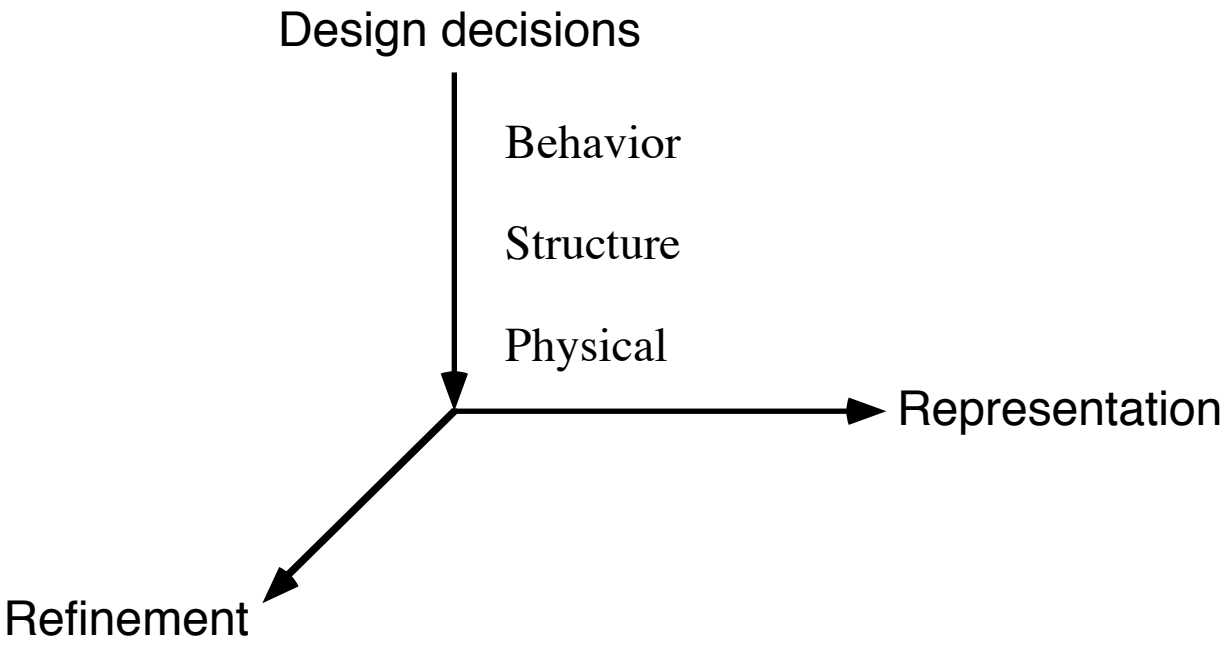
Computer design

Datapath synthesis and design style

P.J. Drongowski
SandSoftwareSound.net

Model of the design process



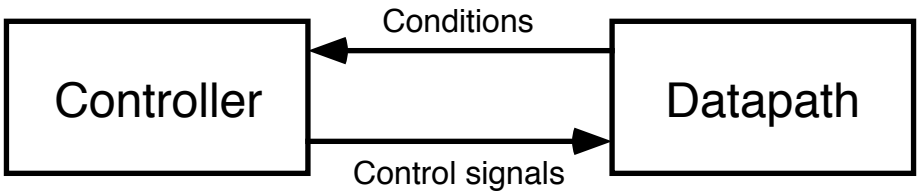
- Representation
  - Abstraction of design objects
  - Granularity
    - Large grain - design in the large
    - Small grain - detailed design
  - Levels
    - Architecture (abstract machine)
    - Organization (control plus datapath)
    - Logic / switch (gates and storage elements)
    - Electrical (electronic behavior)
    - Geometric (physical layout and properties)
- Design decisions
  - "Concurrent engineering" or "co-design"
  - Behavioral
  - Structural
  - Physical
- Refinement
  - Hierarchical design
  - Top-down decomposition into smaller subproblems
  - Bottom-up composition into larger subsystems

Design synthesis

- Transform specification to organization level design
  - Building blocks and connections
  - Control sequencing
- Satisfy engineering constraints and maintain correctness
  - Product characteristics
    - Speed
    - Space (packaging, appearance, etc.)
    - Current consumption (power supply size)
    - Power dissipation (packaging and cooling)
    - Reliability
    - Noise (immunity and radiation)
  - Technical feasibility
  - Market window versus project schedule
  - Price/performance point
    - Non-recurring engineering costs
    - Production costs
    - Promotion, marketing, advertising and administration
    - Profit
- Multidimensional problem
  - Determine datapath (building blocks and connections)
  - Define control and sequencing
    - Schedule computations for execution
    - Assign resources for computation
    - Adjust schedule and datapath for best mix

Overall design style

- Central, synchronous control
- Two-phase, non-overlapping clock
- Separate control and datapath subsystems
  - Condition signals
    - ◊ Datapath state to be sensed
    - ◊ Controller will react to these values
    - ◊ List of values affecting control flow (sequencing)
  - Control signals
    - ◊ Evoke operations and transfers in the datapath
    - ◊ Effectively a list of all building block control inputs



- Microprogrammed control
  - Each control step is a microinstruction
  - A microinstructions is a bundle of control values
- Levels of virtual (abstract) machines

Diagnostic program
ISA emulator (microcode)
Microarchitecture (datapath + control)
Datapath + control simulator
Host machine (VAX/Unix etc.)

