CS221: Digital Design

http://jatinga.iitg.ernet.in/~asahu/cs221/

FF, Register and Memory

A. Sahu

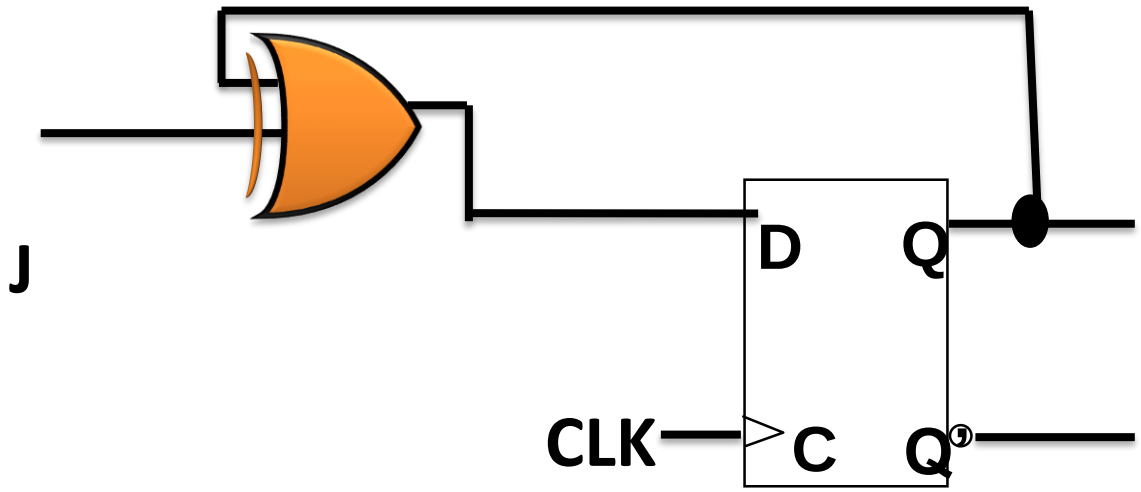
Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- FF: Characterization Table and Equation
  - RS, D, JK and T Flip flop
- Register
  - Parallel Load, Parallel out : (PIPO)
  - Serial Load, Wrap around load, Serial out (SISO)
  - PISO, SIPO Register
- Multifunction Register : How to design?

Given a D FF: Construct T FF



T	D	TQ'+T'Q
0	Q	0Q'+1Q
1	Q'	1Q'+0Q

D= TQ' + TQ

Characteristic Equations

- A descriptions of the next-state table of a flip-flop
- Constructing from the Karnaugh map for Q_{t+1} in terms of the present state and input

Characteristic tables

- The tables that we've made so far are called characteristic tables.
  - They show the next state Q(t+1) in terms of the current state Q(t) and the inputs.
  - For simplicity, the control input C is not usually listed.
  - Again, these tables don't indicate the positive edge-triggered behavior of the flip-flops that we'll be using.

J	K	Q+
0	**0**	**Qt**
0	1	0
1	0	1
1	**1**	**Qt'**

D	Q+
0	0
1	1

T	Q+
0	Qt
1	Qt'

Characteristic equations

- We can also write characteristic equations, where the next state Q(t+1) is defined in terms of the current state Q(t) and inputs.

J	K	Q+
0	**0**	**Qt**
0	1	0
1	0	1
1	**1**	**Qt'**

$$Q+= J'K'Q+JKQt'+JK'$$
$$= J'K'Q+JKQ'+JK'Q'+JK'Q$$
$$= J'K'Q+JK'Q + JKQ'+JK'Q'$$
$$= (J+J')K'Q+J(K+K')Q'$$

$$\mathbf{Q^+= K'Q + JQ'}$$

$$Q(t+1)= K'Q(t) + JQ'(t)$$

Characteristic equations

- We can also write characteristic equations, where the next state Q(t+1) is defined in terms of the current state Q(t) and inputs.

D	Q+
0	0
1	1

$Q^{+}=D$

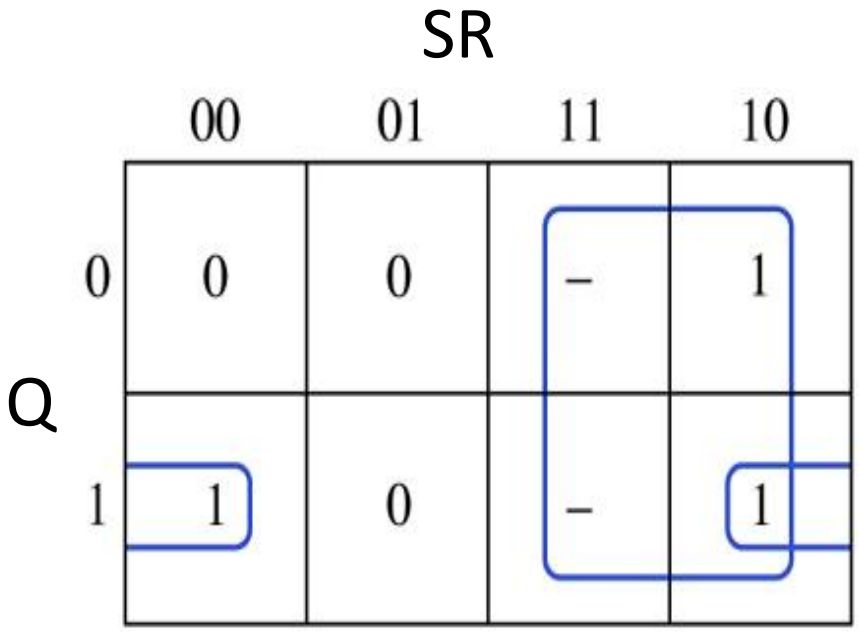
$Q(t+1) = D$

T	Q+
0	Qt
1	Qt’

$Q+= T'Q+TQ' = T \oplus Q$

$Q(t+1) = T'Q(t) + TQ'(t) = T \oplus Q(t)$

Characteristic equations



$Q^+ = S + R'Q \quad (SR=0)$

Flip Flop Type	Characteristic Equation
SR	$Q^+ = S + R'Q \quad (SR=0)$
JK	$Q^+ = JQ' + K'Q$
D	$Q^+ = D$
T	$Q^+ = TQ' + T'Q = T \oplus Q$

FF with Asynchronous Inputs

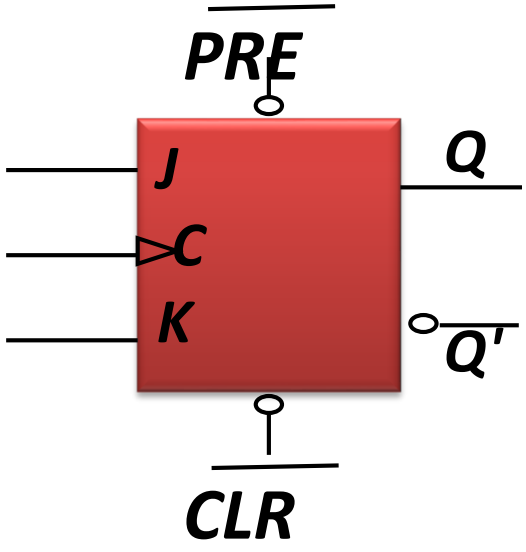
- S-R, D and J-K inputs are synchronous inputs
  - As data on these inputs are transferred to the flip-flop's output
  - Only on the triggered edge of the clock pulse.
- Asynchronous inputs affect the state of the flip-flop independent of the clock
- Example:
  - *Preset* (*PRE*) and *clear* (*CLR*)
  - or *direct set* (*SD*) and *direct reset* (*RD*)

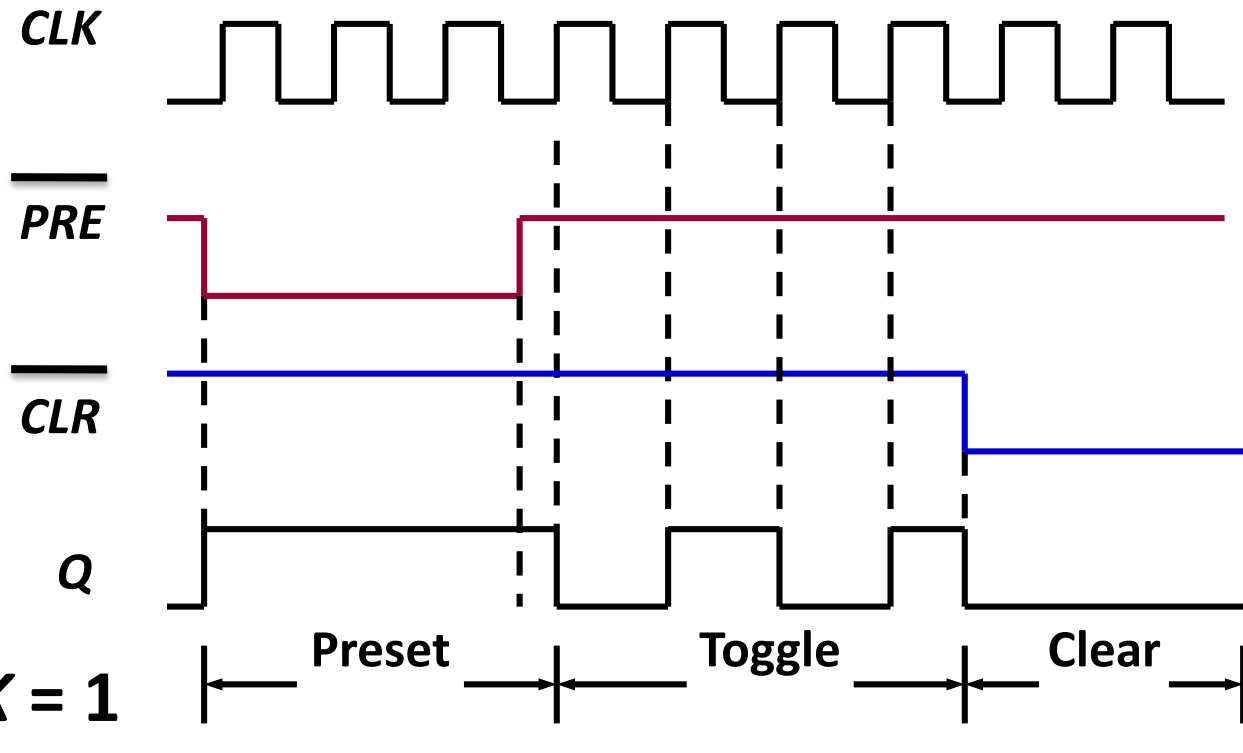
FF with Asynchronous Inputs

- When *PRE*=HIGH, *Q* is immediately set to HIGH.
- When *CLR*=HIGH, *Q* is immediately cleared to LOW.
- Flip-flop in normal operation mode when both *PRE* and *CLR* are LOW.

