Notes on SRC Shift RTN

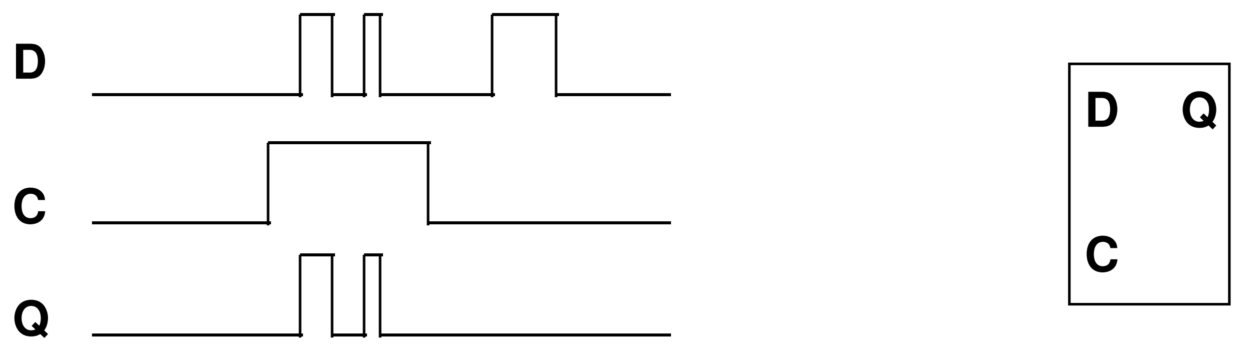
- In the abstract RTN, n is defined with :=
- In the concrete RTN, it is a physical register
- n not only holds the shift count but is used as a counter in step T6
- Step T6 is repeated n times as shown by the recursion in the RTN
- The control for such repeated steps will be treated later

Data Path/Control Unit Separation

- Interface between data path and control consists of gate and strobe signals
- A gate selects one of several values to apply to a common point, say a bus
- A strobe changes the values of the flip-flops in a register to match new inputs
- The type of flip-flop used in registers has much influence on control and some on data path
  - Latch: simpler hardware, but more complex timing
  - Edge triggering: simpler timing, but about twice the hardware

Reminder on Latch- and Edge-Triggered Operation

- Latch output follows input while strobe is high



- Edge-triggering samples input at edge time

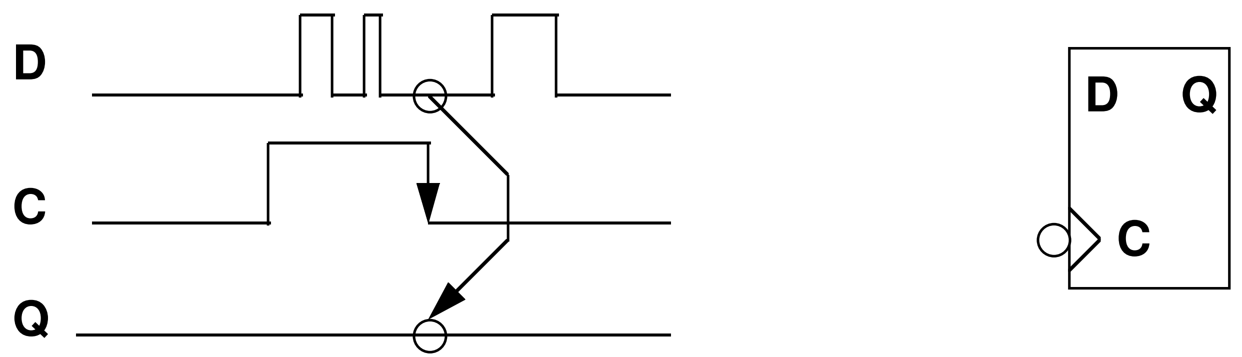
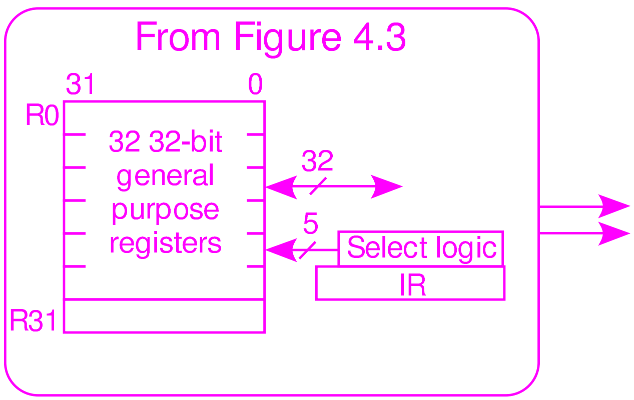
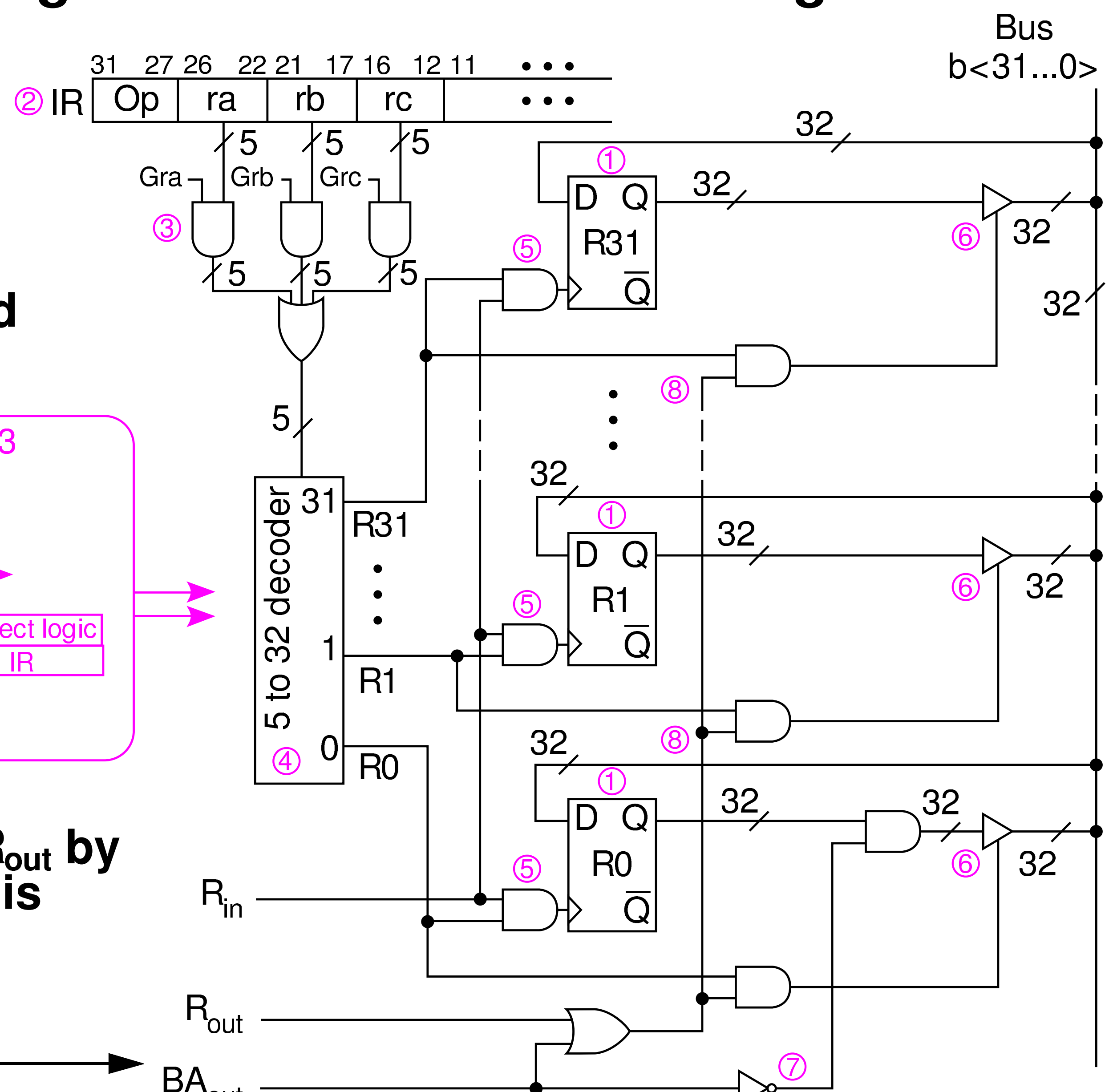