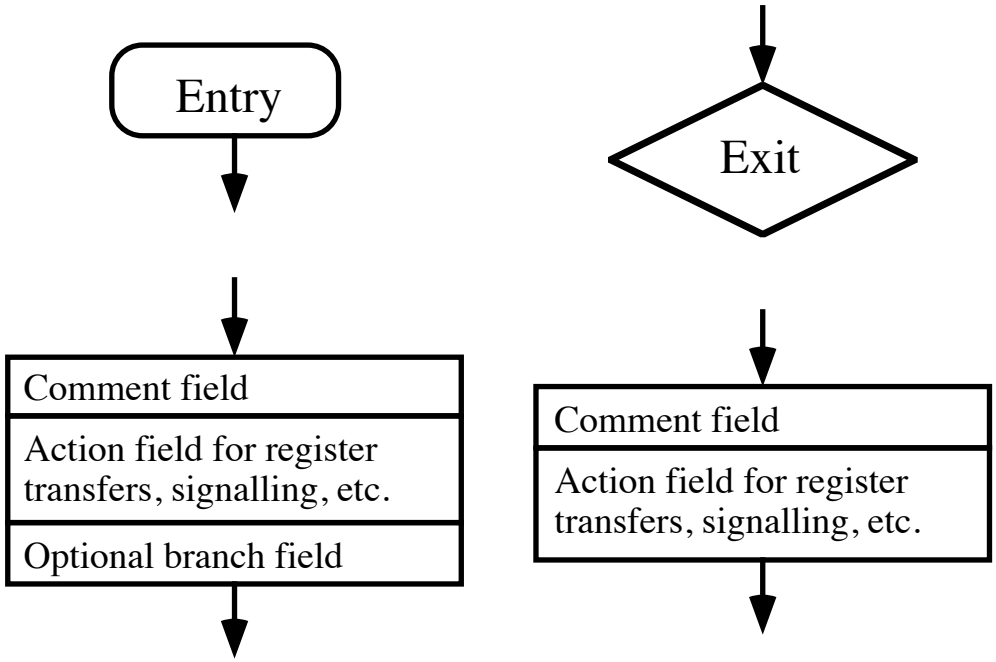
Ad hoc synthesis technique

- Choose a bussing style
  - 0 bus: Data dependency driven connections
  - 2 bus: One read, one write bus
  - 3 bus: Two read, one write bus
- Translate C code into behavior graph
  - Use graph schemas for control constructs
  - Choose RT scheduling policy for initial cut
    - Maximally serial
    - Maximally parallel
  - Schedule and draw graphs for basic blocks
- Iteratively adjust datapath or control graph
  - More parallelism, additional components / busses
    - Select RT's to be merged into one control step
    - Is datapath adequate?
    - Add "private" operators or busses if necessary
    - Merge control steps
  - Fewer components / busses, more serial flow
    - Find control step to be split into two or more steps
    - Remove unneeded components or busses
    - Remove only if unneeded globally WRT control graph
- Adjust datapath delay, space and power mix
  - Faster delay path
    - Choose or design faster components
    - Collapse levels of multiplexing
    - Avoid or eliminate long busses
    - Decrease distance between critical components
  - Use less space
    - Try to remove components / busses from datapath
    - Select smaller components (possible speed penalty)
  - Dissipate less heat, lower current
    - Use efficient components on non-critical paths
    - Slow down critical path, increase parallelism

Behavior

- What is the behavior of the system?
- Organization level is sequencing plus register transfers
  - Sequencing represented in control graph
  - Register transfer
    - Parallel assignment statement
    - Transfer from sources to destinations via operators
  - Sources and destinations are registers, ports, memory
- Does the behavior execute fast enough?
  - Interacts heavily with structural and physical decisions
  - Increase parallelism with more components / busses
  - Combine or split control steps
  - Choose faster (higher power) components
  - Decrease (interconnect) bus lengths
  - Assign communicating subsystems to same chip

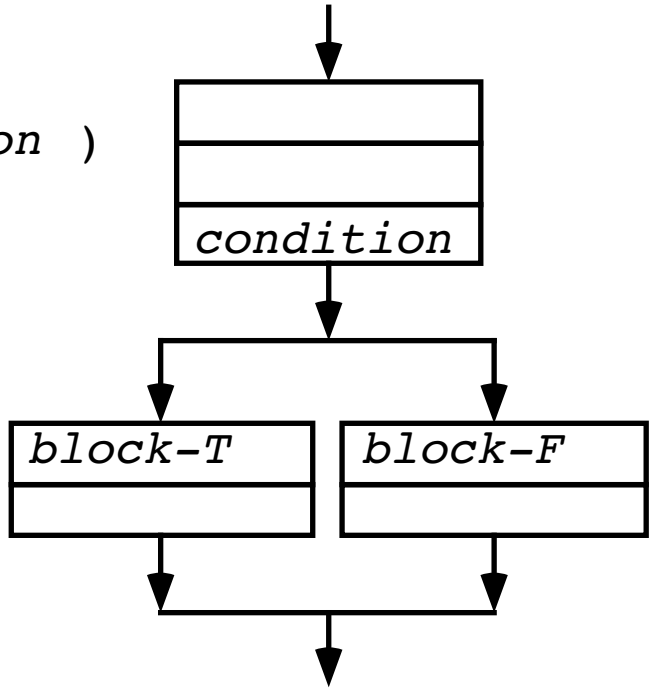
Control graph notation



Control graph schemas

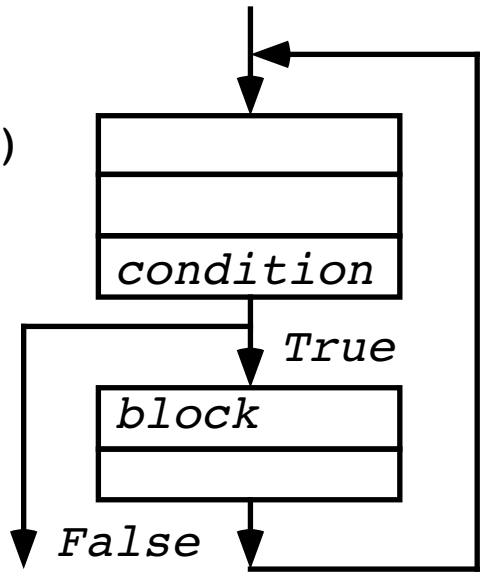
- Mapping from HLL control statement to graph
- Similar correspondence as source code to flowchart

```
if ( condition )
   {
   block-T
   }
else {
   block-F
   }
```



