SIE40AM

VLSI Design for Digital Signal Processing Applications

Lars Wanhammar
Departments of Telecommunication and Physical Electronics
The Norwegian University of Science and Technology
Trondheim, Norway
and
Department of Electrical Engineering
Linköping University
Linköping, Sweden
+46 13 281344
larsw@isy.liu.se
Textbook


Wideband OFDM Radio Systems

DSP/Coding → IFFT → D/A → H(s) → DSP/Decoding

Digital → Mixed Signal → Analog

H(z) → D/A → H(s) → OSC → ej0

Analog → Mixed Signal → Digital

OSC → ejπ/2

H(s) → A/D → H(z) → FFT

DSP/Coding

Department of Electrical Engineering
Lars Wanhammar
Linköping University
larsw@isy.liu.se
http://www.es.isy.liu.se/
Heterogeneous Software/Hardware Architectures

- μProcessors
- Control Processors
- DSP Processors
- Algorithm-Specific Processors
- Fixed-Function Units
- Analog Circuits
- Mixed-Signal Circuits
Type of DSP Systems

Real-Time – Nonreal-Time
Resource Adequate – Resource Limited
Complexity – Throughput
Recursive Algorithms – Nonrecursive Algorithms
Data Dependent – Data Independent
Static Scheduling – Dynamic Scheduling of Operations
Control Dominated – Data Dominated
Irregular – Regular/Modular
Design and Implementation Constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Design Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed/Variable Throughput</td>
<td>Design Resources</td>
</tr>
<tr>
<td>Resource Adequate/Limited</td>
<td>Skilled Manpower</td>
</tr>
<tr>
<td>Fixed/Multi Functional</td>
<td>Previous Design Experience</td>
</tr>
<tr>
<td>Energy/Power Limited</td>
<td>Product Family</td>
</tr>
<tr>
<td>Size</td>
<td>Design Reuse</td>
</tr>
<tr>
<td>Volume (number of units)</td>
<td>Platform Based</td>
</tr>
<tr>
<td>Flexibility (Design modifications)</td>
<td>CAD Tools</td>
</tr>
<tr>
<td>Technology Independence</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Design Issues

- **Design Efficiency**
  - Correctness
  - Design time
  - Flexibility
  - Reuse

- **Constraints**
  - Resource adequate vs. resource limited
  - Power consumption – Energy limited

- **Cost**
  - Standard digital CMOS technology
  - Chip Area

- **Energy Efficiency**
Some Observations

- More and more sophisticated and complex systems
- Research intensive – small step from research to application
- More efficient design methods needed – short design time
- Broad competence in application domain, signal processing, algorithms, arithmetic, and electronics, is required
- **Necessary to work in small teams at all levels of a design** (A good team of 11 players)
- New cost measures – Design and energy efficiency
- Necessary to optimize energy efficiency/power consumption at all levels of the design
- Data dependencies are becoming more important
Standard and Application/Algorithm-Specific Digital Signal Processors

<- High Flexibility

High Performance ->
### Standard DSPs

+ Standardized hardware structure
+ Emphasis on short design time
+ Flexible
+ Easy to modify/correct errors
+ Low cost due to the wide applicability of the hardware
  – Low performance/throughput
  – High power consumption
  – The flexibility is not needed in many applications/overhead
  – Not always cost-effective

### Application-Specific DSPs

+ Some IP protection
+ Some CAD tools available
  – Inflexible
  – Very difficult
+ Lower unit cost, but higher initial cost
+ Higher performance/throughput
+ Lower power consumption
+ Optimized

...
Algorithm-Specific Digital Signal Processors

- Programmable (at design time) processor cores
- Specialized DSP cores
- Direct Mapping Techniques

The basic idea is to optimize the amount and usage of resources with respect to the requirements and thereby minimize some cost function.
Direct Mapping Techniques

- Partition the system into parts that are implemented with suitable software/hardware structures
- Subsystems are connected according to the signal-flow graph of the whole system
- Asynchronous communication is used between the subsystems (no global clock)
- Synchronous clocking of the subsystems (well-known design problem)
- Globally Asynchronous and Locally Synchronous – GALS Approach
- Schedule the processing elements (PEs) to meet the requirements and minimize a cost function
Describing and Modeling DSP Systems

Facets

Assumption:

A human can only handle a handful of items/concepts at a time

Hence, we need to use many different facets, or views to describe a complex system
Definition: A **behavioral description** is an input–output description that defines the required action of a system in response to prescribed inputs.