Asynchronous Inputs

- A J-K flip-flop with active-LOW preset and clear inputs.

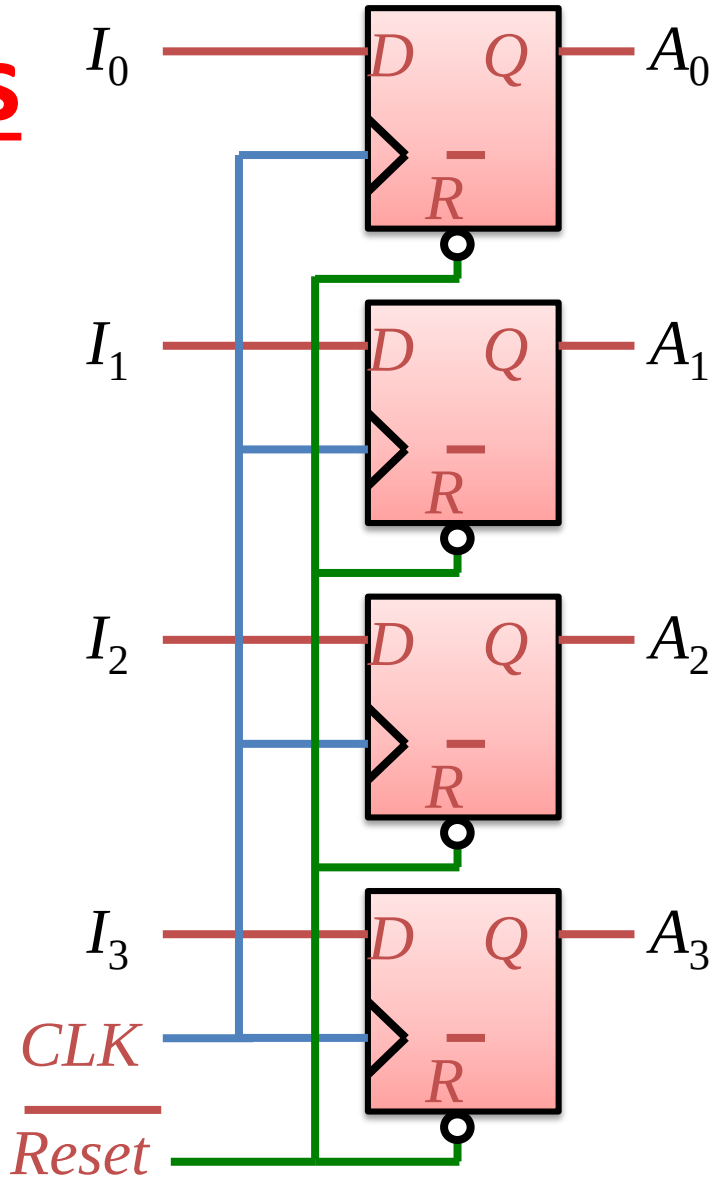




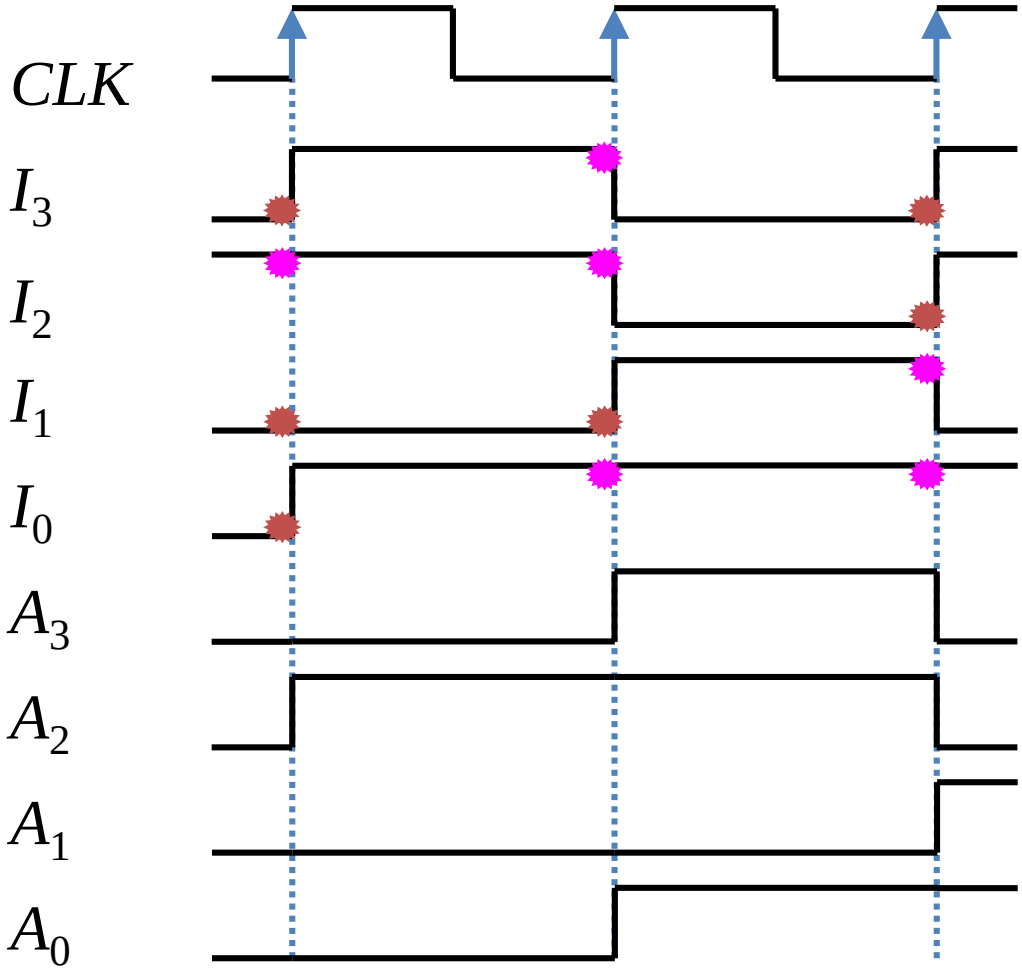
$J = K = 1$

Registers

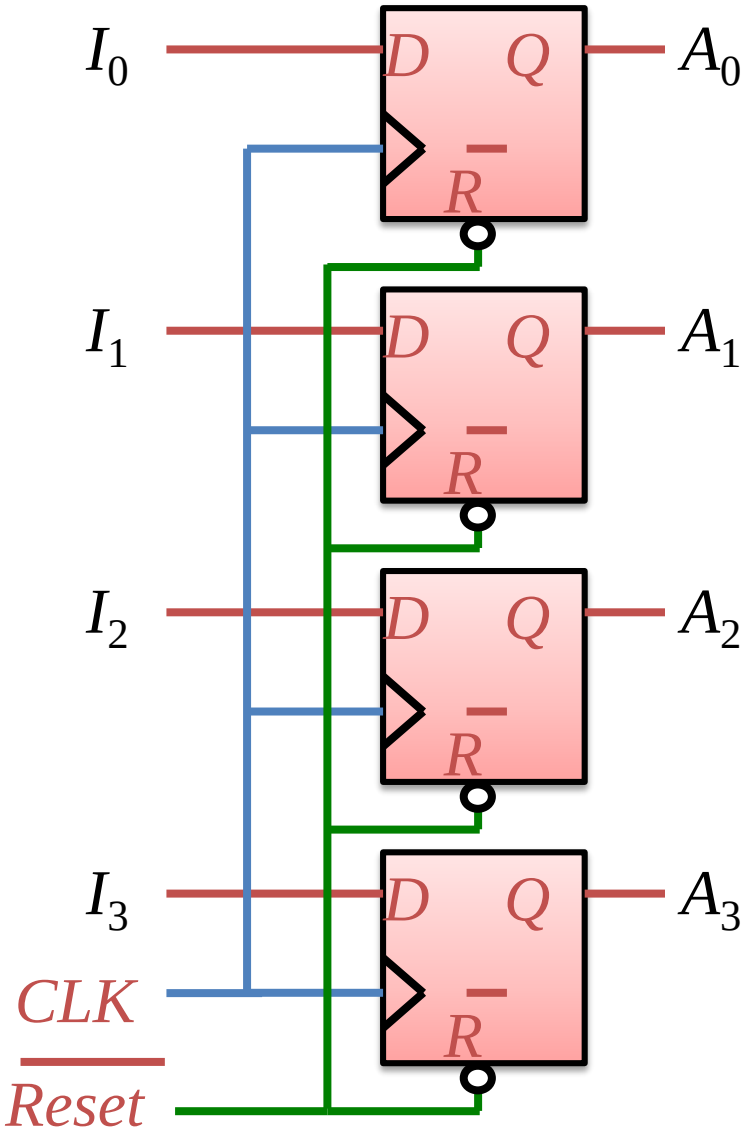
- Group of *D* Flip-Flops
- Synchronized (Single Clock)
- Store Data



