ESE532:
System-on-a-Chip Architecture

Day 28: Dec 11, 2017
Wrapup

Today

- What was course about
- Final
- Review w/ Simple Models
- Questions/Discussion
- Other Courses

Goal

- How to design/select/map to SoC to reduce Energy/Area/Delay.

Day 1

Outcomes

- Design, optimize, and program a modern System-on-a-Chip.
- Analyze, identify bottlenecks, design-space
- Decompose into parallel components
- Characterize and develop real-time solutions
- Implement both hardware and software solutions
- Formulate hardware/software tradeoffs, and perform hardware/software codesign

Day 1

Outcomes

- Understand the system on a chip from gates to application software, including:
  - on-chip memories and communication networks, I/O interfacing, RTL design of accelerators, processors, firmware and OS/infrastructure software.
- Understand and estimate key design metrics and requirements including:
  - area, latency, throughput, energy, power, predictability, and reliability.

Day 2

Message for Day

- Identify the Bottleneck
  - May be in compute, I/O, memory, data movement
- Focus and reduce/remove bottleneck
  - More efficient use of resources
  - More resources
- Repeat

Day 1

Abstract Approach

- Identify requirements, bottlenecks
- Decompose Parallel Opportunities
  - At extreme, how parallel could make it?
  - What forms of parallelism exist?
    - Thread-level, data parallel, instruction-level
- Design space of mapping
  - Choices of where to map, area-time tradeoffs
- Map, analyze, refine

SoC Designer Hardware Building Blocks

- Computational blocks
  - Adders, multipliers, dividers, ALUs
- Registers
- Memory blocks
- Busses
- Multiplexers, Crossbars
- DMA Engines
- Processors
- I/O blocks
  - Ethernet, USB, PCI, DDR, Gigabit serial links

Final

Final

- Analysis
  - Bottleneck
  - Amdhal's Law Speedup
  - Computational requirements
  - Resource Bounds
  - Critical Path
  - Latency/throughput
- Model/estimate speedup, area, energy, yield
- From Code
- Forms of Parallelism
- Dataflow, SIMD, VLIW, hardware pipeline, threads
- Map/schedule task graph to (multiple) target substrates
- Memory assignment and movement
- Area-time points
- Real Time

Final

- Like midterm
  - Content
  - Style
  - Closed book, notes
  - Calculators allowed (encouraged)
- Last term's final and solutions
  - Linked to Spring 2017 syllabus
- Thursday, December 21 3pm—5pm
  - Towne 321 (here)
  - 2 hours (standard final exam period)

Review

Sequential Computation

- Computation requires a collection of operations
  - Arithmetic
  - Logical
  - Data storage/retrieval

$$T = \sum T_{op_i}$$

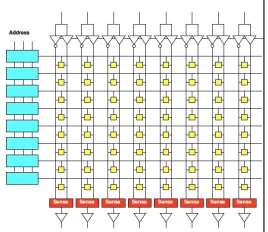
Computations in C

- Express computations in a programming language, such as C
- Can execute computation in many different ways
  - Sequential, parallel, spatial hardware

```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
}
```


Memory Characteristics

- Small memories
  - Fast
  - Low energy
  - Not dense
    - (high area per bit)
- Large memories
  - Slow
  - High energy
  - Dense (less area per bit)



Memory and Compute

- Computation involves both arith/logical ops and memory ops

$$T = \sum T_{op_i}$$

$$T = \sum (op_i == mem) \times T_{mem} + \sum (op_i == alu) \times T_{alu}$$

$$T = N_{memop} \times T_{mem} + N_{alu} \times T_{alu}$$

Memory and Compute

- Computation involves both arith/logical ops and memory ops
- Either can dominate
  - Be bottleneck

$$T = \sum T_{op_i}$$

$$T = N_{memop} \times T_{mem} + N_{alu} \times T_{alu}$$

Memory and Compute

- Timing
  - $T_{mem}=10$
  - $T_{alu}=1$
- $N_{mpy}=$
- $N_{add}=$
- $N_{mem}=$
- Total Time, T?

```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
}
```

$$T = N_{memop} \times T_{mem} + N_{alu} \times T_{alu}$$

(not count loop, indexing costs in this sequence)

Memory and Compute

- Where bottleneck?
  - T_{mem}=10
  - T_{alu}=1
- N_{mpy}=
- N_{add}=
- N_{mem}=

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```

$$T = N_{memop} \times T_{mem} + N_{alu} \times T_{alu}$$

Data Reuse

- Reduce memory operations to large memory by storing in
  - Registers
  - Small memories

$$T = N_{memop} \times T_{mem} + N_{alu} \times T_{alu}$$

$$T = N_{smem} \times T_{smem} + N_{lmem} \times T_{lmem} + N_{alu} \times T_{alu}$$

$$N_{lmem} < N_{smem}$$

Memory and Compute

- a,b in large mem
- y,z in small mem
  - T_{lem}=10
  - T_{sem}=1
  - T_{alu}=1
- N_{mpy}=16
- N_{add}=15
- N_{lmem}=?
- N_{smem}=?