- Conditions must be resolved into real signals
- Ok to have multi-way case branches
- Conditions can be AND'ed to decrease levels of branching

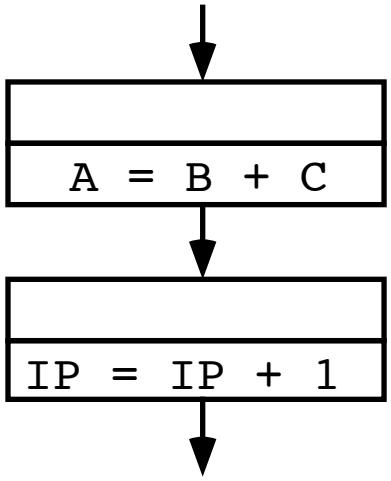
```
while ( condition )
   {
   block
   }
```



Initial schedule

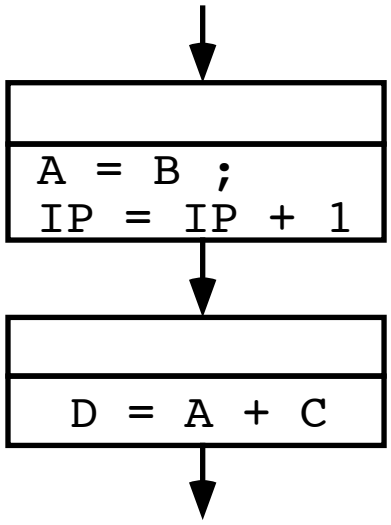
- Maximally serial
  - Assign each transfer to a single control step
  - Compatible with sequential, procedural HLL
  - Will have largest number of control steps
  - Fits well with 2- or 3-bus style
  - Steps may require splitting due to datapath restrictions

```
A = B + C ;
IP = IP + 1 ;
```



- Maximally parallel
  - Assign as many transfers to one step as possible
  - Transfer may be re-ordered if independent
  - Data dependency due to side-effect needs an extra step
  - Datapath may not have enough parallelism

```
A = B ;
D = A + C ;
IP = IP + 1 ;
```



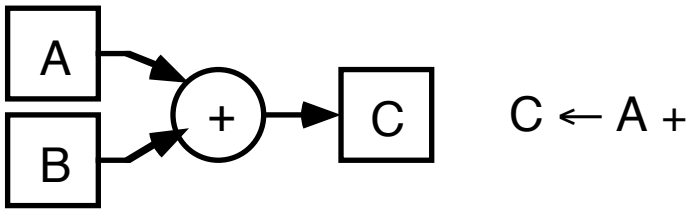
- Could be combined to one step
- Action: `A = B ; D = B + C ; IP = IP + 1`

Structure

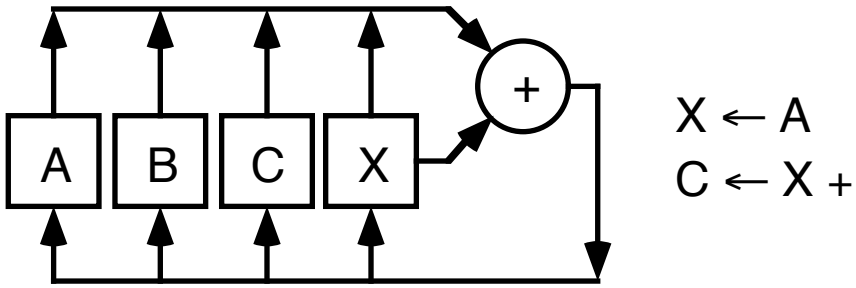
- What are the control, storage and operator resources?
- Structure is drawn in a block diagram
- How are the resources interconnected?
  - 0-bus style: Data dependency driven
  - 2-bus style: Read /write busses with common resources
  - 3-bus style: 2-read, 1-write bus with common resource
- How much space is consumed?
  - Delete component / bus ⇒ reschedule control graph
  - Reduce number of off-chip connections
  - Reduce power / speed ⇒ smaller size / drive transistors
- How much current is required (grid sizing?)
  - Reduce transistor size (drive) ⇒ lower transfer speed
  - High current ⇒ wide power distribution rails
- What is the power dissipation (cooling?)
  - Reduce current / speed of non-critical paths
  - Decrease number of off-chip connections (esp. outputs)
- Control
  - Must provide control component (FSM, etc.)
  - Establish interconnection for:
    - Datapath conditions to controller
    - Control signals from controller to datapath
  - Translate control graph to control "programming"

Bus styles

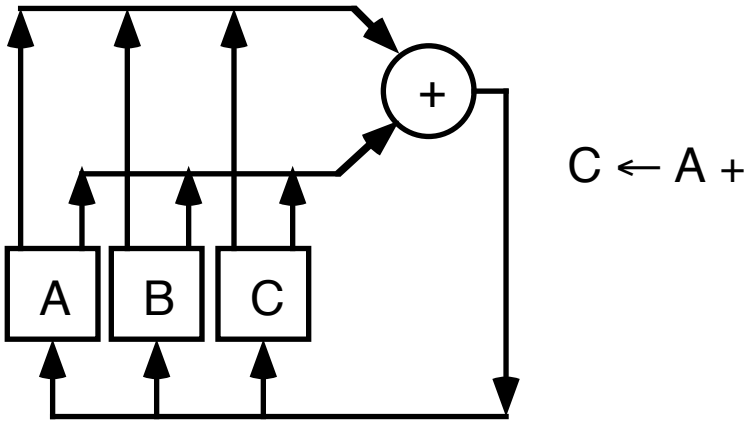
- 0 bus - data dependency driven design
  - Will probably yield the most parallelism
  - Some parallelism may go unused or be too costly
  - Build list of all data dependencies
  - Allocate operator or bus as needed
  - Example abstract transfer: C ← A + B



