15-745

Topic: Exotic Architectures

- Doug Burger et.al., "*Scaling to the End of Silicon with EDGE Architectures*", IEEE Computer July 2004.
- Jan Hoogerbrugge, et al., *``Software pipelining for transport-triggered architectures''*, MICRO 24 (1991).
- Steve Swanson, et al. "*WaveScalar*", MICRO-36, December 2003

Its not about computing after all!

- What is the fundamental operation in a computer?

It's not about computing after all!

- What is the fundamental operation in a computer?
  - It is not the add, the multiply, the xor, etc.
  - It is the move
- Typical (read x86,etc.) architectures don't ALLOW this to be expressed!
- All three papers share a common goal:

  Represent the data movement involved in computation explicitly

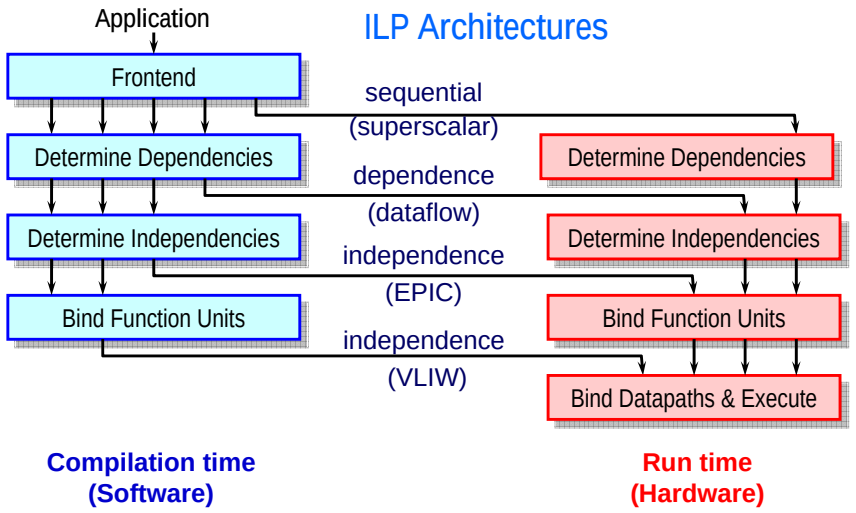
BTW: Really bad slide, why?

What is Exotic?

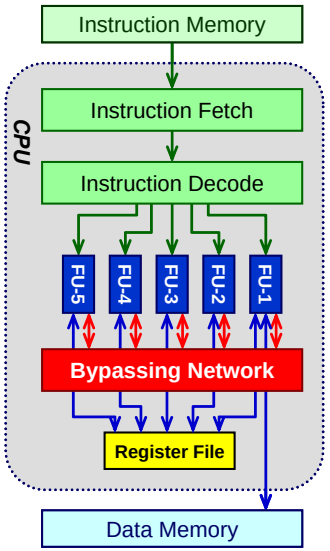
- ISA
  - An abstraction provided by computer designer

    E.g., no change in programs when
    - transistor shrinks by factor of 2 or even 10!
    - start using aluminum to transmit info (and then copper!)
    - voltage changes by factor of 5x!
    - change from micro-coded engine to risc core!
    - 10x registers introduced (internal ones)
  - Limits what can be expressed
    - no "move"s
    - what else?

One view on compiler/Arch Divide

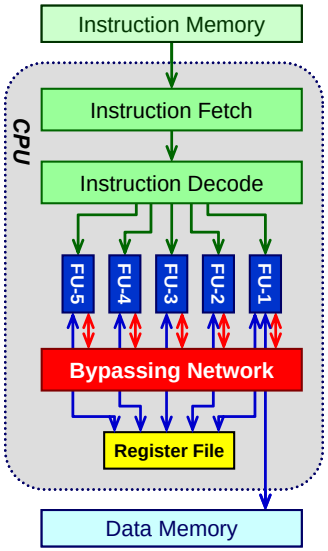


VLIW



- Scaling Drawbacks?

