6/2 = 3

\[ A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \ldots \]
\[ A_1 \rightarrow B_1 \Rightarrow A_3 \rightarrow B_3 \Rightarrow A_5 \ldots \]
\[ A_N \Rightarrow B_{N+1} \Rightarrow A_{N+2} \Rightarrow B_{N+3} \ldots \]

Same loop bound
Different critical path

Retiming
Long execution time for finding the iteration bound
- It’s because the number of loops in a DFG can be exponentially with respect to the number of nodes

Two algorithms for computing $T_\infty$
- Longest Path Matrix (LPM) Algorithm
- Minimum Cycle Mean (MCM) Algorithm
LPM Algorithm (1/2)
LPM Algorithm (2/2)

- Example:

```
\begin{align*}
L^{(1)} &= \begin{pmatrix}
-1 & 0 & -1 & -1 \\
4 & -1 & 0 & -1 \\
5 & -1 & -1 & 0 \\
5 & -1 & -1 & -1 
\end{pmatrix} \\
L^{(2)} &= \begin{pmatrix}
4 & -1 & 0 & -1 \\
5 & 4 & -1 & 0 \\
5 & 5 & -1 & -1 \\
-1 & 5 & -1 & -1 
\end{pmatrix} \\
L^{(3)} &= \begin{pmatrix}
5 & 4 & -1 & 0 \\
8 & 5 & 4 & -1 \\
9 & 5 & 5 & -1 \\
9 & -1 & 5 & -1 
\end{pmatrix} \\
L^{(4)} &= \begin{pmatrix}
8 & 5 & 4 & -1 \\
9 & 8 & 5 & 4 \\
10 & 9 & 5 & 5 \\
10 & 9 & -1 & 5 
\end{pmatrix}
\end{align*}
```

$T_\infty = \max\{4/2, 4/2, 5/3, 5/3, 5/3, 8/4, 8/4, 5/4, 5/4\} = 2.$
**MCM Algorithm**

- Example:

![Graph](image.png)

\[ G_d \text{ to } G_d' \]

<table>
<thead>
<tr>
<th></th>
<th>m=0</th>
<th>m=1</th>
<th>m=2</th>
<th>m=3</th>
<th>( \max_{m \in {0, 1, \ldots, d-1}} \left( \frac{f^{(d)}(i) - f^{(m)}(i)}{(d-m)} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td>-2</td>
<td>-(\infty)</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>i=2</td>
<td>-(\infty)</td>
<td>-5/3</td>
<td>-(\infty)</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>i=3</td>
<td>-(\infty)</td>
<td>-(\infty)</td>
<td>-2</td>
<td>-(\infty)</td>
<td>-2</td>
</tr>
<tr>
<td>i=4</td>
<td>-(\infty)</td>
<td>-(\infty)</td>
<td>-(\infty)</td>
<td>-(\infty)</td>
<td>(\infty)</td>
</tr>
</tbody>
</table>

\[ T_\infty = -\min\{-2, -1, -2, \infty\} = 2 \]
T∞ of Multirate DFGs (MRDFGs)

✓ Construct the equivalent single-rate DFG (SRDFG)
  - Compute the iteration bound of the equivalent SRDFG
  - The iteration bound of the MRDFG is the same as the iteration bound of the equivalent SRDFG.

\[ k_x : \text{the number of nodes related to } "x" \]

- \[ k_a = 3 \]
- \[ k_b = 4 \]
- \[ k_c = 2 \]

Note: same number of delay elements