Fig 4.4 The Register File and Its Control Signals

- **R_{out} gates selected register onto bus**
- **R_{in} strobed selected register from bus**



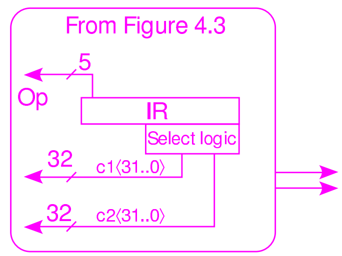


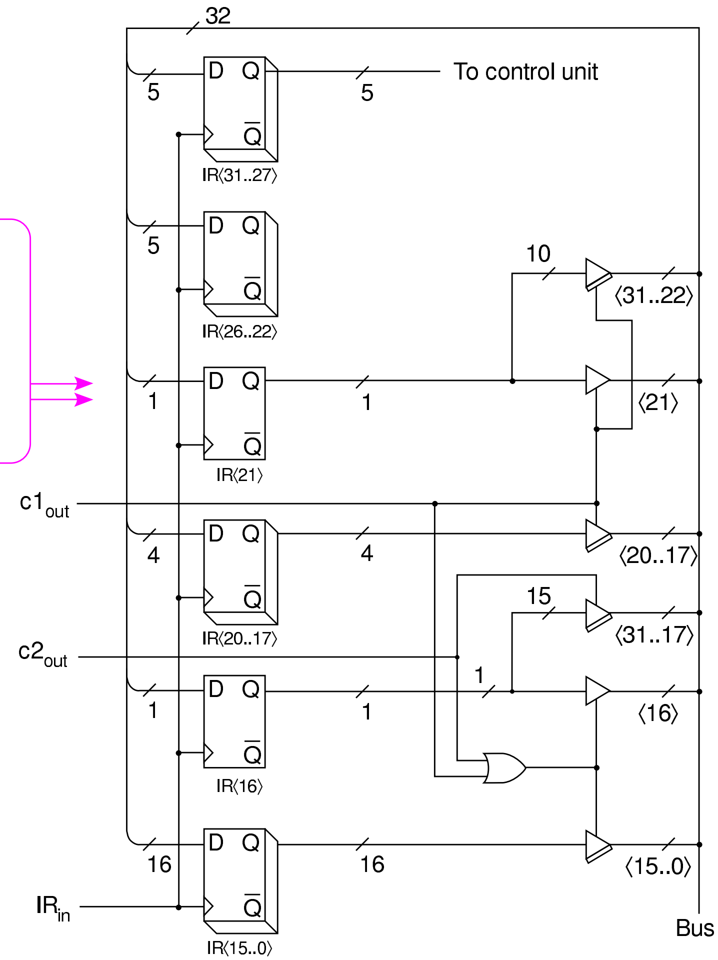
- **BA_{out} differs from R_{out} by gating 0 when R[0] is selected**

BA = Base Address →

Fig 4.5 Extracting c1, c2, and OP from the Instruction Register, IR<31...0>

- **I⟨21⟩ is the sign bit of C1 that must be extended**

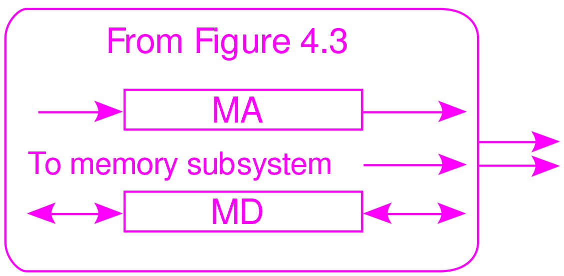


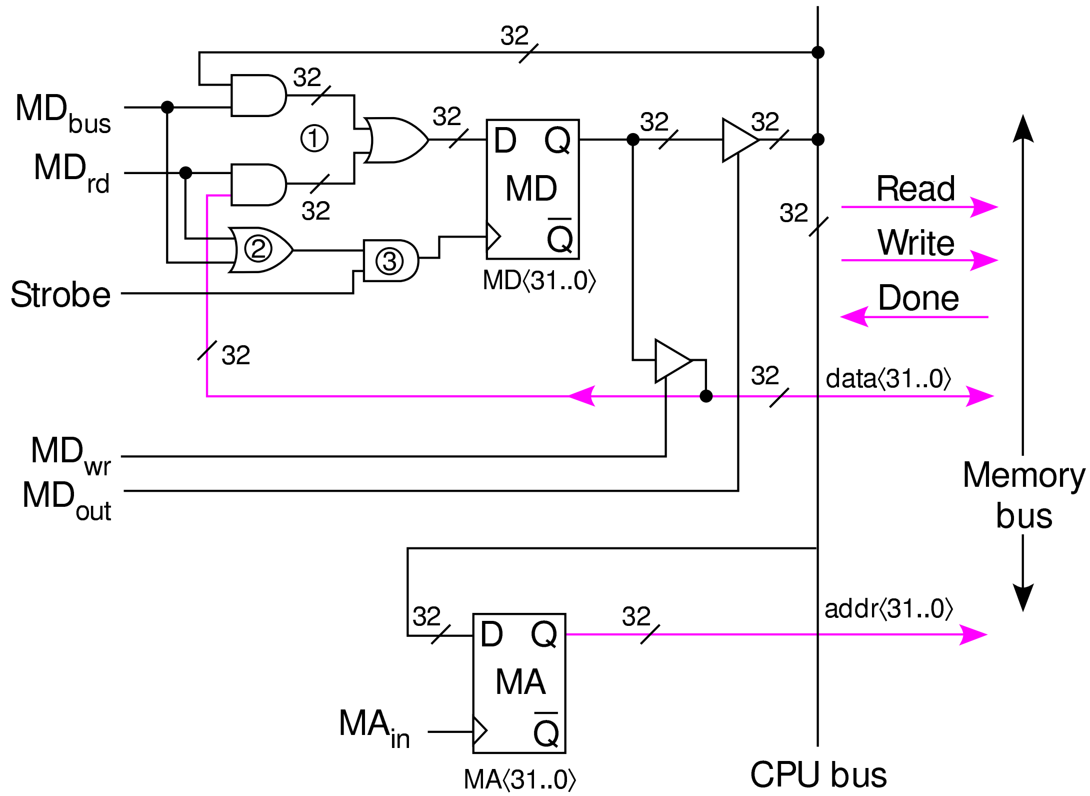


- **I⟨16⟩ is the sign bit of C2 that must be extended**
- **Sign bits are fanned out from one to several bits and gated to bus**

Fig 4.6 The CPU–Memory Interface: Memory Address and Memory Data Registers, MA<31...0> and MD<31...0>