- 2 bus - one read, one write bus
  - Highly sequential RT scheduling
  - Conservative use of busses; easy to layout
  - Need temporary register for dyadic operations

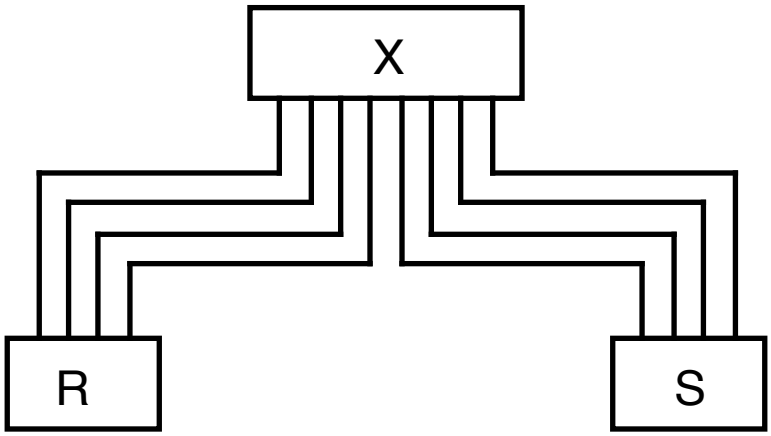


- 3 bus - two read, one write bus
  - One step evaluation of dyadic operation
  - Good fit with dual port register file, ALU



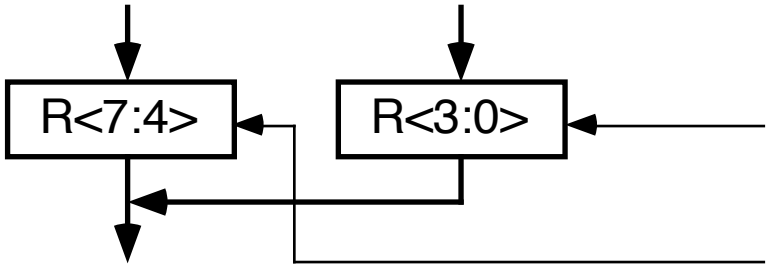
Busses

- Bus split (or rip)
  - Selectively route wires from source to destination
  - What is the datatype of the result? Unsigned?
  - Example: `R = X<7:4> ; S = X<3:0>`



- Bus join (or concatenate)
  - Route wires to different bits of common destination
  - Assign transfers to same machine step if possible

- Source and destination of different word sizes (length)
  - Assume `R` is N bits, `S` is M bits
  - N > M: `R = S ;`
    - Assign `S` to `R`<M-1:0>
    - Set `R`<N:M> to 0?
    - Leave `R`<N:M> unaffected?
  - N < M: `R = S ;`
    - Assign `S`<N-1:0> to `R`
    - Bits `S`<M:N> are discarded (unconnected)
  - May need to clock portions of register independently



Variables

- Kinds of variables
  - Communication ports
    - Data ports (in, out, bidirectional)
    - Control ports (signalling, synchronization)
  - Architectural registers or memory
    - Part of the visible interface or ISA
    - Must be included in design
  - Temporary registers or memory
    - Hidden storage
    - Can be flexibly scheduled (shared)

- Implementation styles
  - Dedicated register
    - Can add intrinsic operations
    - More exploitable datapath parallelism
  - Register file (scratchpad RAM)
    - No special intrinsic operations
    - Sequential access to variables
  - Large, read / write memory
    - Can hold a large number of variables
    - Usually requires off-chip memory subsystem

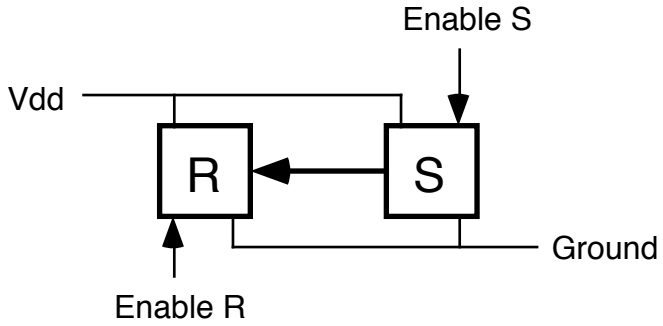
- Live variable analysis
  - Determine variable lifetime
  - Construct register compatibility graph
    - Find set of disjoint registers for each register
    - Edge represents disjoint usage times of 2 registers
  - Merge registers with disjoint lifetimes
  - Good especially for temporary registers
  - Algorithms
    - Clique partitioning (NP-complete)
    - Graph coloring

Register transfers

- Each RT has implications for:
  - ◊ Binding of variables to storage or ports
  - ◊ Selection of and binding to operator components
  - ◊ Interconnection and possibly multiplexing
  - ◊ Routing of data through the datapath
- Scheduling of RT's can affect:
  - ◊ Variable and operator bindings
  - ◊ Availability of routes through datatpath
- Heuristics
  - Serial execution permits more sharing
  - Parallel execution needs more dedicated resources
  - Parallel execution may cause resource conflicts

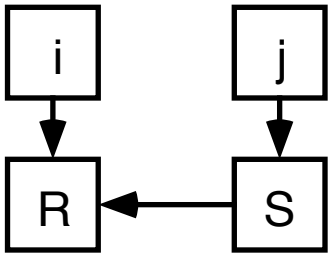
Simple register transfer

- Transfer from one register to another: `R = S ;`
- Implications
  - Two storage objects for `S` and `R`
  - A path from `S` to `R`
  - Connect LSB's to LSB's: `R<N:0> = S<M:0> ;`
  - Enable (control) signals routed to controller
  - Power connections must be made to rails
- Dead variable analysis for temporary binding to register