Registers



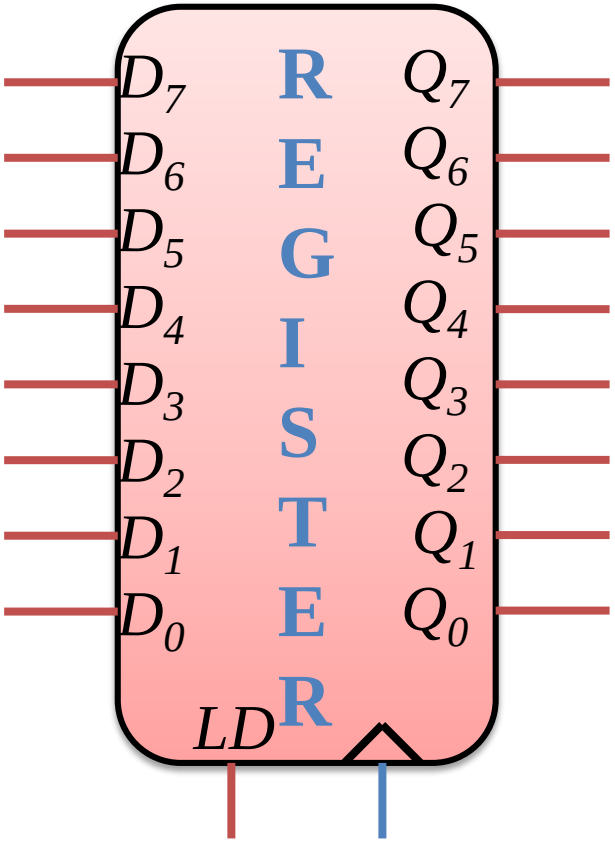
Note: New data has to go in with every clock



See carefully: Input at the dotted will be reflected to output : Just before rising edges

Registers with Parallel Load

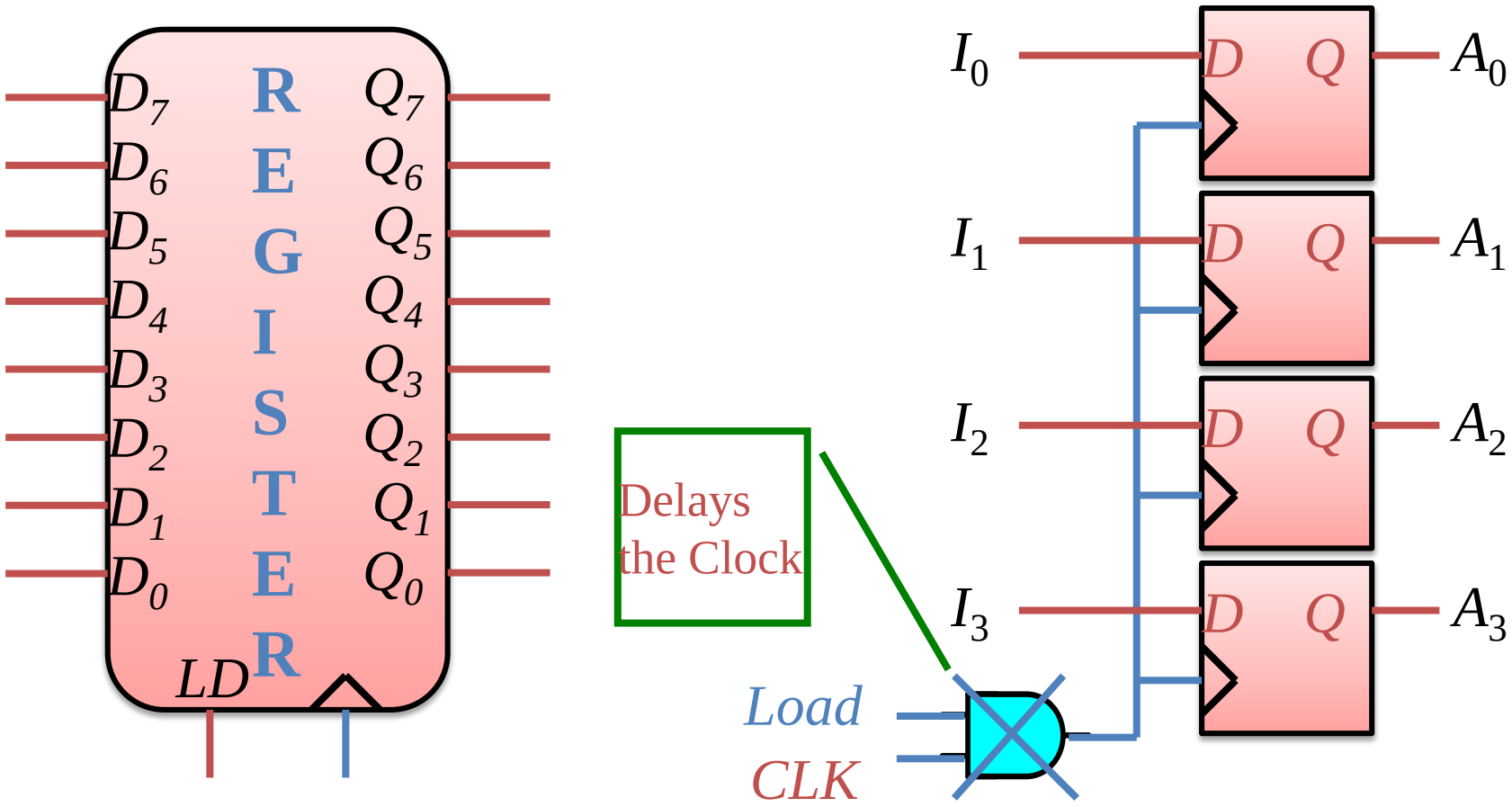
- Control ***Loading*** the Register with New Data



LD	Q(t+1)
0	Q(t)
1	D

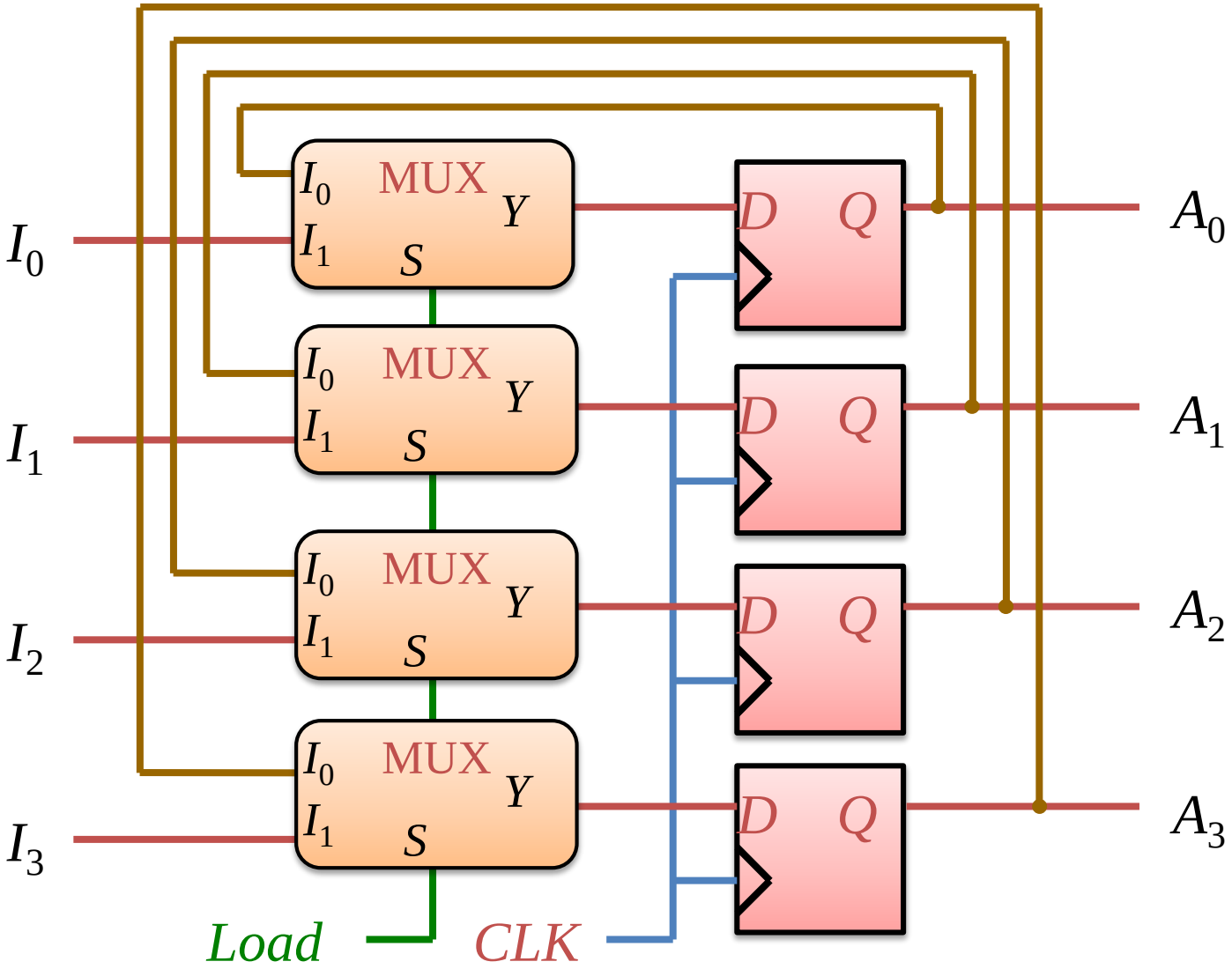
Registers with Parallel Load

- Should we block the “Clock” to keep the “Data”?