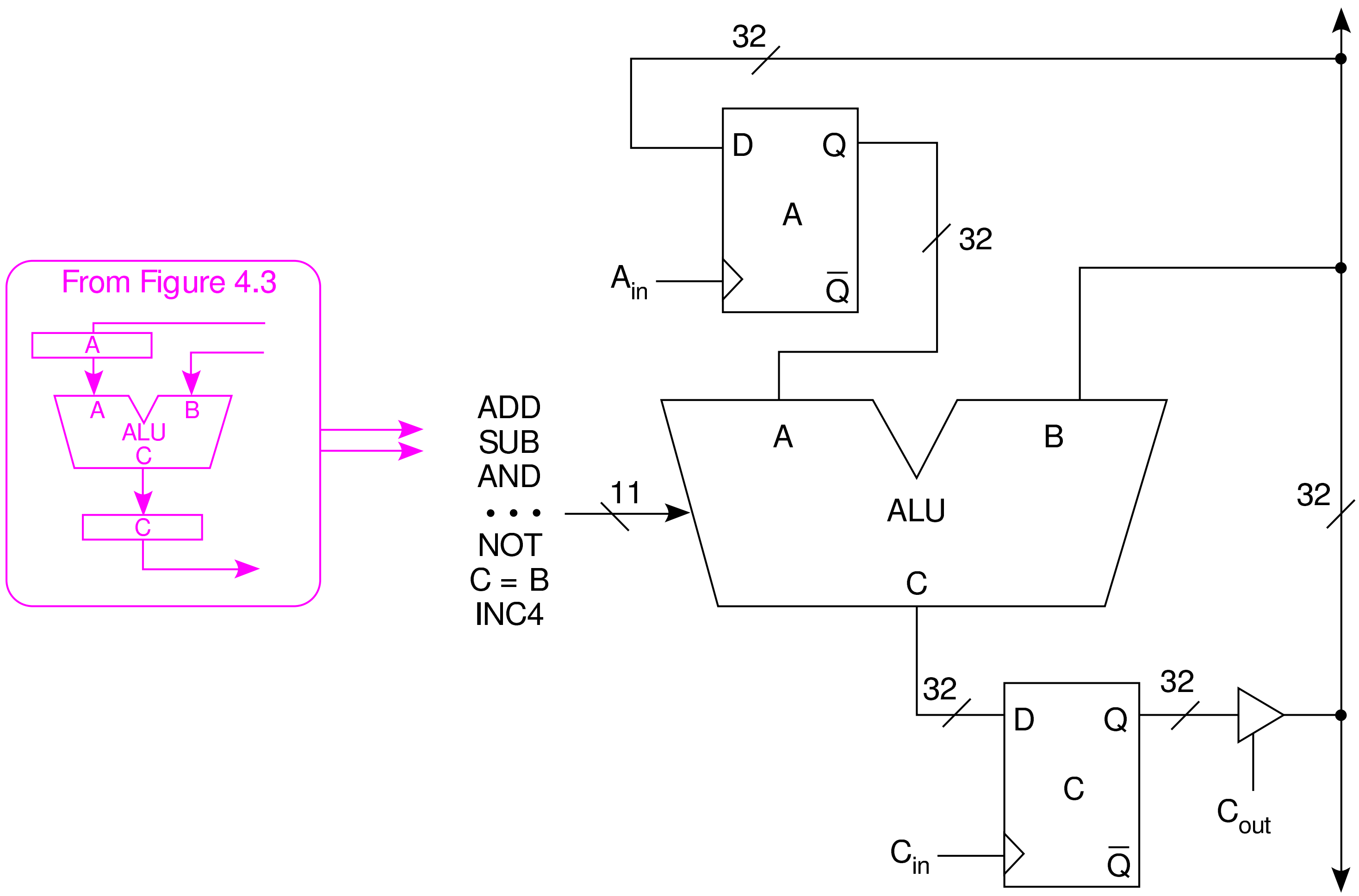
- **MD is loaded from memory or from CPU bus**





- **MD can drive CPU bus or memory bus**

Fig 4.7 The ALU and Its Associated Registers



From Concrete RTN to Control Signals: The Control Sequence

Tbl 4.6 The Instruction Fetch

Step	Concrete RTN	Control Sequence
T0	MA ← PC: C ← PC + 4;	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]: PC ← C;	Read, C_{out}, PC_{in}, Wait
T2	IR ← MD;	MD_{out}, IR_{in}
T3	Instruction_execution	

- The register transfers are the concrete RTN
- The control signals that cause the register transfers make up the control sequence
- Wait prevents the control from advancing to step T3 until the memory asserts Done

Control Steps, Control Signals, and Timing

- Within a given time step, the order in which control signals are written is irrelevant
  - In step T0, C_{in}, Inc4, MA_{in}, PC_{out} == PC_{out}, MA_{in}, INC4, C_{in}
- The only timing distinction within a step is between gates and strobes
- The memory read should be started as early as possible to reduce the wait
- MA must have the right value before being used for the read
- Depending on memory timing, Read could be in T0

Control Sequence for the SRC add Instruction

add (:= op = 12) $\rightarrow$ R[ra] $\leftarrow$ R[rb] + R[rc]:

Tbl 4.7 The add Instruction

Step	Concrete RTN	Control Sequence
T0	MA $\leftarrow$ PC: C $\leftarrow$ PC + 4;	PC_{out}, MA_{in}, INC4, C_{in}, Read
T1	MD $\leftarrow$ M[MA]: PC $\leftarrow$ C;	C_{out}, PC_{in}, Wait
T2	IR $\leftarrow$ MD;	MD_{out}, IR_{in}
T3	A $\leftarrow$ R[rb];	Grb, R_{out}, A_{in}
T4	C $\leftarrow$ A + R[rc];	Grc, R_{out}, ADD, C_{in}
T5	R[ra] $\leftarrow$ C;	C_{out}, Gra, R_{in}, End

- Note the use of Gra, Grb, and Grc to gate the correct 5-bit register select code to the registers
- End signals the control to start over at step T0

Control Sequence for the SRC addi Instruction

addi (:= op= 13) → R[ra] ← R[rb] + c2⟨16..0⟩ {2's comp., sign ext.} :

Tbl 4.8 The addi Instruction

Step	Concrete RTN	Control Sequence
T0.	MA ← PC: C ← PC + 4;	PC_{out}, MA_{in}, Inc4, C_{in}, Read
T1.	MD ← M[MA]; PC ← C;	C_{out}, PC_{in}, Wait
T2.	IR ← MD;	MD_{out}, IR_{in}
T3.	A ← R[rb];	Grb, R_{out}, A_{in}
T4.	C ← A + c2⟨16..0⟩ {sign ext.};	$c2_{out}$, ADD, C_{in}
T5.	R[ra] ← C;	C_{out}, Gra, R_{in}, End

- The $c2_{out}$ signal sign extends IR⟨16..0⟩ and gates it to the bus

Control Sequence for the SRC st Instruction

st (:= op = 3) $\rightarrow$ M[disp] $\leftarrow$ R[ra] :
disp$\langle 31..0 \rangle$:= ((rb=0) $\rightarrow$ c2$\langle 16..0 \rangle$ {sign extend} :
(rb$\neq$0) $\rightarrow$ R[rb] + c2$\langle 16..0 \rangle$ {sign extend, 2's complement}) :