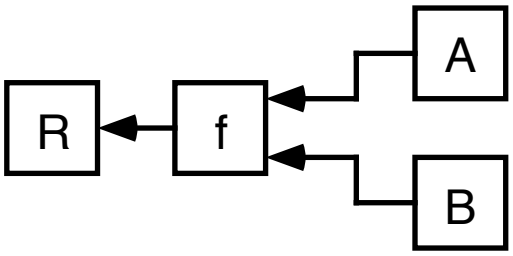


Complex register transfers

- Register indices must be computed
- Choices
  - Produce index from controller (μword)
  - Generate with dedicated hardware
  - Route from register field (e.g., `IP`)
  - Compute with shared hardware
- Example: `R[ i ] = S[ j ] ;`
- `R` and `S` must be register files (scratchpad



- Function call on right hand side (RHS)
- Does the function have side-effects?
  - Yes
    - Implies a parallel control (execution) flow, or
    - A serial "subroutine" to be called / expanded in-line
    - Must be scheduled WRT control graph
    - Difficult problem; disallow side-effects
  - No
    - Can be implemented as combinational logic
    - Decompose function into library components
    - Generate PLA for logic function
- Example: `R = f(A, B) ;`



General registers

- Register to register architecture

  $R[i] \leftarrow R[j] \ \Delta \ R[k]$

- Single port RAM implementation
  - Only one value can be read at a time
  - Sequential computation needed

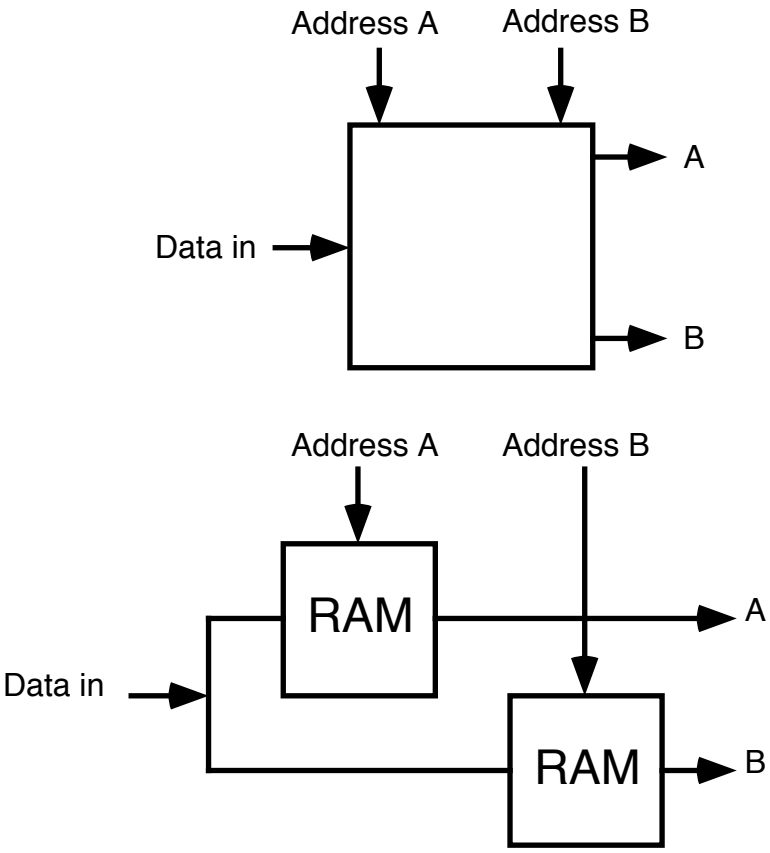
    $X \leftarrow R[j]$

    $Y \leftarrow Y \ \Delta \ R[k]$

    $R[i] \leftarrow Y$
- Dual port RAM implementation
  - Two read ports; two values out at a time
  - Reduce to two step evaluation

    $Y \leftarrow R[j] \ \Delta \ R[k]$

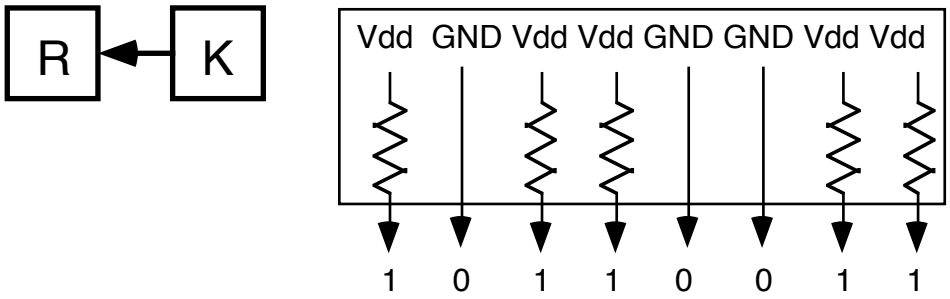
    $R[i] \leftarrow Y$



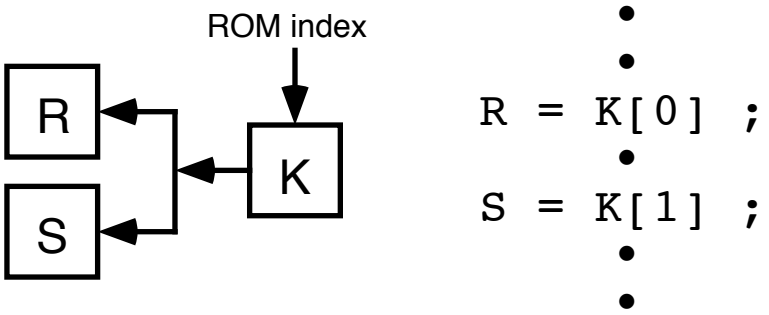
Constants

- Constants must be stored somewhere!