Registers with Parallel Load

- Circulate the “old data”



Shift Registers

- Register (Set of FFs)
- 4-Bit Shift Register (Example)
- Serial in Serial Out (SISO)

Shift Register

- Right Shift Example (Left shift is similar)
  - Move each bit one position right
  - Shift in 0 to leftmost bit

Q: Do four right shifts on 1001, showing value after each shift

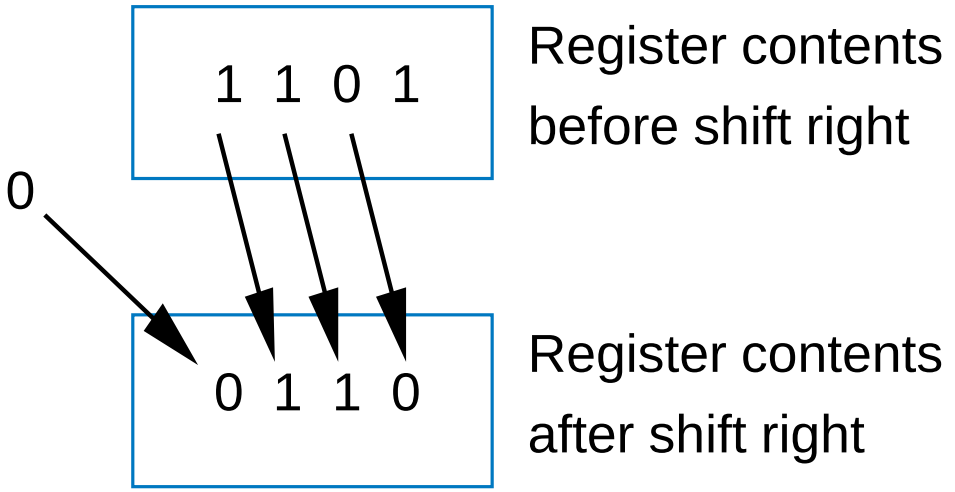
A: 1001 (original)

0100

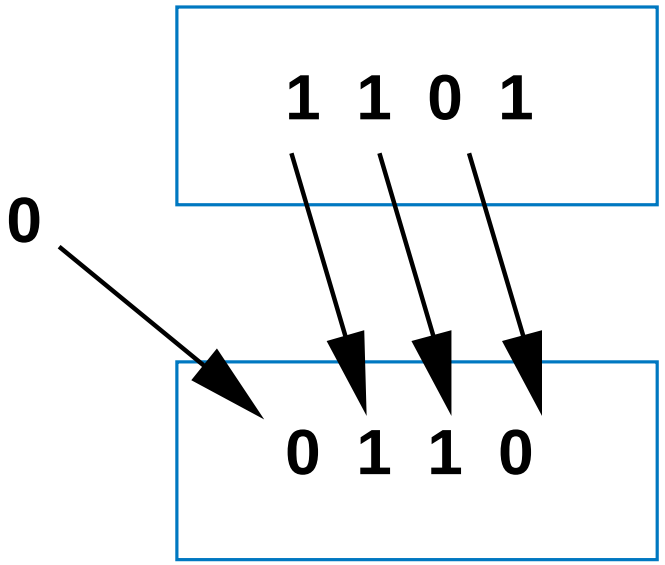
0010

0001

0000



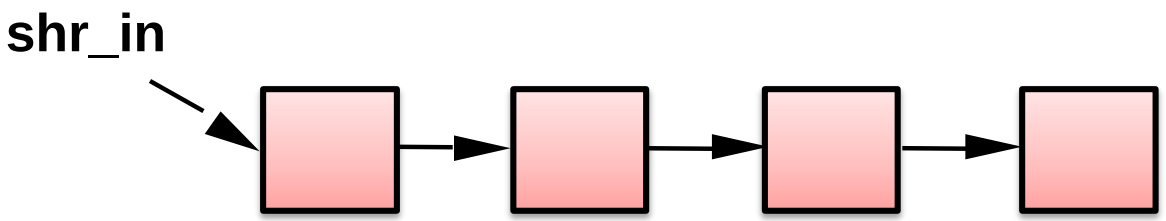
Shift Register



Register contents before shift right

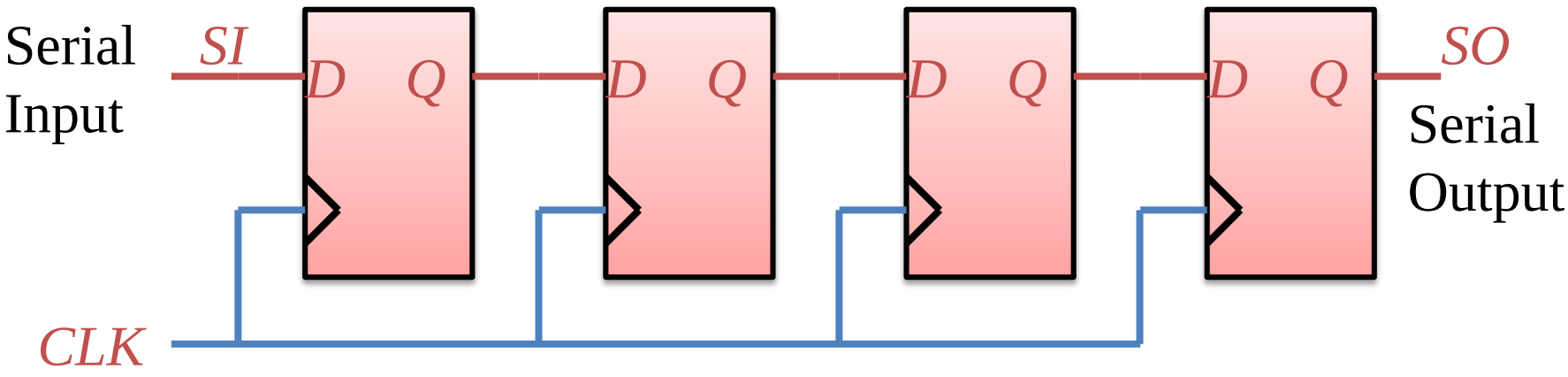
Register contents after shift right

- **Implementation:** Connect flip-flop output to next flip-flop's input

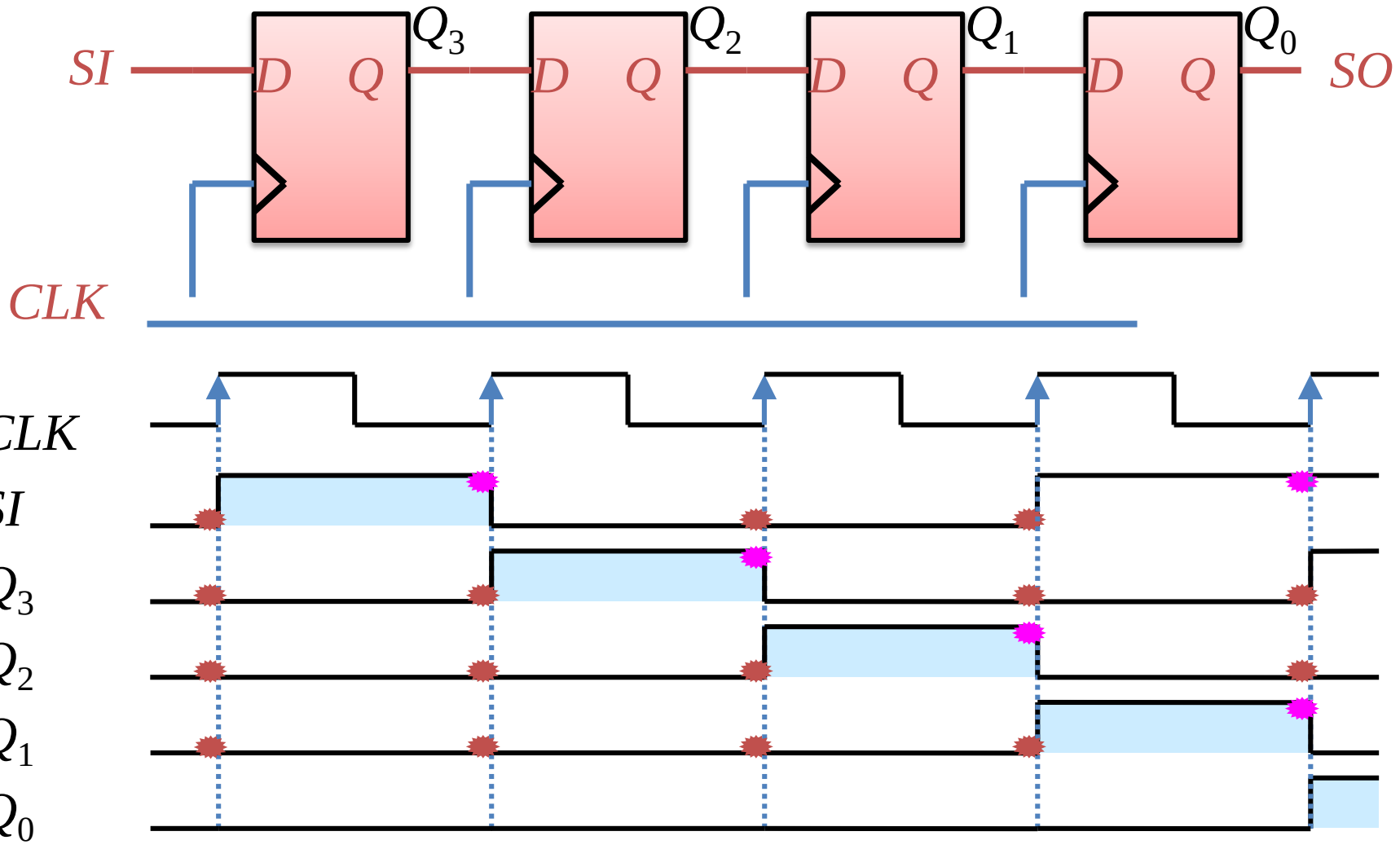


Shift Registers

- 4-Bit Shift Register
- Serial in Serial Out (SISO)