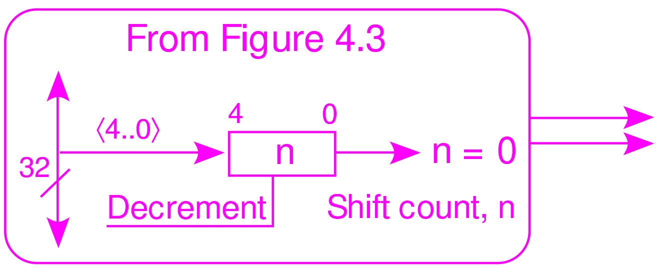
The st Instruction

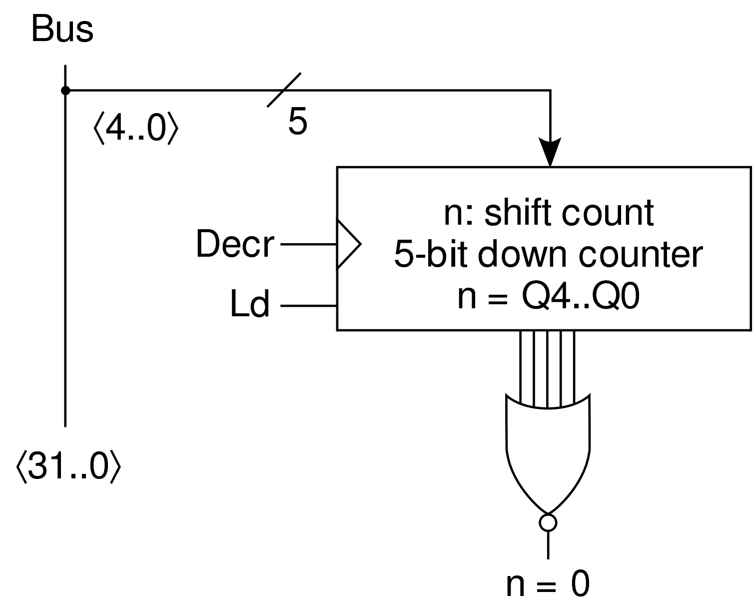
Step	Concrete RTN	Control Sequence
T0–T2	Instruction fetch	Instruction fetch
T3	A $\leftarrow$ (rb=0) $\rightarrow$ 0: rb $\neq$ 0 $\rightarrow$ R[rb];	Grb, BA_{out}, A_{in}
T4	C $\leftarrow$ A + c2$\langle 16..0 \rangle$ {sign-extend};	$c2_{out}$, ADD, C_{in}
T5	MA $\leftarrow$ C;	C_{out}, MA_{in}
T6	MD $\leftarrow$ R[ra];	Gra, R_{out}, MD_{in}, Write
T7	M[MA] $\leftarrow$ MD;	Wait, End

- Note BA_{out} in T3 compared to R_{out} in T3 of addi

Fig 4.8 The Shift Counter

- The concrete RTN for shr relies upon a 5-bit register to hold the shift count
- It must load, decrement, and have an = 0 test





Tbl 4.10 Control Sequence for the SRC shr Instruction—Looping

Step	Concrete RTN	Control Sequence
T0–T2	Instruction fetch	Instruction fetch
T3	n ← IR⟨4..0⟩;	$c1_{out}$, Ld
T4	(n=0) → (n ← R[rc]⟨4..0⟩);	n=0 → (Grc, R_{out}, Ld)
T5	C ← R[rb];	Grb, R_{out}, C=B, C_{in}
T6	Shr (:= (n≠0) → (C⟨31..0⟩ ← 0#C⟨31..1⟩: n ← n-1; Shr));	n≠0 → (C_{out}, SHR, C_{in}, Decr, Goto6)
T7	R[ra] ← C;	C_{out}, Gra, R_{in}, End

- Conditional control signals and repeating a control step are new concepts

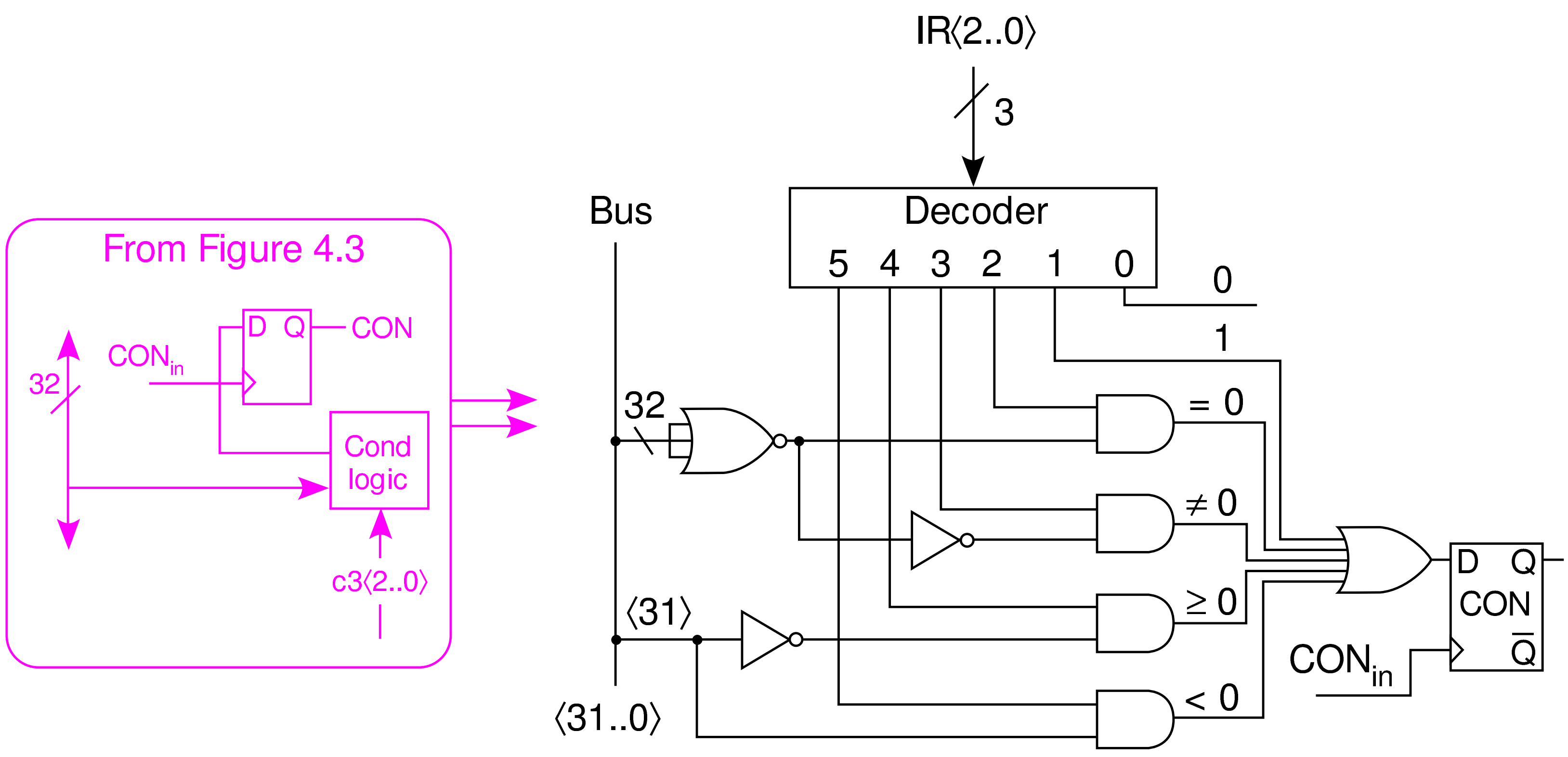
Branching

$$\begin{aligned}\text{cond} := (\ & c3\langle 2..0\rangle = 0 \rightarrow 0: \\ & c3\langle 2..0\rangle = 1 \rightarrow 1: \\ & c3\langle 2..0\rangle = 2 \rightarrow R[rc] = 0: \\ & c3\langle 2..0\rangle = 3 \rightarrow R[rc] \neq 0: \\ & c3\langle 2..0\rangle = 4 \rightarrow R[rc]\langle 31\rangle = 0: \\ & c3\langle 2..0\rangle = 5 \rightarrow R[rc]\langle 31\rangle = 1\ ):\end{aligned}$$

- **This is equivalent to the logic expression**

$$\begin{aligned}\text{cond} = {} & (c3\langle 2..0\rangle = 1) \vee (c3\langle 2..0\rangle = 2) \wedge (R[rc] = 0) \vee \\ & (c3\langle 2..0\rangle = 3) \wedge \neg(R[rc] = 0) \vee (c3\langle 2..0\rangle = 4) \wedge \neg R[rc]\langle 31\rangle \vee \\ & (c3\langle 2..0\rangle = 5) \wedge R[rc]\langle 31\rangle\end{aligned}$$