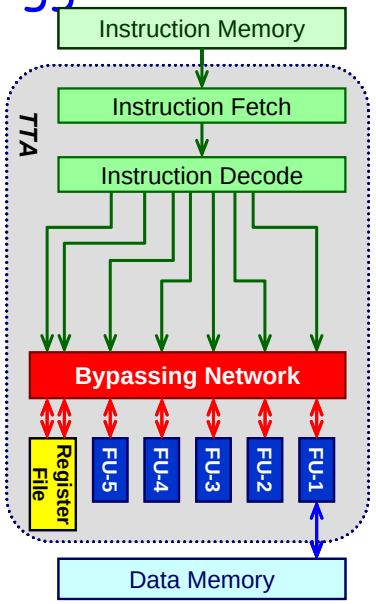
VLIW



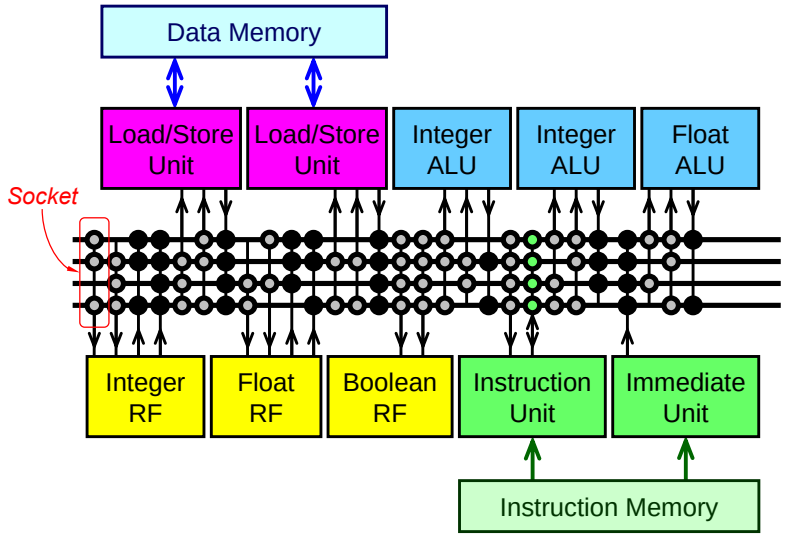
- Scaling Drawbacks?
  - Bypass complexity
  - Register file complexity
  - Register file design restricts FU flexibility
  - Operation encoding format restricts FU flexibility

Transport-Triggered Arch

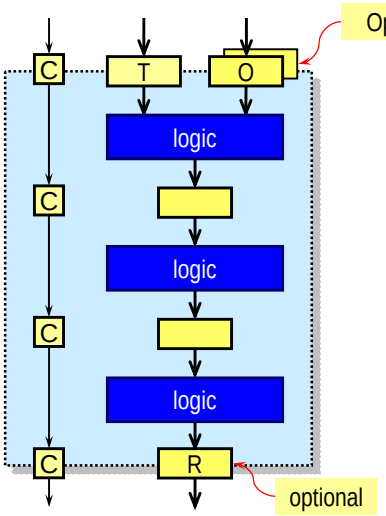
- Only 1 instruction: MOVE
- Don't specify operations, specify register mov't