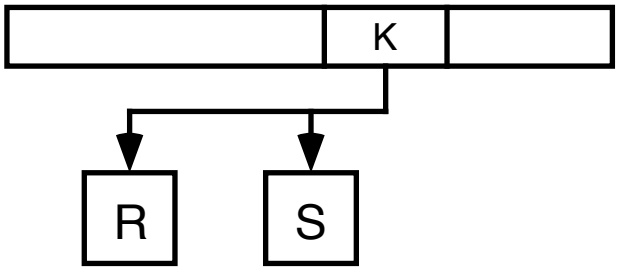
- Single constant
  - Connect outputs to Vdd or ground
  - Use current limiting resistors



- Read only memory (ROM) implementation
  - Useful if several constants are needed
  - Generate index from controller (μword) or datapath

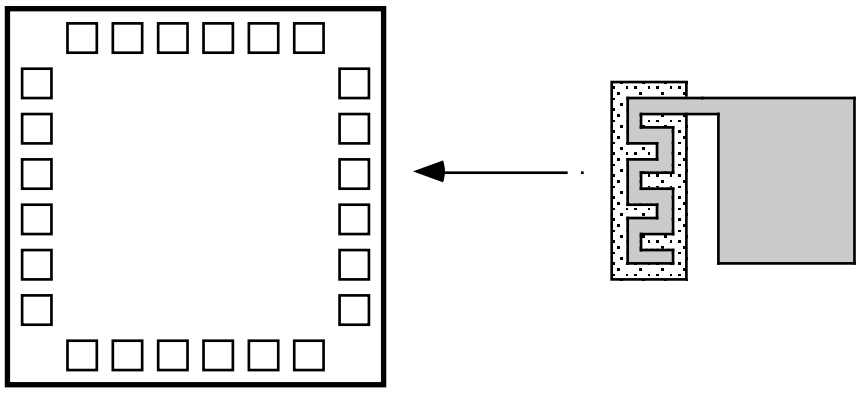


- Microinstruction "emit field"
  - Control program is often stored in ROM
  - Design field into microword to emit constant
  - Bits are allocated even if unused (disadvantage)
  - Access to only one constant at a time



External input / output

- Chip must be connected to printed circuit board
  - Die is mounted in a carrier package
  - Carrier has traces leading to pins
  - Pins connect to PCB-level pads and wires
  - Wires must be bonded to chip and carrier traces
  - Chip connections are made to metal "bonding pads"
  - Pads must be placed at perimeter of die
- Typical component sizes (1 μm = 1 micron)
  - Transistor: 1 μm × 1 μm
  - Metal wire: 3 μm wide
  - Bonding pad: 100 μm × 100 μm
  - Human hair: 40 μm diameter
- Input
  - Voltage level shifting or hysteresis
  - Static protection
- Output
  - Must drive large off-chip wires (10+ pF capacitance)
  - Drive transistor: 80 μm × 1 μm
  - Large current requirement and power dissipation
- Bidirectional
  - Permits bidirectional flow of data
  - Must have a disconnected (high-Z) state
- Power supply (e.g., Vdd and ground)

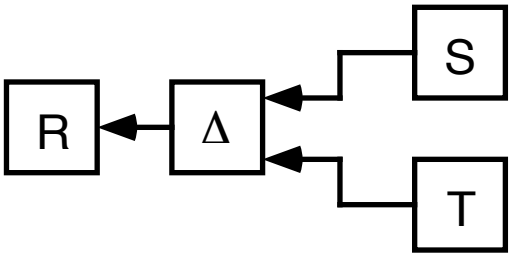


Dedicated versus general

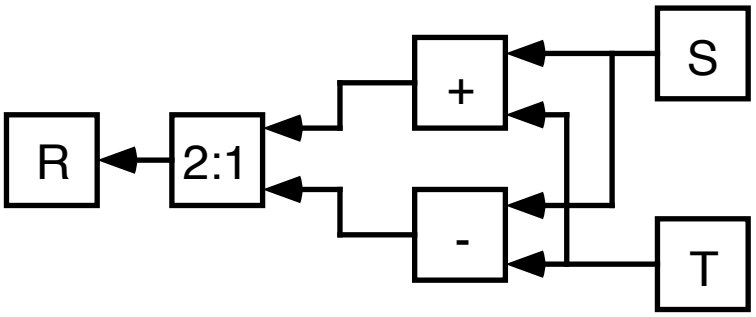
- Trade-off is not simple or obvious
- Consider the example below

```
        •
        •
R = S + T ;
        •
        •
R = S - T ;
        •
        •
```

- + and – could be bound to an



- Or, two dedicated operators could be allocated
- Note extra delay due to multiplexing
- Routing complexity is potentially higher, too



- Which solution requires less area?
- Dissipates less heat?
- How do wire lengths affect delay?