Fig 4.9 Computation of the Conditional Value CON



- NOR gate does = 0 test of R[rc] on bus

Tbl 4.11 Control Sequence for SRC Branch Instruction, br

$$\text{br } (:= \text{op} = 8) \rightarrow (\text{cond} \rightarrow \text{PC} \leftarrow \text{R[rb]}):$$

Step	Concrete RTN	Control Sequence
T0–T2	Instruction fetch	Instruction fetch
T3	CON ← cond(R[rc]);	Grc, R_{out}, CON_{in}
T4	CON → PC ← R[rb];	Grb, R_{out}, CON → PC_{in}, End

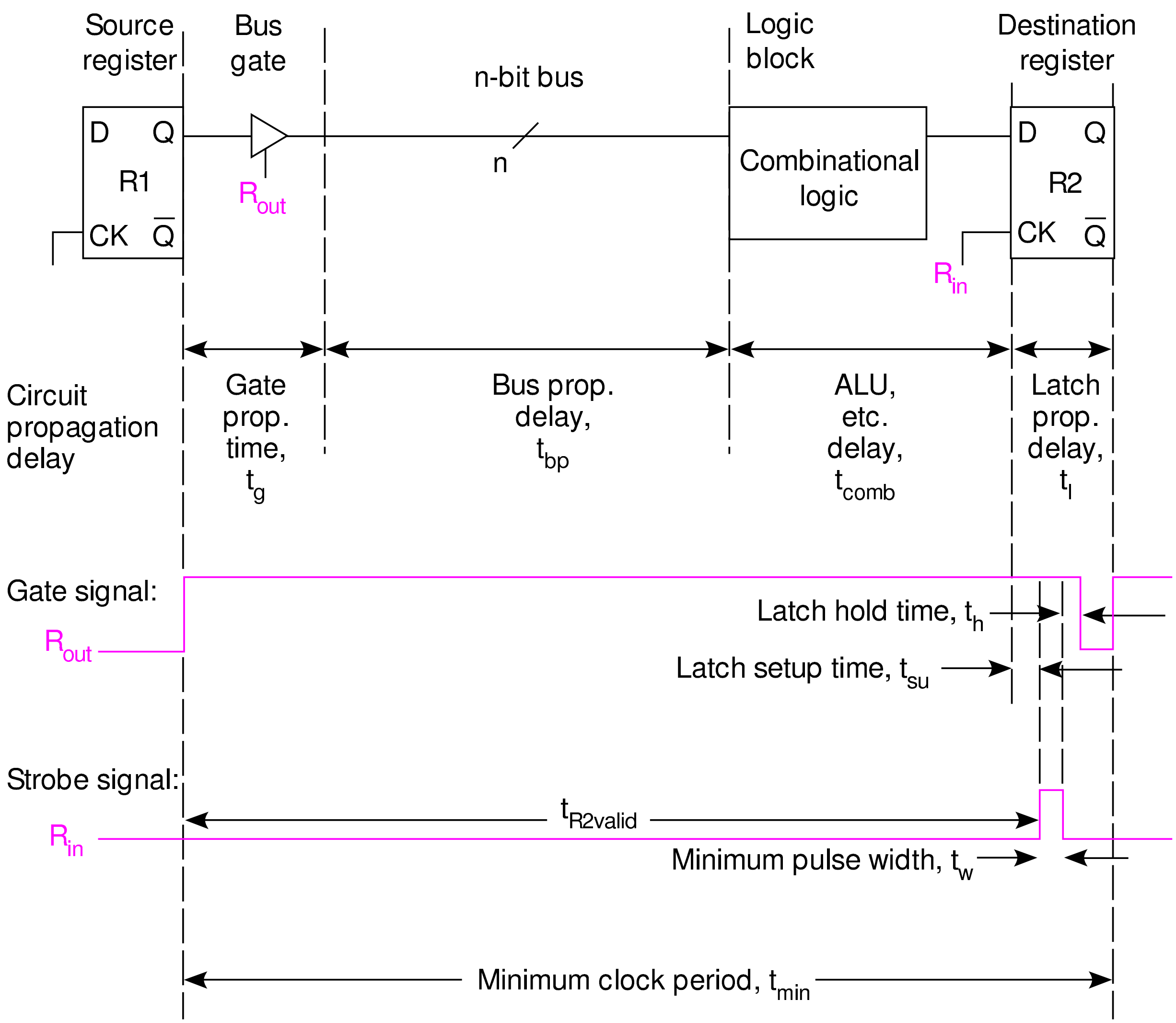
- Condition logic is always connected to CON, so R[rc] only needs to be put on bus in T3
- Only PC_{in} is conditional in T4 since gating R[rb] to bus makes no difference if it is not used

Summary of the Design Process

Informal description ⇒ formal RTN description ⇒ block diagram architecture ⇒ concrete RTN steps ⇒ hardware design of blocks ⇒ control sequences ⇒ control unit and timing

- At each level, more decisions must be made
  - These decisions refine the design
  - Also place requirements on hardware still to be designed
- The nice one-way process above has circularity
  - Decisions at later stages cause changes in earlier ones
  - Happens less in a text than in reality because
    - Can be fixed on re-reading
    - Confusing to first-time student

Fig 4.10 Clocking the Data Path: Register Transfer Timing



- $t_{R2valid}$ is the period from begin of gate signal till inputs to R2 are valid
- t_{comb} is delay through combinational logic, such as ALU or cond logic

Signal Timing on the Data Path

- Several delays occur in getting data from R1 to R2
- Gate delay through the 3-state bus driver—t_g
- Worst case propagation delay on bus—t_{bp}
- Delay through any logic, such as ALU—t_{comb}
- Set up time for data to affect state of R2—t_{su}
- Data can be strobed into R2 after this time

$$t_{R2valid} = t_g + t_{bp} + t_{comb} + t_{su}$$

- Diagram shows strobe signal in the form for a latch. It must be high for a minimum time—t_w
- There is a hold time, t_h, for data after strobe ends

Effect of Signal Timing on Minimum Clock Cycle

- A total latch propagation delay is the sum

$$T_l = t_{su} + t_w + t_h$$

  - All above times are specified for latch
  - t_h may be very small or zero
- The minimum clock period is determined by finding longest path from ff output to ff input
  - This is usually a path through the ALU
  - Conditional signals add a little gate delay
- Using this path, the minimum clock period is