TTA Datapath

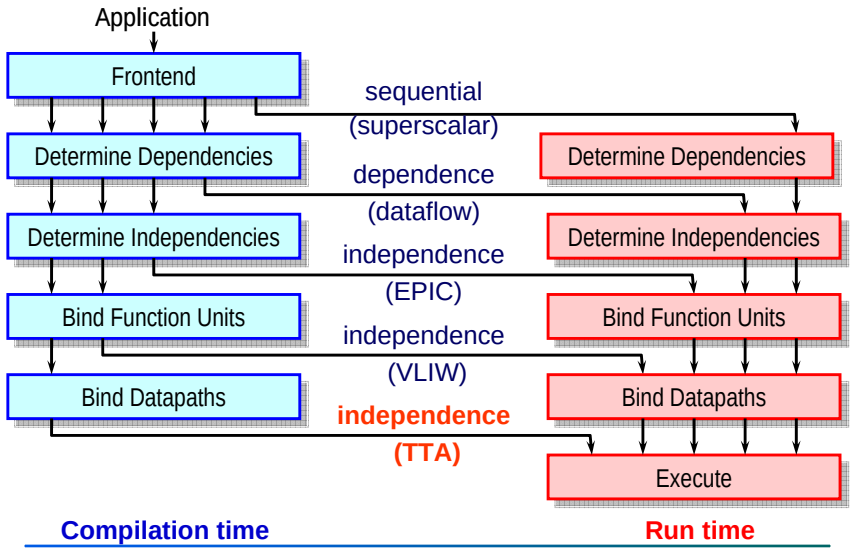


Function Units



- Operands written to operand registers (O)
- Operation performed when last operand written to trigger register (T)
- Pipeline synchronized with control bits (C)
- Standard interface
  - FU_ready
  - Result_ready
  - Global_lock

ILP Architectures



TTA Characteristics: HW

- Modular
  - Can be constructed with standard building blocks
- Very flexible and scalable
  - FU functionality can be arbitrary
  - Supports user defined Special Function Units (SFU)
- Lower complexity
  - Reduction on # register ports
  - Reduced bypass complexity
  - Reduction in bypass connectivity
  - Reduced register pressure
  - Trivial decoding (implies long instructions)

TTA Characteristics: SW

- Traditional operation-triggered instruction:

```
mul r1,r2,r3;
```

- Transport-triggered instruction:

```
r1→mul.o;
r2→mul.t;
mul.r→r3;
```

or

```
r1→mul.o, r2→mul.t;
mul.r→r3;
```

- Reminds dataflow and time-stationary coding

TTA Specific Optimizations

- TTA allows extra scheduling optimizations
- E.g., software bypassing
  - Bypassing can eliminate the need of RF access

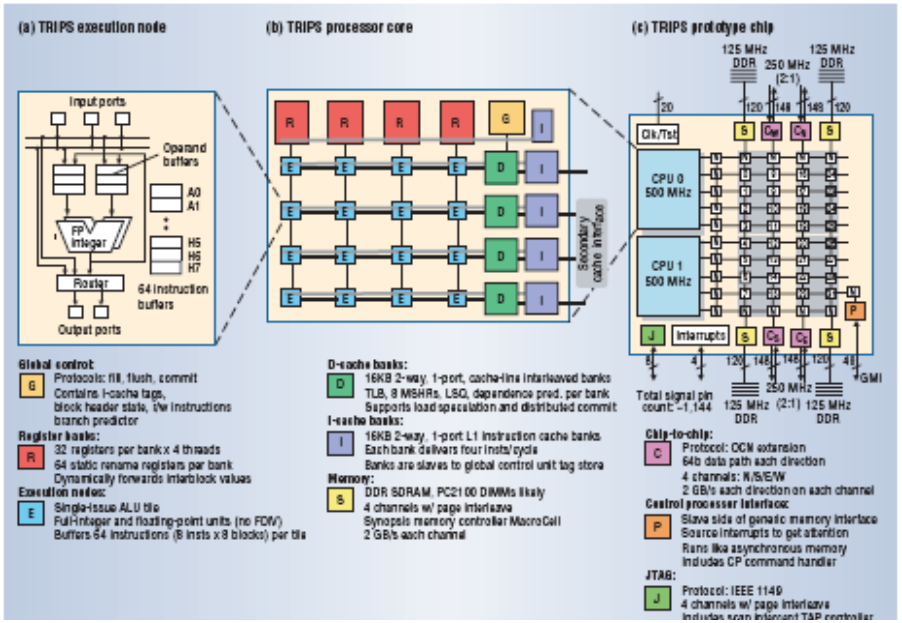
Example:	*r1 → add.o, r2 → add.t;* *add.r → **r3**;* ***r3** → sub.o, r4 → sub.t* *sub.r → r5;*
Translates to:	*r1 → add.o, r2 → add.t;* ***add.r → sub.o**, r4 → sub.t;* *sub.r → r5;*

- However, more difficult to schedule !