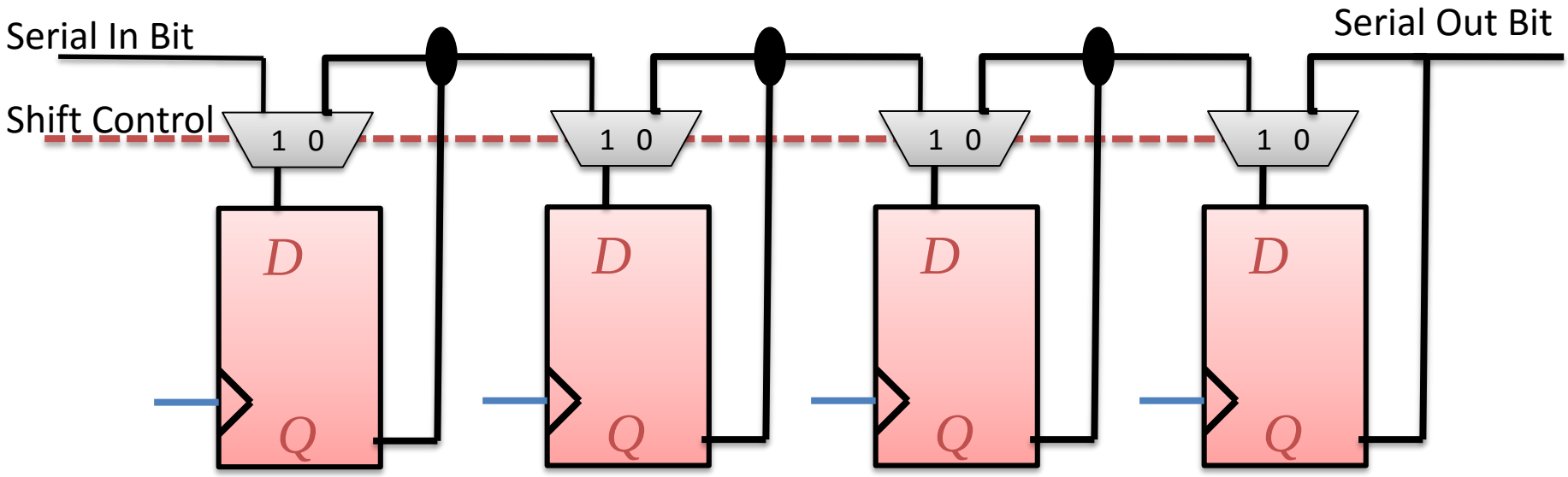


Shift Registers (SISO)



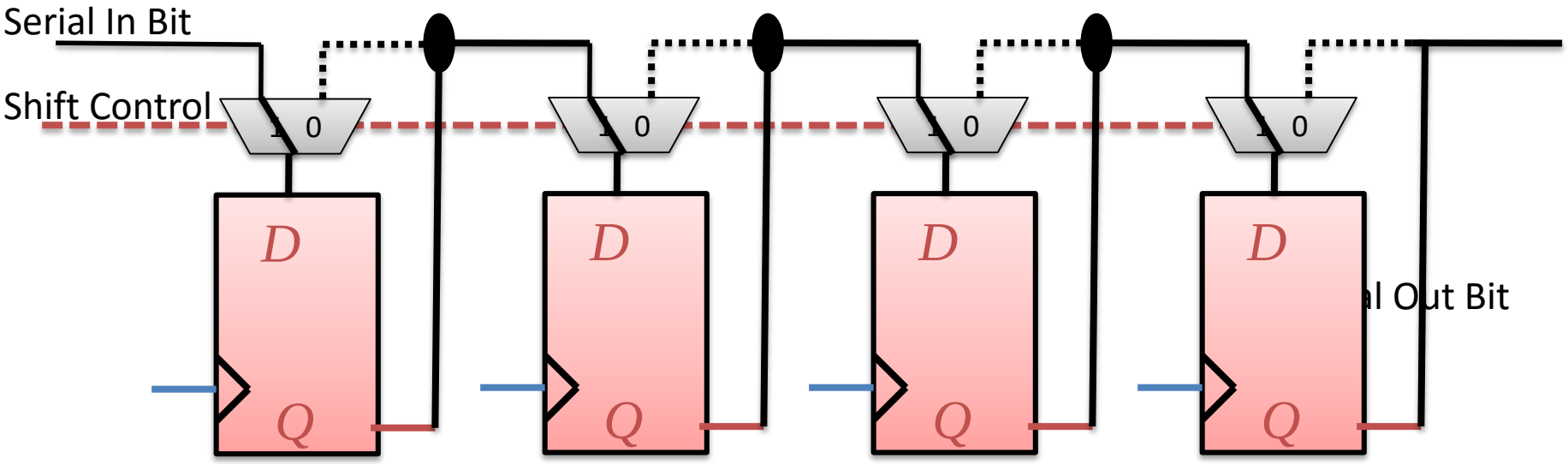
Shift Register with Control

- To allow register to either shift or retain, use 2x1 muxes
  - **Shift Control: shr: 0 means retain, 1 shift**
  - shr_in: value to shift in : May be 0/1, or right most bit