Data operations

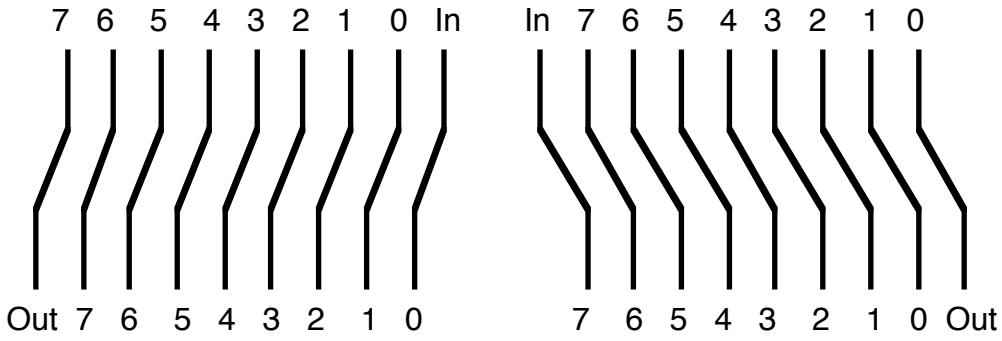
- Intrinsic operations
  - Operations that can be performed in place
  - Avoid additional combinational logic and wiring
  - Choose library component with operations built-in
  - Shift and count not usually implemented together
  - Must add extra control lines

Transfer	Name	Component
`R = 0`	Clear	Register
`R = 0xF••F`	Preset	Register
`R = R + 1`	Increment	Counter
`R = R – 1`	Decrement	Counter
`R = R / 2`	Shift right	Shift register
`R = R * 2`	Shift left	Shift register

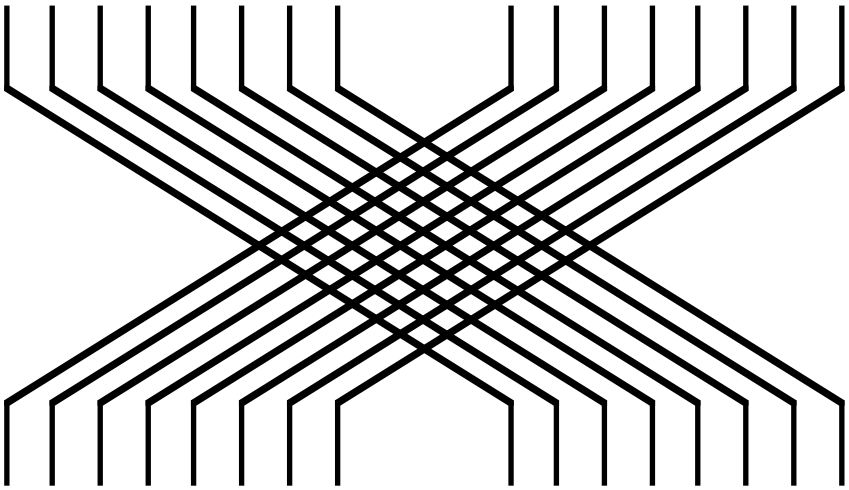
- Dedicated operator component
  - Map operator directly to combinational component
  - Should yield most parallelism
  - Costly in space, current, power, interconnect
  - Many stages of logic will slow clock speed
    - ◊ Longer, fewer steps versus
    - ◊ More steps of shorter duration
  - Exploit commutivity where ever possible
  - Non-commutative operators constrain interconnection

- "Generalized" operator component
  - Typical example is an arithmetic logic unit (ALU)
  - Good choice in serial flow with opportunity for sharing
  - May force long wires to be routed to central resource
    - ◊ Wire delay could dominate switching delay
    - ◊ Selectively employ dedicated components
  - See comments on commutivity above

Left and right shift

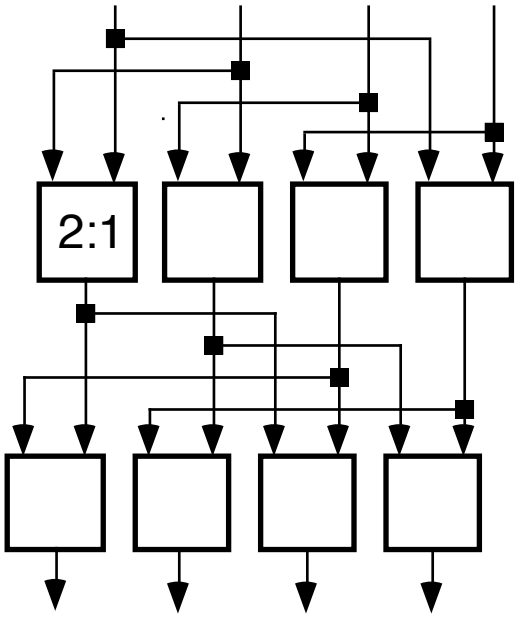


Byte swap



Barrel switch

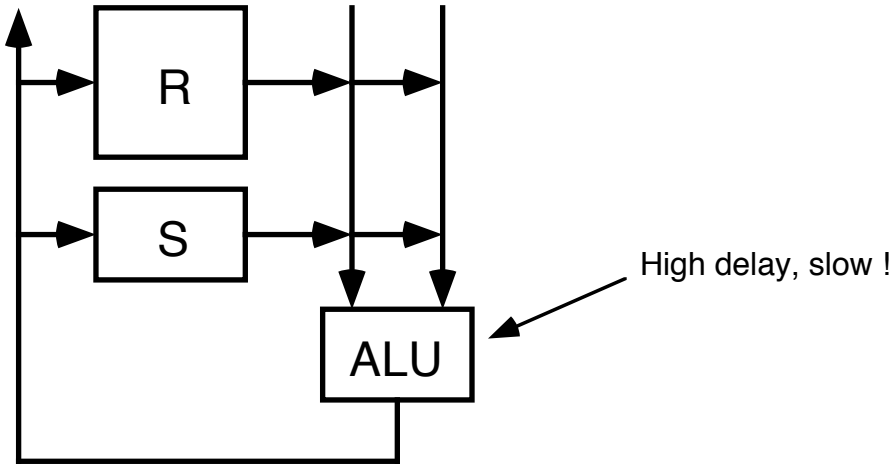
- Rotate multiple places left or right
- Mask to obtain shifted data