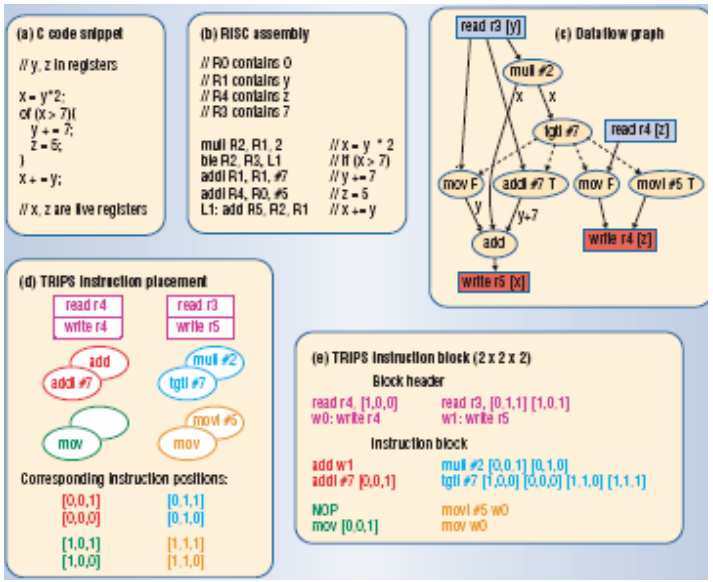
Registers aren't everything

- TRIPS
  - operand-based dataflow architecture
- Wavescalar
  - (operand-based?) dataflow architecture
  - Makes memory dependencies explicit
- Pegasus
  - dataflow, operand/wires explicit
  - Memory dependencies explicit
- All Three
  - basic unit is a hyperblock