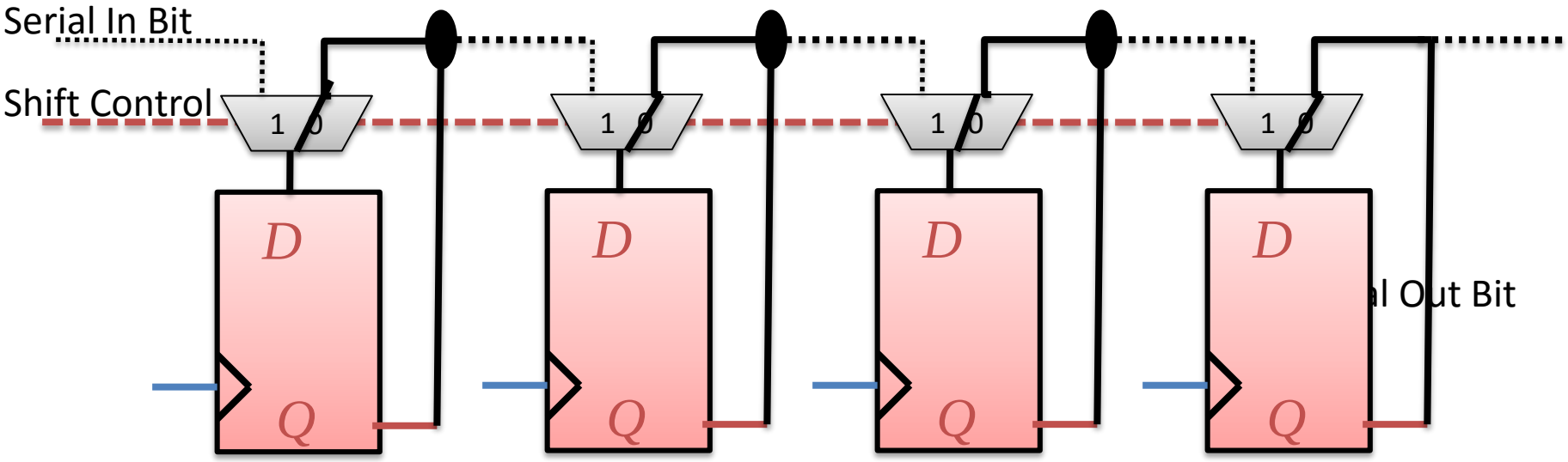
Shift Register with Control

Shift Control=1, Do one right shift per cycle

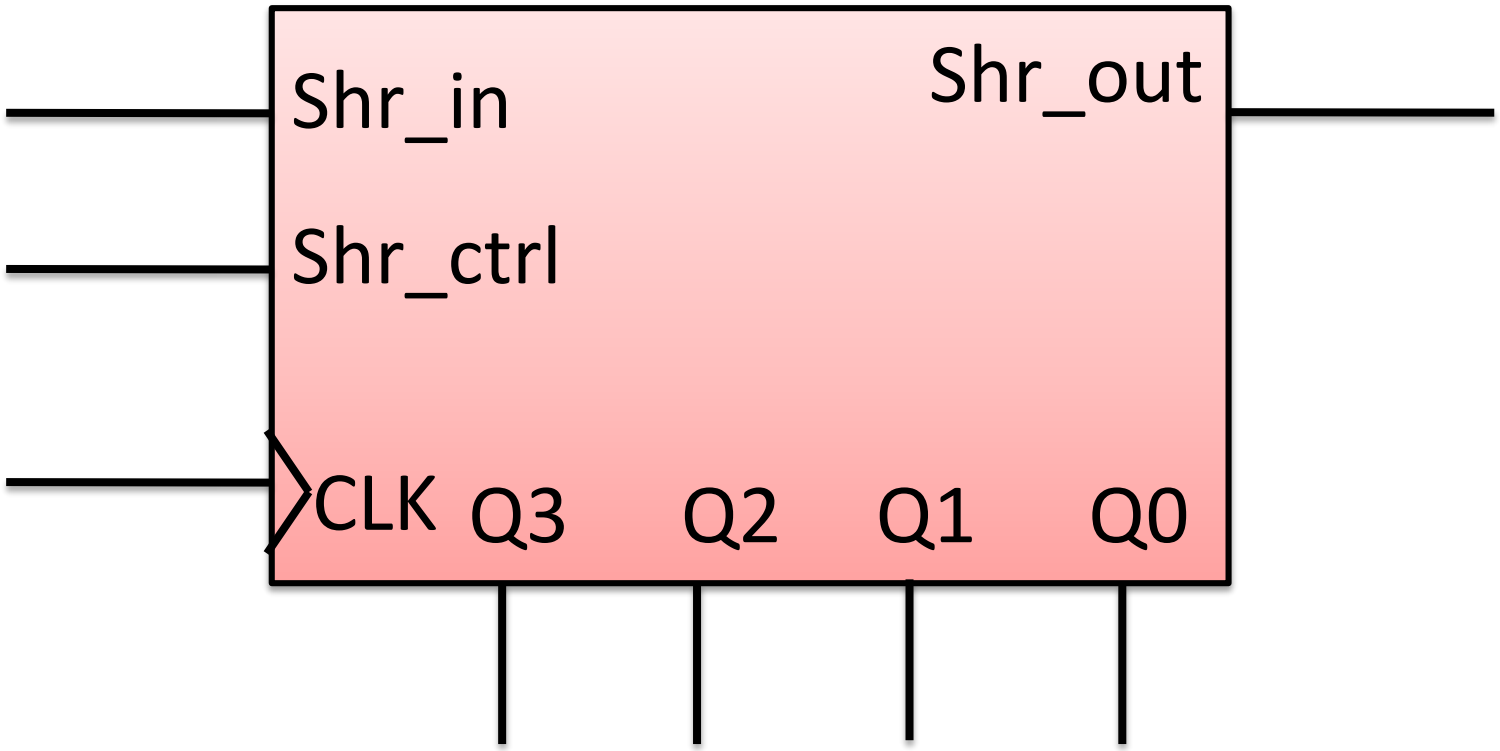


Shift Register with Control

Shift Control=0, No change to Data

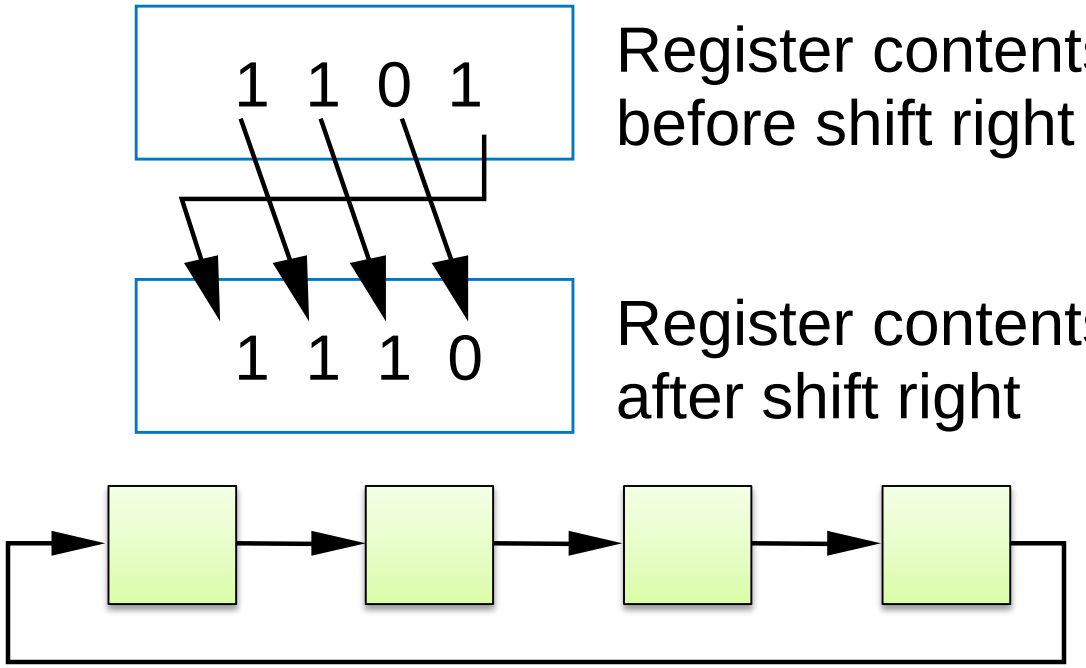


Shift Register with Control

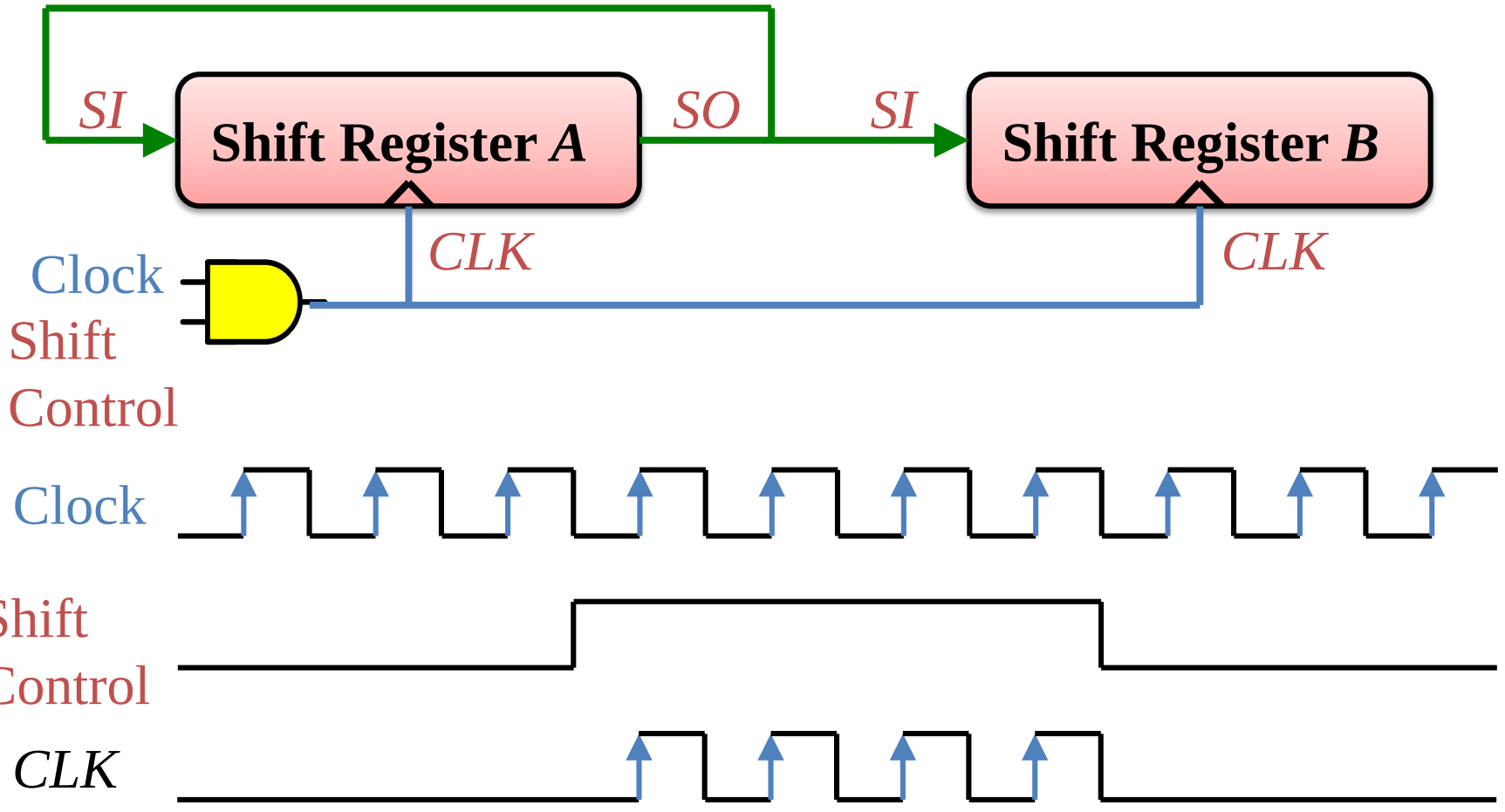


Rotate Register

- Rotate right: Like shift right, but leftmost bit comes from rightmost bit

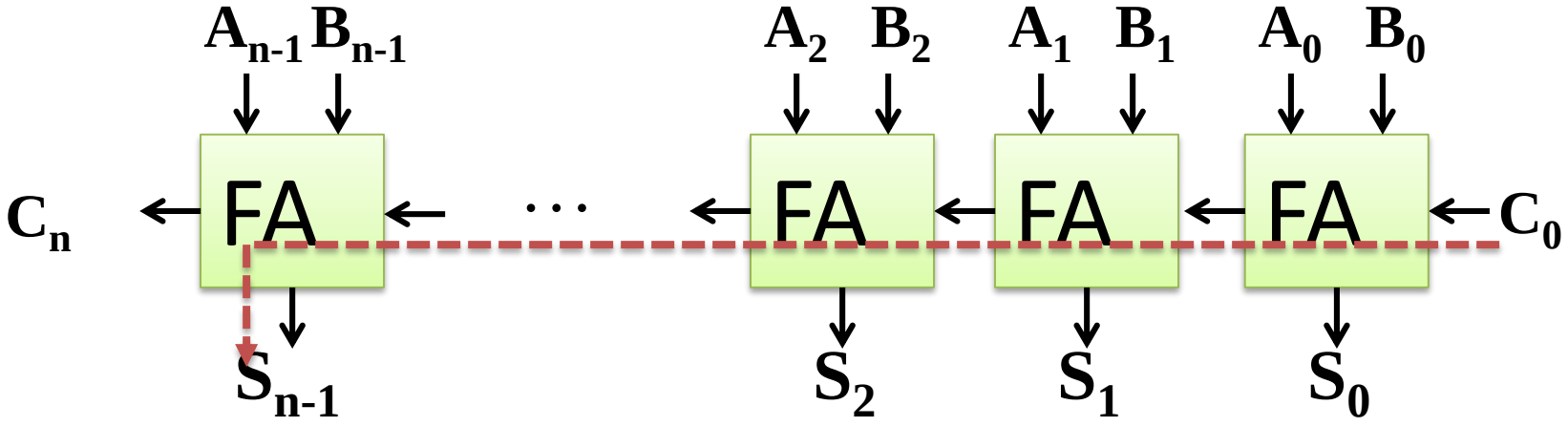


Serial Transfer

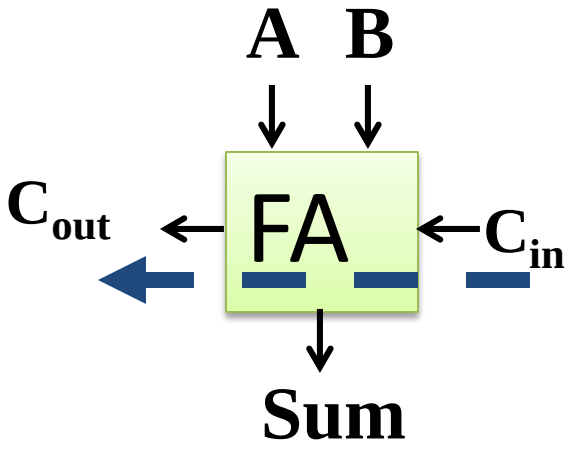