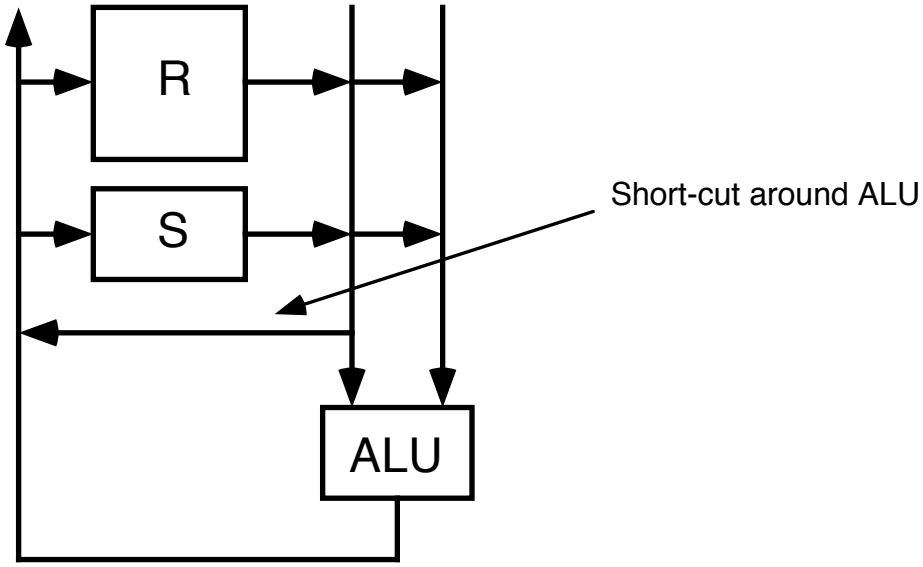
- Rotate one place
- Rotate two places

Why wait?

- Common three bus structure



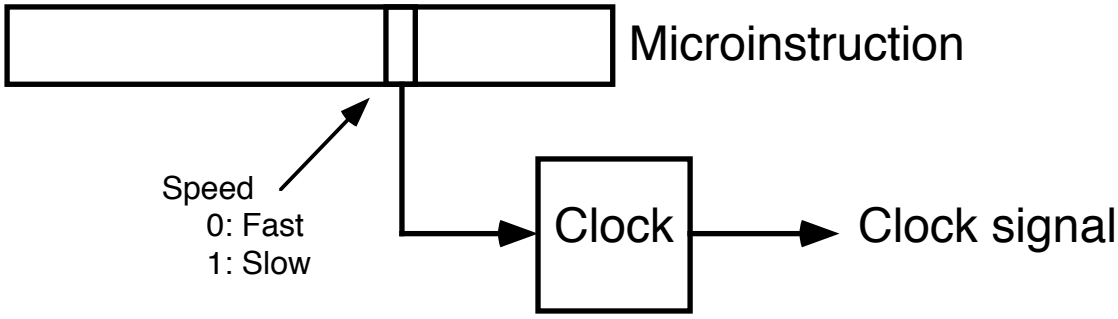
- Bottleneck
  - ALU adds delay to simple transfers: S ← R¨



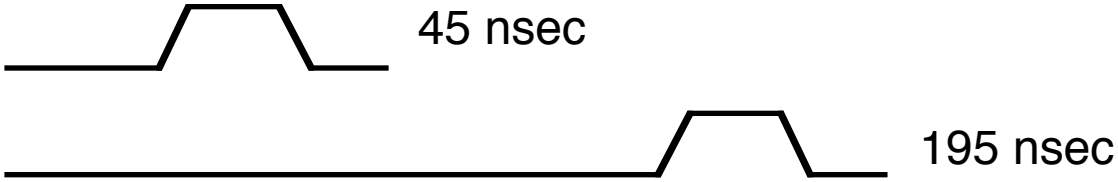
- But! Clock period is determined by slowest component.
- Make clock rate variable instead.

Programmable clock

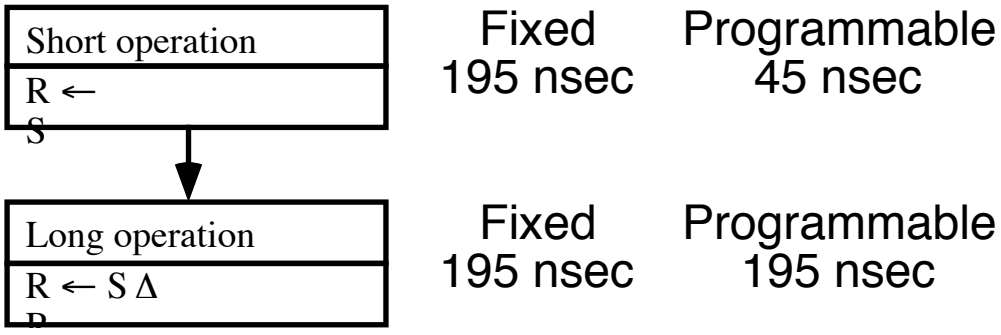
- Performance limited by slowest component
- Make clock speed selectable
- Choose the fastest speed possible for a transfer



- Two or more speeds



- Fixed speed versus programmable



Physical

- Where should components be placed?
- Size and arrangement are shown in floor plan
- Increase speed of critical register transfer
  - Move components on path closer
  - Add private bus instead of using long bus
  - Increase drive (power) on critical path
- Reduce component area
  - Eliminate component ⇒ reschedule control graph
  - Use smaller drive transistors ⇒ slower transfers
- Reduce interconnect area
  - Use wiring by abutment and pitch matching
  - Move components closer together (compact layout)
- Decrease size of power grid
  - Decrease current requirements of subsystems
  - Do not exceed maximum current density of wires
- Reduce off-chip connections
  - Bonding pads are 10,000 times as big as transistors
  - Outputs are chief source of heat (roughly 40 percent)
- Other concerns
  - Noise
  - Quality of the power supplied
  - Clocks, power and control must be routed globally
  - Few layers for wire routing