$$t_{min} = t_g + t_{bp} + t_{comb} + t_l$$

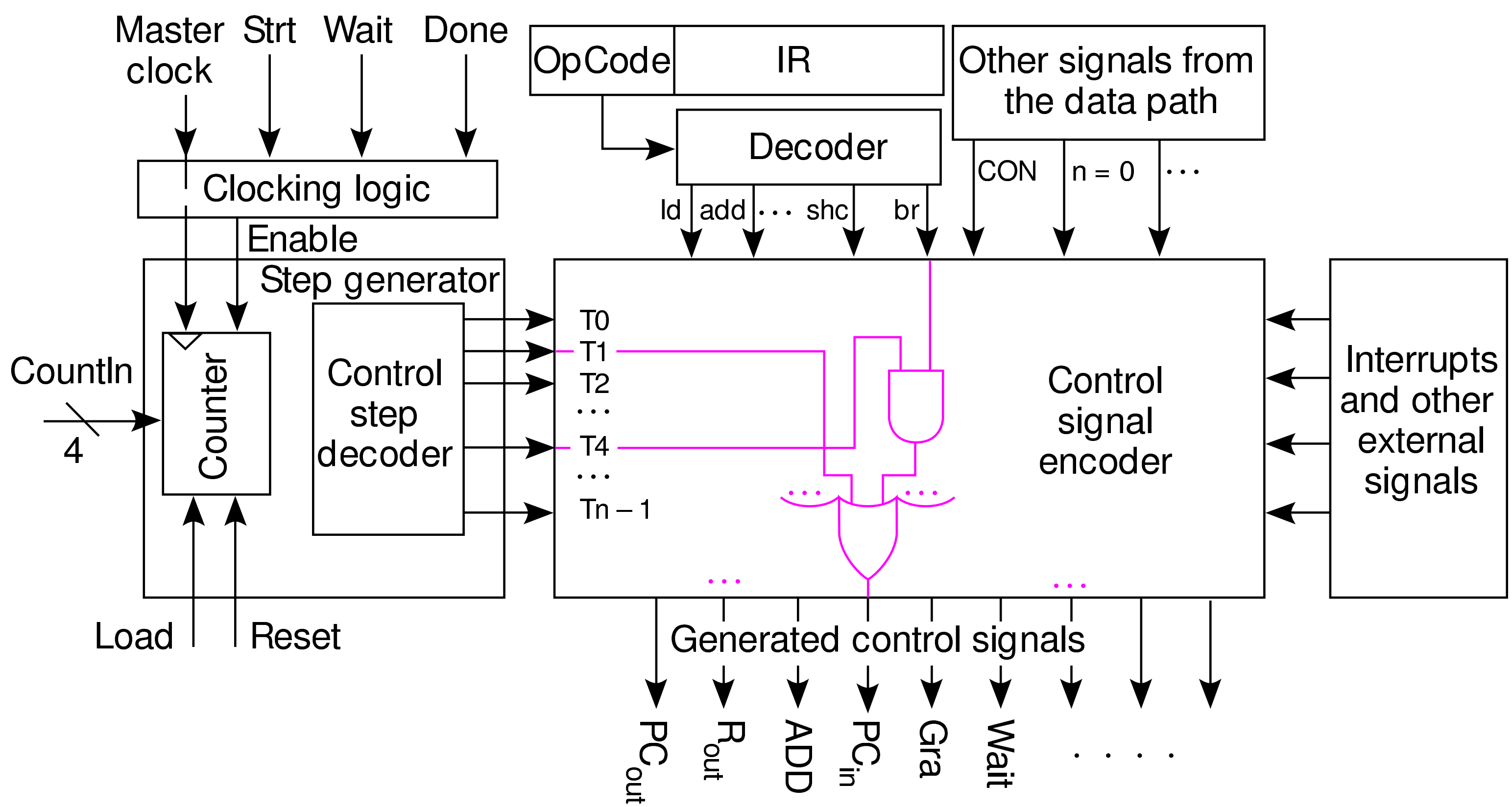
Latches Versus Edge-Triggered or Master-Slave Flip-Flops

- **During the high part of a strobe a latch changes its output**
- **If this output can affect its input, an error can occur**
- **This can influence even the kind of concrete RTs that can be written for a data path**
- **If the C register is implemented with latches, then**
  $C \leftarrow C + MD$; is not legal
- **If the C register is implemented with master-slave or edge-triggered flip-flops, it is OK**

The Control Unit

- The control unit's job is to generate the control signals in the proper sequence
- Things the control signals depend on
  - The time step Ti
  - The instruction opcode (for steps other than T0, T2, T2)
  - Some few data path signals like CON, n = 0, etc.
  - Some external signals: reset, interrupt, etc. (to be covered)
- The components of the control unit are: a time state generator, instruction decoder, and combinational logic to generate control signals

Fig 4.11 Control Unit Detail with Inputs and Outputs



Synthesizing Control Signal Encoder Logic

Step	Control Sequence
T0.	PC_{out}, MA_{in}, Inc4, C_{in}, Read
T1.	C_{out}, PC_{in}, Wait
T2.	MD_{out}, IR_{in}

add

Step	Control Sequence
T3.	Grb, R_{out}, A_{in}
T4.	Grc, R_{out}, ADD, C_{in}
T5.	C_{out}, **Gra,** R_{in}, End

addi

Step	Control Sequence
T3.	Grb, R_{out}, A_{in}
T4.	$c2_{out}$, ADD, C_{in}
T5.	C_{out}, **Gra,** R_{in}, End

st

Step	Control Sequence
T3.	Grb, BA_{out}, A_{in}
T4.	$c2_{out}$, ADD, C_{in}
T5.	C_{out}, MA_{in}
T6.	**Gra,** R_{out}, MD_{in}, Write
T7.	Wait, End

shr

Step	Control Sequence
T3.	$c1_{out}$, Ld
T4.	$n=0 \rightarrow$ (Grc, R_{out}, Ld)
T5.	Grb, R_{out}, C=B
T6.	$n \neq 0 \rightarrow$ (C_{out}, SHR, C_{in}, Decr, Goto7)
T7.	C_{out}, **Gra,** R_{in}, End

• • •

Design process:

- Comb through the entire set of control sequences.
- Find all occurrences of each control signal.
- Write an equation describing that signal.

Example: Gra = T5·(add + addi) + T6·st + T7·shr + ...

Use of Data Path Conditions in Control Signal Logic

Step	Control Sequence
T0.	PC_{out}, MA_{in}, Inc4, C_{in}, Read
T1.	C_{out}, PC_{in}, Wait
T2.	MD_{out}, IR_{in}

add

Step	Control Sequence
T3.	Grb, R_{out}, A_{in}
T4.	**Grc,** R_{out}, ADD, C_{in}
T5.	C_{out}, Gra, R_{in}, End

addi

Step	Control Sequence
T3.	Grb, R_{out}, A_{in}
T4.	$c2_{out}$, ADD, C_{in}
T5.	C_{out}, Gra, R_{in}, End

st

Step	Control Sequence
T3.	Grb, BA_{out}, A_{in}
T4.	$c2_{out}$, ADD, C_{in}
T5.	C_{out}, MA_{in}
T6.	Gra, R_{out}, MD_{in}, Write
T7.	Wait, End

shr

Step	Control Sequence
T3.	$c1_{out}$, Ld
T4.	**n=0 → (Grc,** R_{out}, Ld)
T5.	Grb, R_{out}, C=B
T6.	n≠0 → (C_{out}, SHR, C_{in}, Decr, Goto7)
T7.	C_{out}, Gra, R_{in}, End

● ● ●

Example: Grc = T4·add + T4·(n=0)·shr + ...