N-Bit RCA: Series of FA Cells

- To add two n-bit numbers

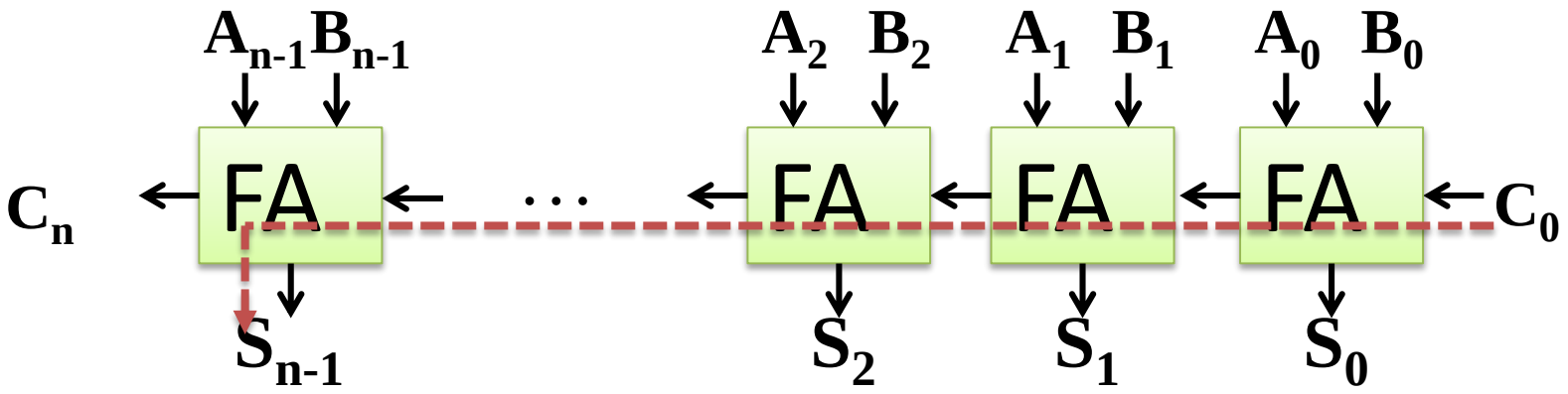


- Adder delay = Tc * n
- Tc = (C_{in} to C_{out} delay) of a FA



N-Bit Ripple-Carry Adder

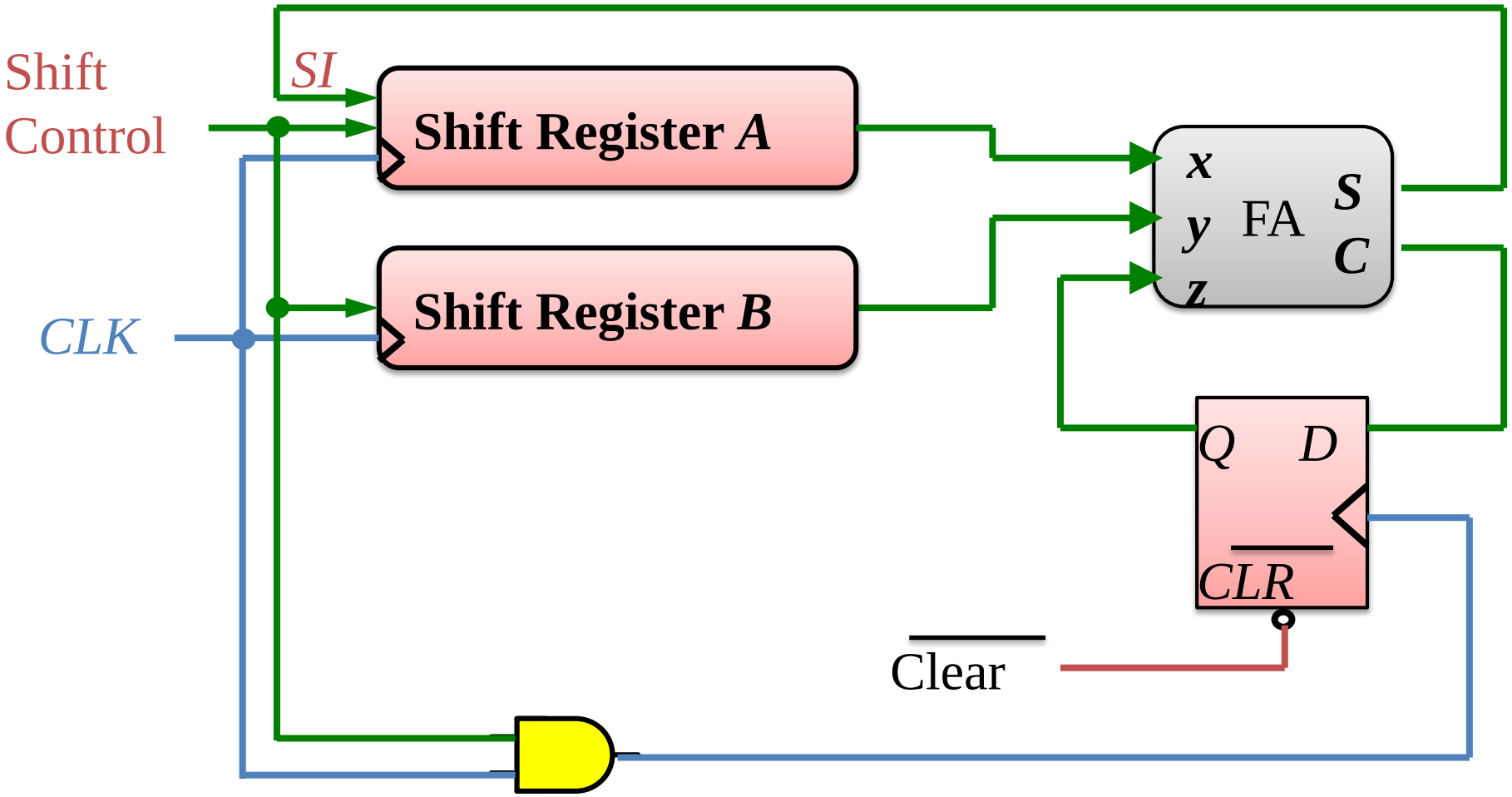
- To add two n-bit numbers



- Any point of time: only one FA is active

Do we require N Full Adder?

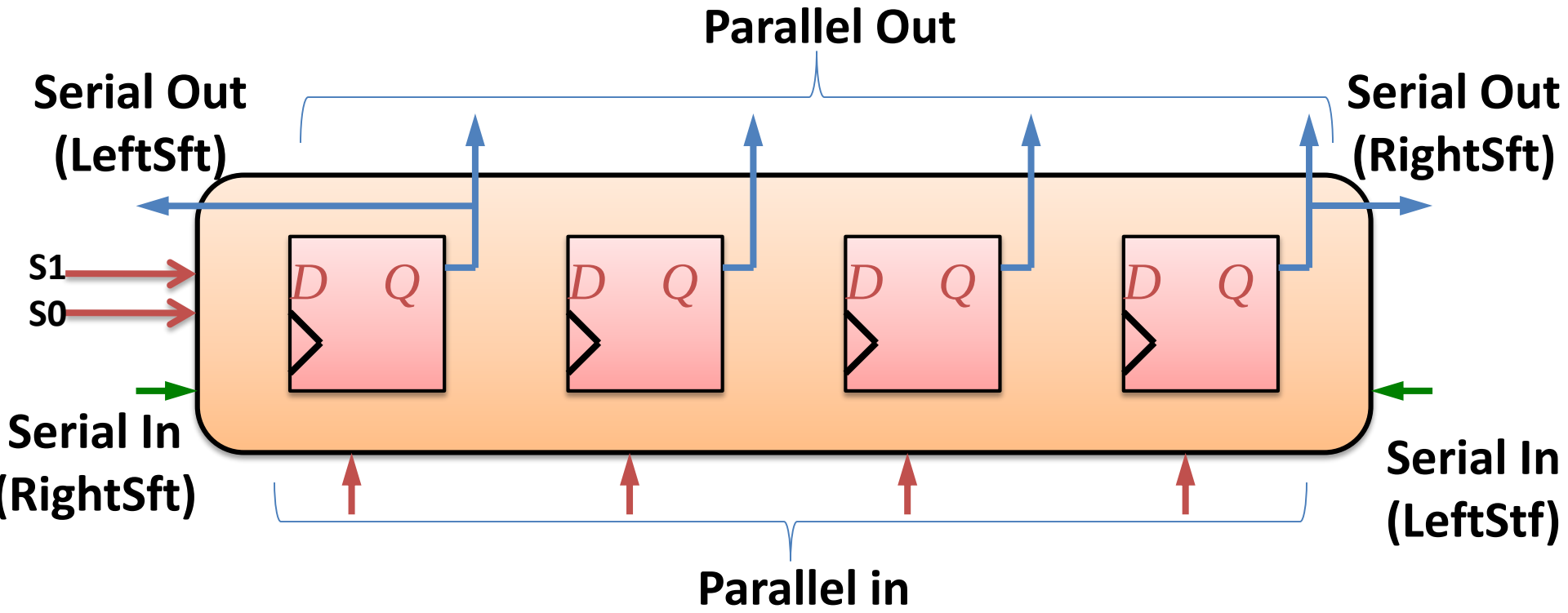
Serial Addition



Universal Shift Register

- Parallel-in Parallel-out (PIPO)
- Serial-in Serial-out (SISO)
- Serial-in Parallel-out (SIPO)
- Parallel-in Serial-out (PISO)

