Application Specific Integrated Circuits for Digital Signal Processing
Lecture 12

Oscar Gustafsson

Today's topic

- Course review

Signal flow graphs

- Can be expressed either in time domain (delay T) or frequency domain (delay z^{-1})
- No delay free loops to be sequentially computable
- Fully specified: expressed in the operations that will be used for implementation
- Precedence graph: determines which operations are sequential and which can be performed concurrently
- Operations in computable order: difference equations can be derived based on the precedence graph to obtain a sequential "program"

Algorithm properties

- Parallelism
  - Recursive algorithms

$$T_{\text{min}} = \max_i \frac{\sum T_{L,i}}{N_i}$$

  - Non-recursive: unlimited
  - Critical path: longest time directed path, T_{cp}
- Operations
  - Latency: delay from input to corresponding output $\rightarrow$ scheduling, T_{min}, T_{cp}
  - Execution time: delay from input (output) to next input (output) $\rightarrow$ resource usage
- Finite word-length effects
  - Scaling: avoid overflows while utilizing the complete numerical range
  - Data quantization: round-off noise, possibly stability issues for recursive algorithms
  - Coefficient quantization: static deviation

Algorithm transformations

- Pipelining: introduce delay elements → change T_{cp}
- Retiming: move delay elements → change T_{cp}
- Reordering and/or merging operations → change T_{cp} and T_{min}
- Lookahead pipelining and block processing → change T_{min} (and T_{cp}), numerical issues
- Unfolding: express algorithm over several sample periods → does NOT change T_{min} (but may allow transformations)

Resource allocation and assignment

- Resource allocation: determining the number of resources
- Resource assignment: determining which process/operation is using which resource
- Often performed concurrently
  - Clique partitioning: inclusion or exclusion graph, NP-hard
  - Left-edge algorithm: fast heuristic
- Resource assignment will determine communication complexity
- Memory/storage
  - Number of concurrent accesses determines the number of memories/memory ports
  - Number of concurrent variables to be stored determines the number of cells

Scheduling

- When to execute which process (variable storage is also a process)
- Will determine the resource requirements
- Cyclic scheduling
  - For iterative processing, there will always be a next iteration
  - Outputs does not have to be explicitly "flushed out"
  - Can schedule on a cylinder → no explicit start or stop time
  - Moving an operation across the scheduling boundary equivalent to pipelining/retiming
- Scheduling period
  - Integer multiple of T_{sample}
  - Integer multiple of measure time (commonly clock cycles)
  - At least as long as maximum execution time (including variable storage)

Architecture

- Four separate parts
  - Processing elements: performs the computations
  - Memory: store information
  - Communication: connects memory and PEs
  - Control: determines what and when things happen
- Shared memory architecture
  - N PE and M memories, each PE has exclusive access during one cycle
  - Memory bottleneck

$$T_{exe,PE} \geq 2NT_{exe,mem}$$

  - Increase $T_{exe,PE}$: use larger operations → less flexible schedule, fewer memory accesses
  - Decrease $T_{exe,mem}$: interleave memories → constraints on schedule
  - Decrease memory accesses: broadcasting, cache memories → constraints on schedule

Arithmetic

- Fixed-point representations: main problem is to represent negative numbers
- Floating-point representations: higher dynamic range, but lower resolution, more complex operations
- Adders: carry propagation
- Multiplication: handling sign-bits
- Distributed arithmetic: efficient way to compute inner products
- Constant multiplication: avoid unnecessary computations, share computations

Course description style

- General methods and principles used for designing and implementing application specific integrated circuits are presented
- The emphasis is put on a systematic design methodology going from system specification down to complete integrated circuit
- After completing the course you are expected to be able to:
  - Analyze computational properties of signal processing algorithms
  - Determine resource requirements for implementation of signal processing algorithms
  - Determine the suitability of different types of architectures for implementation of signal processing algorithms
  - Synthesize nearly optimal architectures
  - Realize and evaluate different types of processing elements

Course description style

- As parts of the course the student is expected to be able to:
  - Determine limits on the obtainable data rate
  - Determine a computational order of operations
  - Compute the latency and execution time of operations and determine the impact of these properties
  - Determine computational graphs and optimize schedules
  - Define and analyze different types of optimality in parts of the design flow
  - Determine and optimize resource allocation and resource assignment
  - Synthesize and analyze different architectures for signal processing systems
  - Realize different types of processing elements using different arithmetic styles