Fig 4.12 Generation of the logic for PC_{in} and G_{ra}

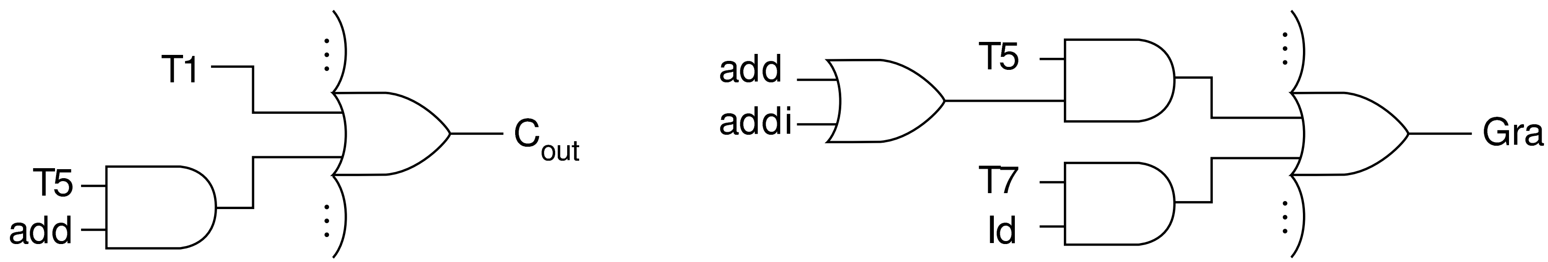
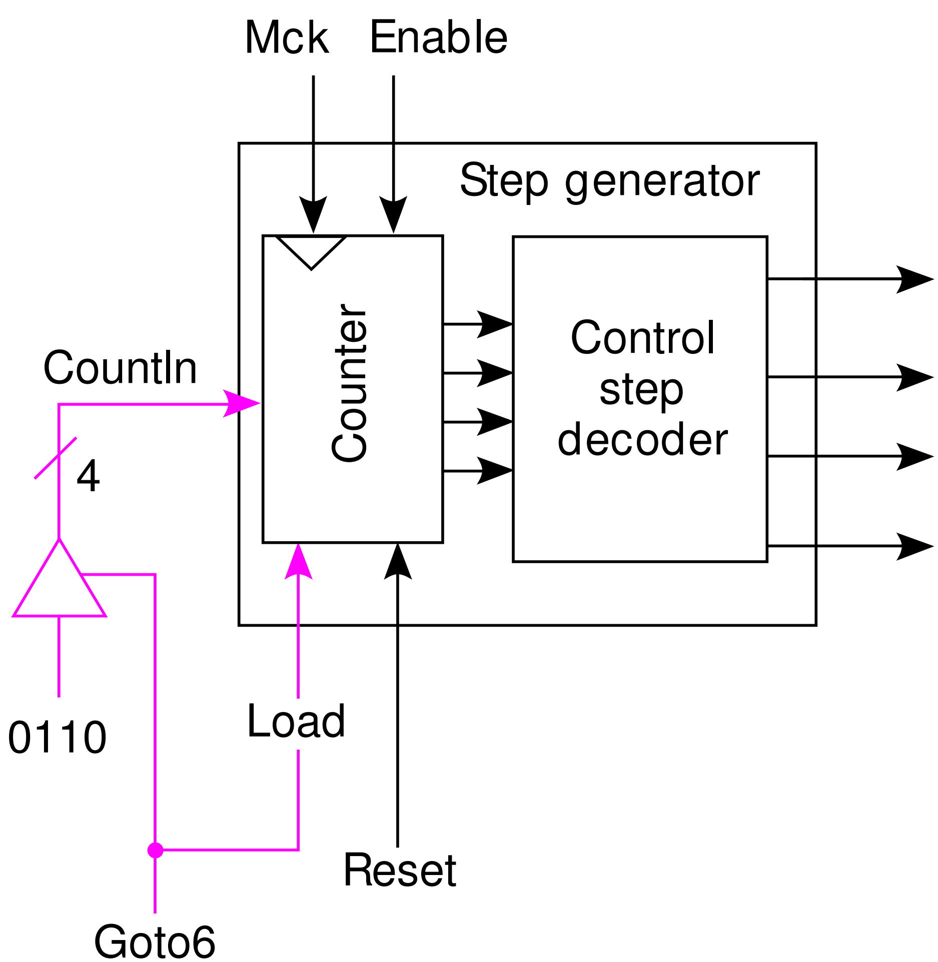
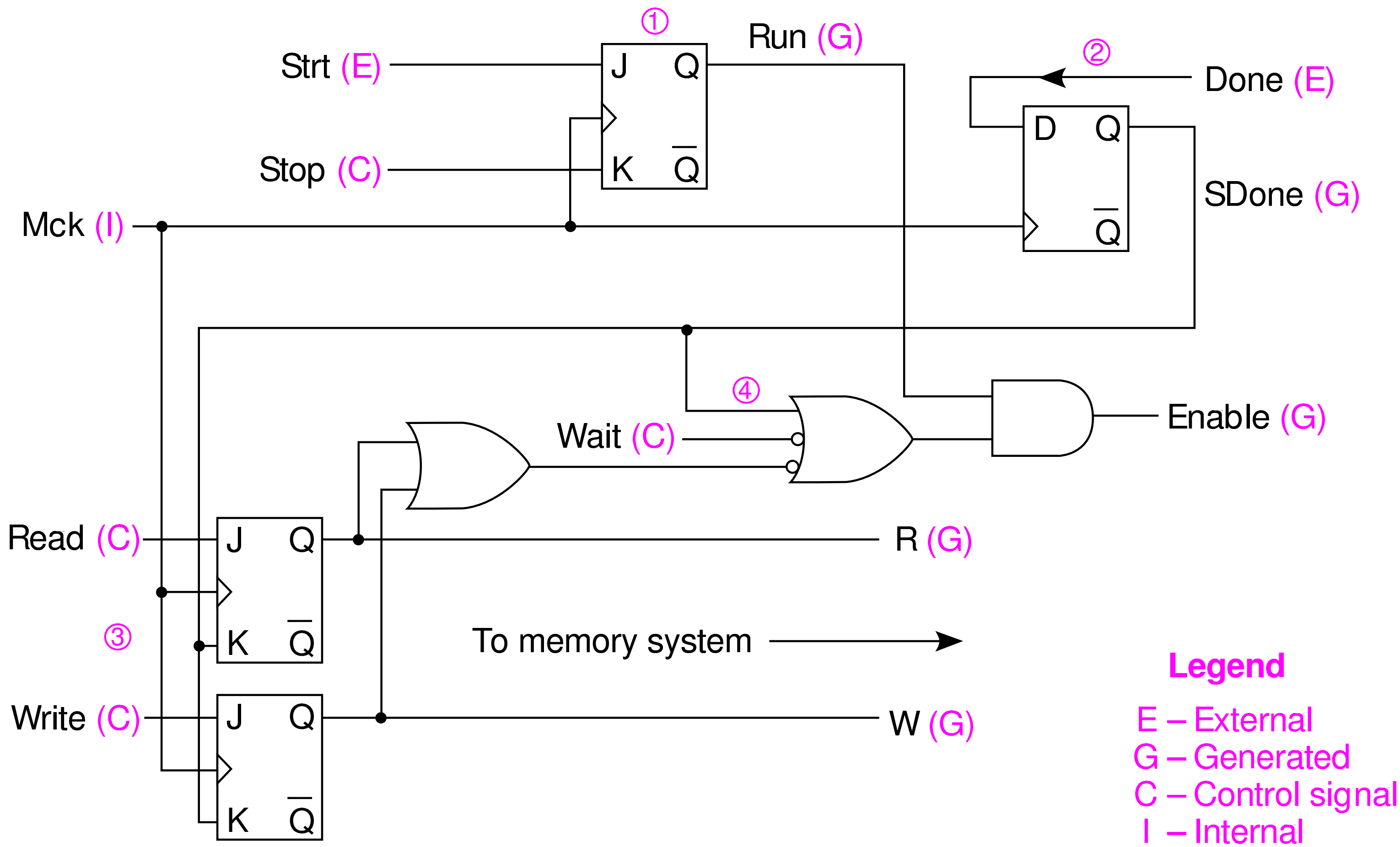


Fig 4.13 Branching in the Control Unit



- **3-state gates allow 6 to be applied to counter input**
- **Reset will synchronously reset counter to step T0**