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```

$$T = N_{smem} \times T_{smem} + N_{lmem} \times T_{lmem} + N_{alu} \times T_{alu}$$

Memory and Compute

- a,b in large mem
- y,z in small mem
  - T_{lem}=10
  - T_{sem}=1
  - T_{alu}=1
- N_{mpy}=16
- N_{add}=15
- N_{lmem}=
- N_{smem}=

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```

- Performance impact?

$$T = N_{smem} \times T_{smem} + N_{lmem} \times T_{lmem} + N_{alu} \times T_{alu}$$

Task Parallel

- Can run unrelated tasks concurrently

$$T = \sum T_{op_i}$$

$$T = \max\left(\sum T_{op_i}(op_i \in Task1), \sum T_{op_i}(op_i \in Task2)\right)$$

Ideal: T(p) = T(1)/p

Data Parallel

- Can run same task on independent data in parallel

$$T = \sum T_{op_i}$$

Ideal: T(p) = T(1)/p

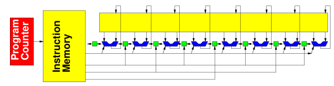
Data Parallel

- What's data parallel?

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```


Vector/SIMD

- Can perform same operation on a set of data items



$$T = \sum T_{op_i}$$

Ideal: T(VL) = T(1)/VL
(Vector Length)

Vector/SIMD

- Can perform same operation on a set of data items
  - Not everything vectorizable

$$T = \sum T_{op_i}$$

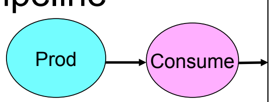
$$T = \left(\frac{1}{VL}\right)\sum T_{op_i}(vectorize(op_i)) + \sum T_{op_i}(\overline{vectorize(op_i)})$$

Vector/SIMD

- What's vectorizable?
- Speedup vector piece VL=4
  - (assume both memory and compute speedup)
- Overall speedup
- Amdahl's Law speedup for VL→infintiy?

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```


Dataflow Pipeline



- With no cycles in flowgraph
  - (no feedback)
  - (no loop carried dependencies)
- Producer and consumer can operate concurrently

$$T = \sum T_{op_i}$$

$$T = \max\left(\sum T_{op_i}(op_i \in Prod), \sum T_{op_i}(op_i \in Consume)\right)$$

Data Flow Pipeline

- Producer/consumer here?
- Time each
  - Assuming T_{mem}=1

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```

Spatial Pipeline

- Can build spatial pipeline of hardware operators to compute a dataflow graph

$$T = \sum T_{op_i}$$

- With no feedback: T=1
- With feedback loop length L: T=L

Spatial Pipeline

- Pipeline taking one a[i], b[i] per cycle?

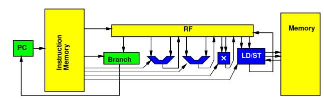
```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
  }
```


Spatial Pipeline

- Full pipeline taking all 16 a's, b's in cycle?
- Latency?

```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
  }
```


VLIW



- Can control datapath to perform multiple, heterogeneous operations per cycle
  - Tune parallelism
- Op types: A, B, C
  - Ops N_A, N_B, N_C
  - Number of Hardware Units H_A, H_B, H_C

$$T = \sum T_{op_i}$$

- $T_{RB}=\max(N_A/H_A, N_B/H_B, N_C/H_C,\ldots)\leq T_{VLIW}$
- $T_{CP}\leq T_{VLIW}$

VLIW

- VLIW
  - 1 load/store (a,b)
    - Assume single cycle
  - 1 mpy
  - 1 add
- RB
- CP
- Schedule

```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
  }
```


VLIW

- VLIW
  - 4 load/store (a,b)
    - Assume single cycle
  - 2 mpy
  - 2 add
- RB
- CP

```
while(true) {
   for (i=0;i<16;i++)
      y[i]=a[i]*b[i];
   z[0]=y[0];
   for (i=1;i<16;i++)
      z[i]=z[i-1]+y[i];
  }
```