The description of the behavior may not include directions about the means of implementation or performance measures such as speed of operation, size, and power dissipation unless they directly affect the application.

Definition: A **functional description** defines the manner in which the system is operated to perform its function.

Functional view of CCITT H.261 video encoder
Onionskin view: The idea is to reduce the design complexity of the system by using a hierarchy of views which usually are referred to as virtual machines.

Each virtual machine provides the basic functions that are needed to realize the virtual machine in the next higher layer.

Hence, the onionskin view represents a hierarchy of virtual machines.
Definition: An architectural description describes how a number of objects (components) are interconnected.

An architectural description is sometimes referred to as a structural description.
System Design Methodology

“The starting point for the system design phase is the system specification.”

Much better to start with a problem understanding phase!

Definition: A *design methodology* is the overall strategy to organize and solve the design tasks at the different steps of the design process.

It is necessary due to the high complexity of the design problem to follow a *structured design* approach that reduces the complexity. Structured design methods are primarily used to

- **Guarantee that the performance goals are met and**
- **Attain a short and predictable design time**
Definition: A *specification* has two main parts

- A behavioral description that specifies *what* is to be designed and

- A verification or validation part that describes *how* the design should be verified (validated).

Definition: *Verification* involves a formal process of proving the equivalence of two different types of representations under all specified conditions.

Definition: *Validation* is an informal and less rigorous correctness check.

Validation is usually done by simulating the circuit with a finite set of input stimuli to assert that the circuit operates correctly.
Partitioning Techniques

Data-Flow Approach

(a) Vertical and (b) horizontal partitioning

Top-Down Approach

Bottom-Up Approach
Edge-In Approach

Emphasis on interfaces and communication

Critical Part Approach

“I do the important (difficult) stuff (that people will notice) and you do the rest.”

“Let’s do the easy stuff first and we move-on to a more important project later on.”
Design Strategy

Meet-In-The-Middle Approach

SYSTEM DESIGN
Specification/Validation
Algorithm
Scheduling
Architecture
Floorplanning

MEET-IN-THE-MIDDLE

VLSI AND CIRCUIT DESIGN
Logic
Gate
Transisor
Module
Cell
Layer
Layout

Emphasis on concurrent design
Sequence of Models Approach

Design Process: Develop a sequence of (executable) models, preferable with a hierarchy, incorporating more and more functionality and details.

Emphasis on problem understanding, design space exploration, and supports innovation.

Example of design of a digital filter

Phase 1: Golden model

Step 1: Determine the transfer function, poles and zeros and verify/validate its function.

Step 2: Select an algorithm and implement it using, e.g., MATLAB, and validate it against step 1.

Step 3: Scale the signal levels and determine the roundoff noise and dynamic range. Validate. …

Phase 2: Develop a corresponding Simulink model
Step 1: Signal-flow model for one sample interval
Step 2: Signal-flow model for maximally fast realization when the latencies of the operators are given. …

Phase 3: Logic level
Step 1: Logic description in simulink/VHDL.
Step 2: Introduce shimming delays.
Step 3: Introduce quantization, timing circuitry.
Step 3: Determine appropriate control signals. …

Phase 4: Bit-serial model
Step 1: Simplify the multipliers. …

…

Documentation: Obtained results and experience
Design Transformations

Synthesis
Abstraction
Optimization
Analysis
Verification
Validation
Complexity Issues

The *complexity* of a system can be measured in terms of the number of interactions between its parts. More formally we have

\[ \langle O, F, R \rangle \]

where \( O \) is a set of objects with their functional description, \( F \), and their interrelations, \( R \).

The reduction in complexity achieved by grouping several parts into a larger object (module) that can be described by a simpler representation, describing only external communication and without explicit references to any internal interactions, is called *abstraction*.

*Hierarchical abstraction* is the iterative replacement of groups of modules.