Fig 4.14 The Clocking Logic: Start, Stop, and Memory Synchronization



- **Mck is master clock oscillator**

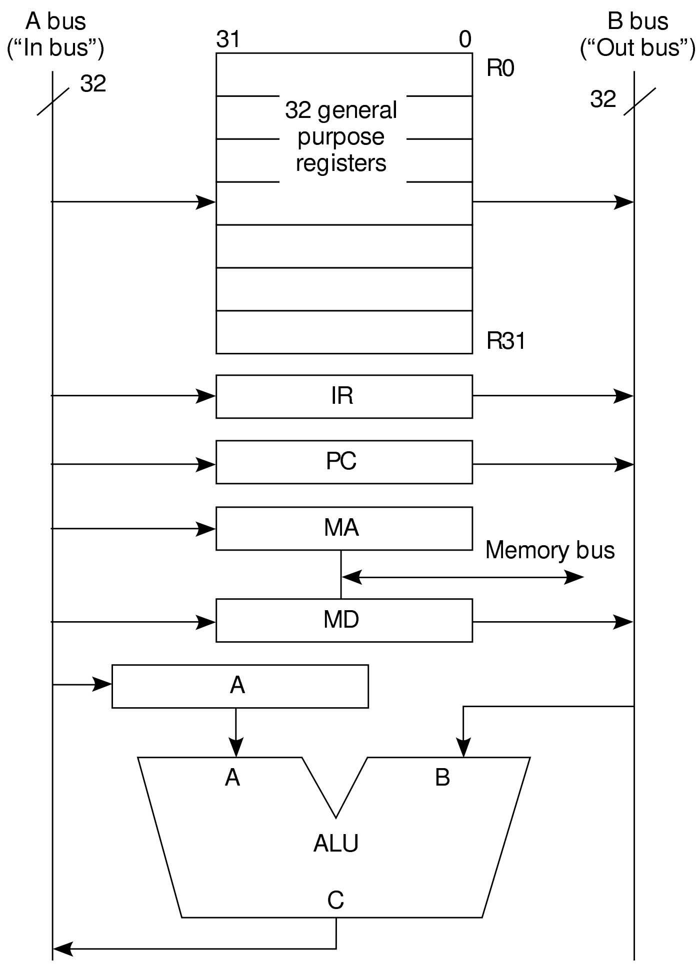
The Complete 1-Bus Design of SRC

- High-level architecture block diagram
- Concrete RTN steps
- Hardware design of registers and data path logic
- Revision of concrete RTN steps where needed
- Control sequences
- Register clocking decisions
- Logic equations for control signals
- Time step generator design
- Clock run, stop, and synchronization logic

Other Architectural Designs Will Require a Different RTN

- More data paths allow more things to be done in one step
- Consider a two bus design
- By separating input and output of ALU on different buses, the C register is eliminated
- Steps can be saved by strobing ALU results directly into their destinations

Fig 4.15 The 2-Bus SRC Microarchitecture



- **Bus A carries data going into registers**
- **Bus B carries data being gated out of registers**
- **ALU function C = B is used for all simple register transfers**

Tbl 4.13 The 2-Bus add Instruction

Step	Concrete RTN	Control Sequence
T0	MA ← PC;	PC_{out}, C = B, MA_{in}, Read
T1	PC ← PC + 4: MD ← M[MA];	PC_{out}, INC4, PC_{in}, Wait
T2	IR ← MD;	MD_{out}, C = B, IR_{in}
T3	A ← R[rb];	Grb, R_{out}, C = B, A_{in}
T4	R[ra] ← A + R[rc];	Grc, R_{out}, ADD, Sra, R_{in}, End

- Note the appearance of Grc to gate the output of the register rc onto the B bus and Sra to select ra to receive data strobed from the A bus
- Two register select decoders will be needed
- Transparent latches will be required at step T2

Performance and Design

$$\%\ Speedup = \frac{T_{1-bus} - T_{2-bus}}{T_{2-bus}} \times 100$$

Where

$$T = Execution\,Time = IC \times CPI \times \tau$$

Speedup By Going to 2 Buses

- **Assume for now that IC and τ don't change in going from 1 bus to 2 buses**
- **Naively assume that CPI goes from 8 to 7 clocks.**

$$\%Speedup = \frac{T_{1-bus} - T_{2-bus}}{T_{2-bus}} \times 100$$

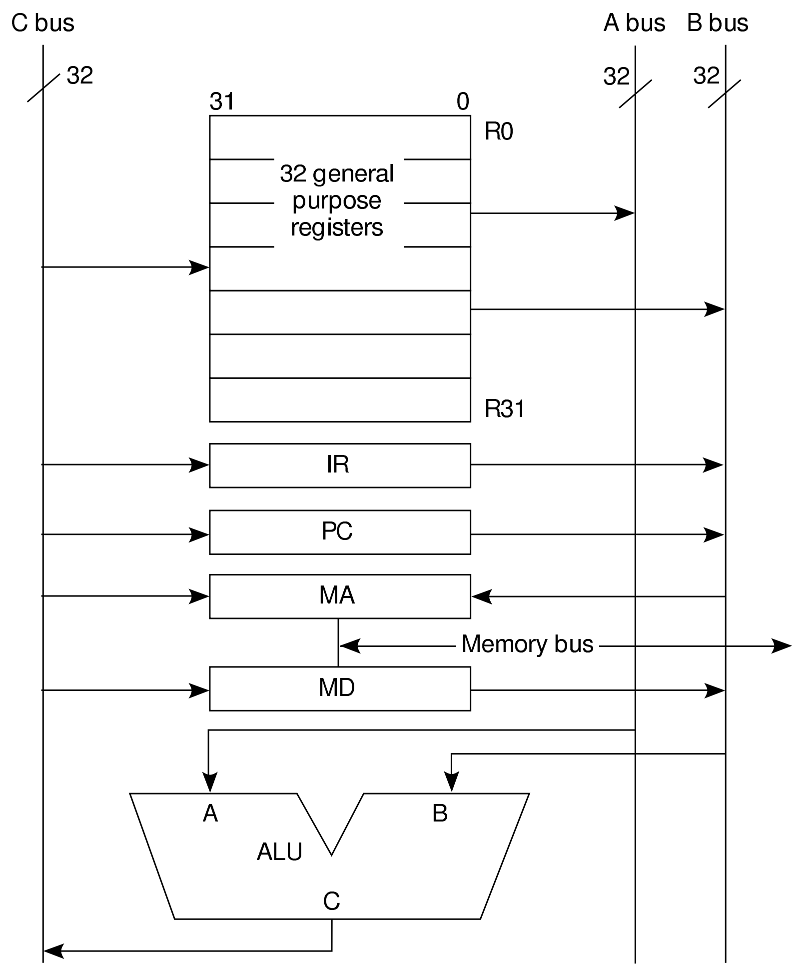
$$= \frac{IC \times 8 \times \tau - IC \times 7 \times \tau}{IC \times 7 \times \tau} \times 100 = \frac{8-7}{7} \times 100 = 14\%$$

Class Problem:
How will this speedup change if clock period of 2-bus machine is increased by 10%?

3-Bus Architecture Shortens Sequences Even More

- A 3-bus architecture allows both operand inputs and the output of the ALU to be connected to buses
- Both the C output register and the A input register are eliminated
- Careful connection of register inputs and outputs can allow multiple RTs in a step

Fig 4.16 The 3-Bus SRC Design



- **A-bus is ALU operand 1, B-bus is ALU operand 2, and C-bus is ALU output**
- **Note MA input connected to the B-bus**

Tbl 4.15 The 3-Bus add Instruction

Step	Concrete RTN	Control Sequence
T0	MA ← PC: MD ← M[MA]; PC ← PC + 4:	PC_{out}, MA_{in}, INC4, PC_{in}, Read, Wait
T1	IR ← MD;	MD_{out}, C = B, IR_{in}
T2	R[ra] ← R[rb] + R[rc];	GArc, RA_{out}, GBrb, RB_{out}, ADD, Sra, R_{in}, End

- Note the use of 3 register selection signals in step T2: GArc, GBrb, and Sra
- In step T0, PC moves to MA over bus B and goes through the ALU INC4 operation to reach PC again by way of bus C
  - PC must be edge-triggered or master-slave
- Once more MA must be a transparent latch

Performance and Design

- How does going to three buses affect performance?
- Assume average CPI goes from 8 to 4, while τ increases by 10%:

$$\%Speedup = \frac{IC \times 8 \times \tau - IC \times 4 \times 1.1\tau}{IC \times 4 \times 1.1\tau} \times 100 = \frac{8 - 4.4}{4.4} \times 100 = 82\%$$