Memory Bottleneck

- Memory can end up being bottleneck

$$T = \sum T_{op_i} = T_{comp} + T_{mem}$$

Ideal: $T(p) = T_{comp}(1)/p + T_{mem}$

Memory Bottleneck

- Pipeline to perform one multiply per cycle
- Memory supplies one data item from large memory (a,b) every 10 cycles
- Performance?
  - (number of cycles)

```
while(true) {
  for (i=0;i<16;i++)
    y[i]=a[i]*b[i];
  z[0]=y[0];
  for (i=1;i<16;i++)
    z[i]=z[i-1]+y[i];
}
```


Memory Bottleneck

- Memory can end up being bottleneck

$$T = \sum T_{op_i} = T_{comp} + T_{mem}$$

Ideal: $T(p) = T_{comp}(1)/p + T_{mem}/\text{Membw}$

Memory Bottleneck

- Invest in higher bandwidth
  - Wider memory
  - More memory banks
- Exploit data reuse
  - Smaller memories may have more bandwidth
  - Smaller memories are additional banks
    - More bandwidth

Communication

- Once parallel, communication may be bottleneck

$$T = \sum T_{op_i} = T_{comp} + T_{mem} + T_{comm}$$

- Ideal (like VLIW):
  $T \leq \max(T_{comp}/p, T_{mem}/\text{membw}, T_{comm}/\text{netbw})$

Area

- Can model area of various solutions
  - Sum of component areas
  - (watch interconnect)
- Cost proportional A/P_{chip}
- Without sparing
  - $P_{chip} = (P_{mm})^A$
- Sparing
  - Pmofn

$$A = \sum A_i$$

$$P_{chip} = \prod P_i$$

Multi-Objective Optimization

- Many forms of parallelism
- Given fixed area (resources, energy)
  - Maximize performance
  - Select best architecture
- Given fixed performance goal
  - Find architecture that achieves
  - With minimum area (energy, cost)

Final

- Analysis
  - Bottleneck
  - Amdhal's Law Speedup
  - Computational requirements
  - Resource Bounds
  - Critical Path
  - Latency/throughput
- Model/estimate speedup, area, energy, yield
- From Code
- Forms of Parallelism
- Dataflow, SIMD, VLIW, hardware pipeline, threads
- Map/schedule task graph to (multiple) target substrates
- Memory assignment and movement
- Area-time points
- Real Time

Question/Discussion

Other Courses

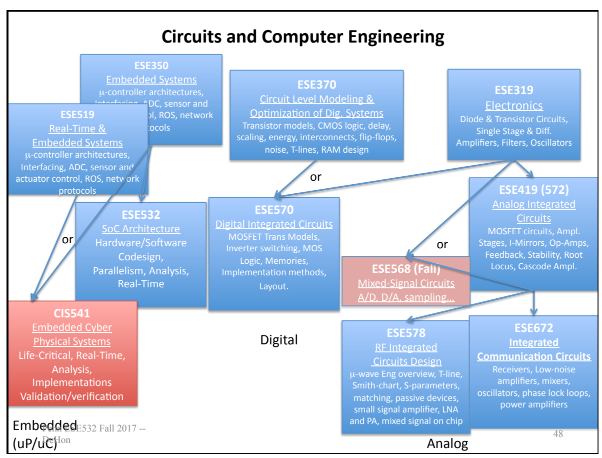
Distinction

CIS240, 371, 501

- Best Effort Computing
  - Run as fast as you can
- Binary compatible
- ISA separation
- Shared memory parallelism
- Caching – automatic memory management
- Superscalar
- Pipelined processor

ESE532

- Hardware-Software codesign
  - Willing to recompile, maybe rewrite code
  - Define/refine hardware
- Real-Time
  - Guarantee meet deadline
- Non shared-memory models



Other Courses

- Security: CIS331, CIS551
- Networking: ESE407, CIS553
- GPGPU (graphics focus): CIS565

Message

- Any interesting, challenging computation will require both hardware and software
- SoC powerful implementation platform
  – Target pre-existing
  – Design customized for problem
  – Exploit heterogeneous parallelism
- Understand and systematically remove bottlenecks
  – Compute, memory, communicate, I/O

Admin

- Return Zed Boards
- Final Q&A (office hour)
  – Friday, Dec. 15th
  – Time TBA (watch piazza)
- Final: Thursday, Dec. 21 3pm—5pm
  – Towne 321 (here)