Note that using hierarchy alone does not reduce the complexity.

A design that can be completely described by using modules (abstractions) is *modular* and will have low complexity.
If the design has only a few types of modules, the complexity is even lower. Such designs have a high degree of *regularity*.

A regularity factor can be defined as the ratio of the total number of modules to the number of different modules.

*Standardization* that restricts the design domain can be applied at all levels of the design to simplify modules and increase the regularity, thereby reducing the complexity.
Algorithm Complexity

■ Comparing algorithms

It is often important to know how rapidly the execution time grows with problem size.

Let $g(n)$ be a function describing the execution time of an algorithm.

A function $g(n)$ is a member of $O(f(n))$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \text{const} < \infty.$$  

That is, the function $g(n)$ grows as fast as $f(n)$.

Growth rate, e.g. $O(\log(n)) < O(n) < O(n\log(n)) < O(n^2) < O(2^n) < O(n!)$.  

■ Average vs. worst-case

Many algorithms have a much worse worst-case execution time than its average performance.
Actual execution time is interesting as well!

Consider two algorithms, one with exponential execution time $1.001^n$ and one with polynomial execution time $10^4 n^6$. The exponential execution time algorithm will be faster for $n < 76738$.

The Divide-And-Conquer Approach

function Solve($P$);
begin
  if size($P$) $\leq$ MinSize then
    Solve := Direct_Solution($P$)
  else
    begin
      Decompose($P$, $P_1$, $P_2$, ..., $P_b$);
      for $i$ := 1 to $b$ do
        $S_i$ := Solve($P_i$);
      end;
      Solve := Combine($S_1$, $S_2$, ..., $S_b$)
    end;
  end if;
end;
The amount of time required at each step is

\[
T(n) = \begin{cases} 
  a & \text{for } n \leq \text{MinSize} \\
  bT\left(\frac{n}{c}\right) + d \cdot n & \text{for } n > \text{MinSize}
\end{cases}
\]

where \( n \) is the size of the problem, 
\( a \) is the time required to solve the minimum-size problem, 
\( b \) is the number of subproblems in each stage, 
\( n/c \) is the size of the subproblems, and 
\( d \cdot n \) is the linear amount of time required for decomposition and combination of the problems.
It can be shown that divide-and-conquer algorithms have the time-complexity:

\[
T(n) \in \begin{cases} 
O(n) & \text{for } b < c \\
O(n \log_c(n)) & \text{for } b = c \\
O\left(n \log_c(b)\right) & \text{for } b > c
\end{cases}
\]

Thus, recursively dividing a problem, using a linear amount of time, into two problems \((b = 2)\) of size \(n/2\), \((c = 2)\), results in an algorithm with time–complexity of \(O(n \log_2(n))\).

The fast Fourier transform (FFT) is an example of this type of algorithm.

If the number of subproblems were \(b = 3, 4,\) or \(8\), then the required execution time would be \(O(n \log_2(3))\), \(O(n^2)\), or \(O(n^3)\), respectively.
Integrated Circuit Design

The turnaround time for design changes ranges from several weeks to many months. Long design times may lead to lost opportunities of marketing the chip ahead of the competition and recouping the investment. The correctness of the design is of paramount importance for a successful project.

Technical Feasibility

- **SYSTEM-RELATED**
  - Partitioning into cabinets, boards, and circuits
  - Mixed digital and analog circuits on the same chip
  - Clock frequencies
  - Power dissipation and cooling
  - Circuit area and packaging
  - I/O interface
CIRCUIT–RELATED

External

Interchip propagation delay
Data transfer frequencies
Input protection
Loads that have to be driven, including expected PCB runs
Available input drivers
Drive capacity for output buffers
Restrictions on pin-outs

Internal

Clock frequencies
Data transfer frequencies and distances
Critical timing paths
Non critical timing paths
Power dissipation and and cooling
Drive capacity for internal buffers
Circuit area, yield, and packaging
Temperature and voltage effects
Maximum and minimum temperatures, voltages, etc.
Process technology

■ DESIGN-EFFORT-RELATED

CAD tools
Layout style
Regularity and modularity of the circuits
Module generators
Cell library