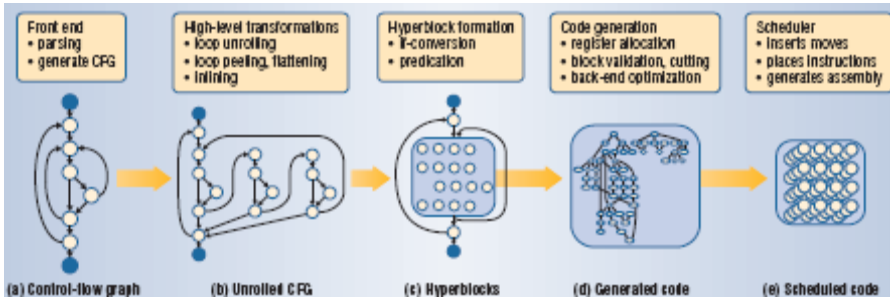
TRIPS



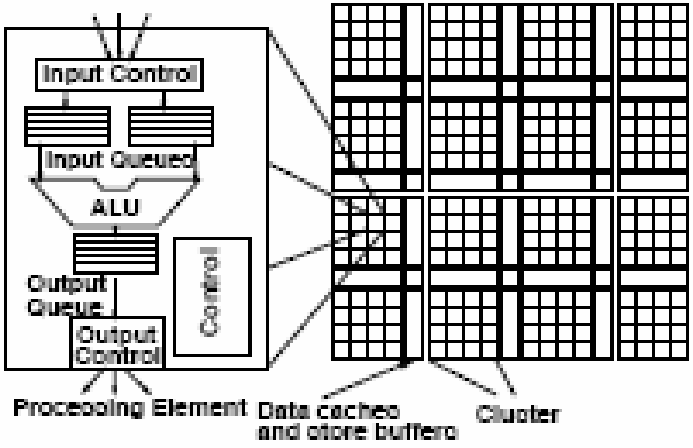
TRIPS: Program Representation



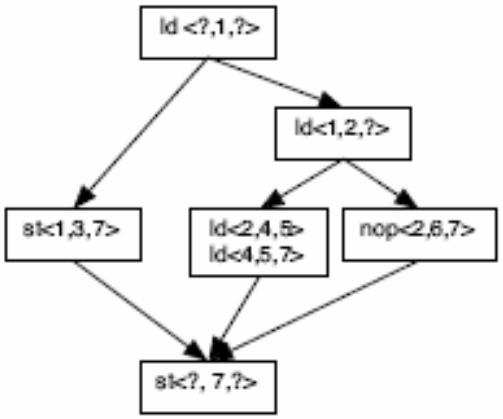
TRIPS: Compiling



Wavescalar



Wavescalar: Memory Dependencies



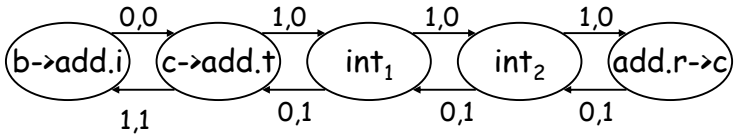
SP on TTA

- Extends LAM's modular scheduling to TTA
- Recall:
  - d(u,v): intra-iter delay between u and v
  - p(u,v): inter-iter distance between u and v
  - $\sigma(v)-\sigma(u) \geq d(u,v)-s*p(u,v)$
  - Find min S, s.t.
    - $\forall(u,v) \in E, \sigma(v)-\sigma(u) \geq d(u,v)-s*p(u,v)$
    - $\forall(t), \sum r(i,t) < R(i)$

Changes to algorithm

- Introduce lower and upper bound for each op
- New priority metric for list scheduler:

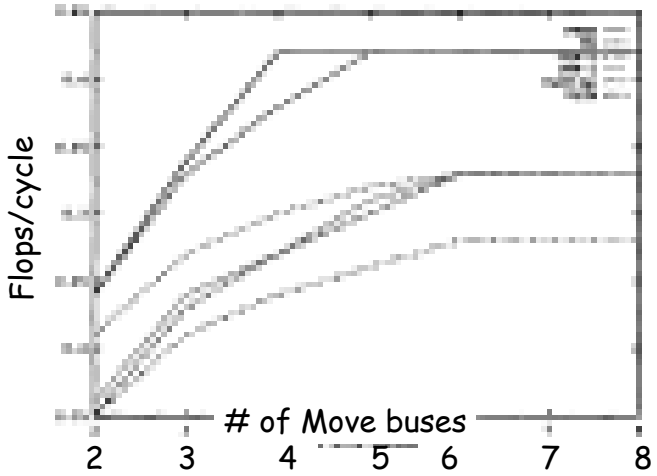
  Priority(v) = $\alpha \frac{r(v)}{s} - \beta(u(v) - l(v))$
- add edges so graph is SC
- Include all RAW and WAR dependencies
- Explicitly model pipelines in FUs



Additional freedom?

- Compare SP for various constraints
  - VLIW
  - VLIW w/software bypassing
  - OTR-1, OTR-2 (do approriate opts for above)
  - operand freedom
  - operand and result freedom

Results



Discussion

- Does TTA address the scaling problem?
- What other opts may be possible?
- What other ways are there to overcome limited registers and ports?
- What about cache misses?
- Are EDGE/WAVESCALAR/CASH decendants of TTA?

Mechanics of slides

- Provide focus
- Reduce distraction

- Make legible
- Label axis, describe graph (axis, top-level bit)

Presentation Mechanics

- 30 minutes with questions
- promote (provoke?) discussion
- Assume your classmates have read the primary paper
- Try and develop a thesis/viewpoint