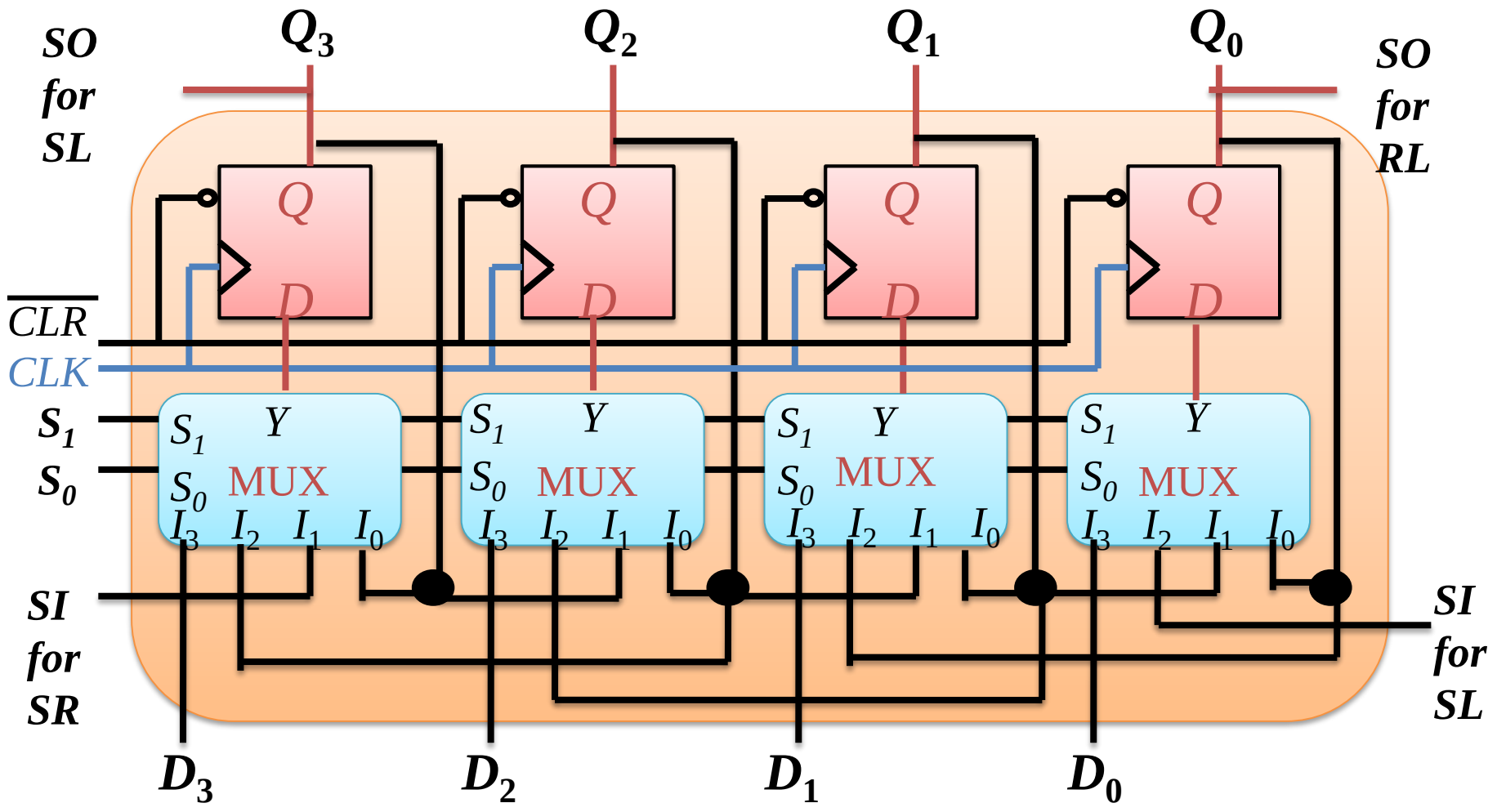
Universal Shift Register



Universal Shift Register

How to design such an USR?

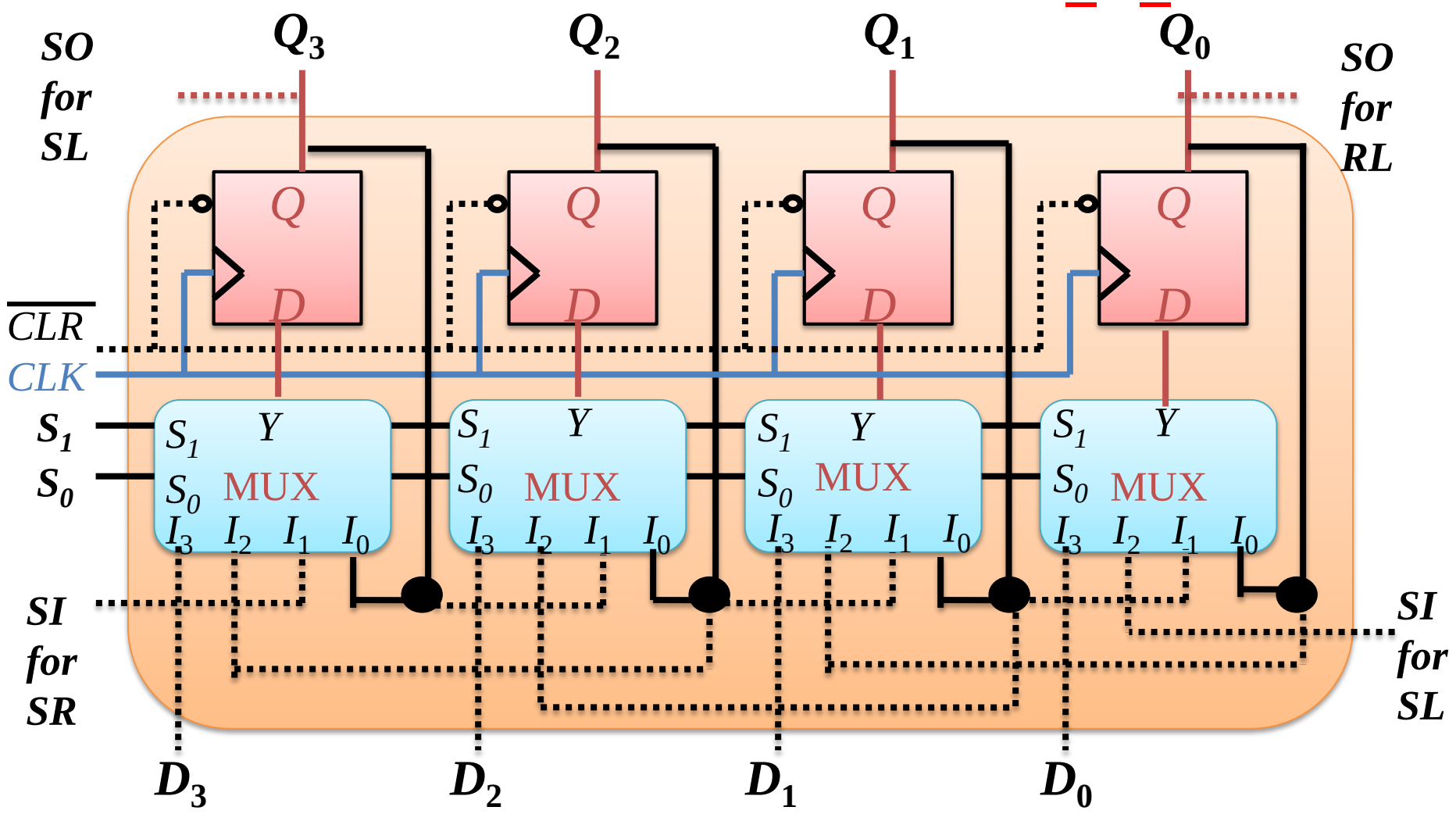
Universal Shift Register



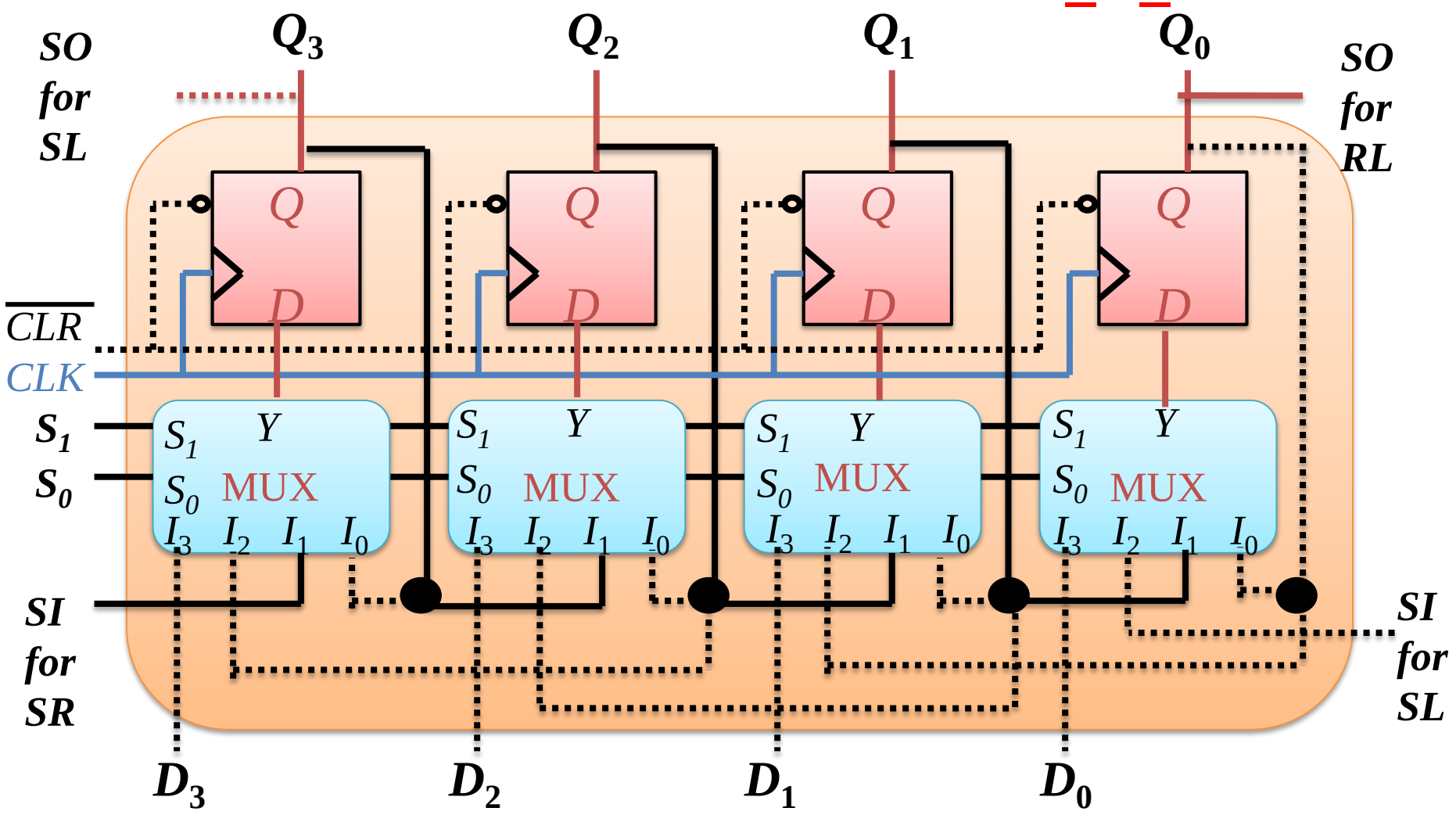
Operation Table for USR

S1	S0	Register Operation
0	0	No Change
0	1	Shift Right
1	0	Shift Left
1	1	Parallel Load

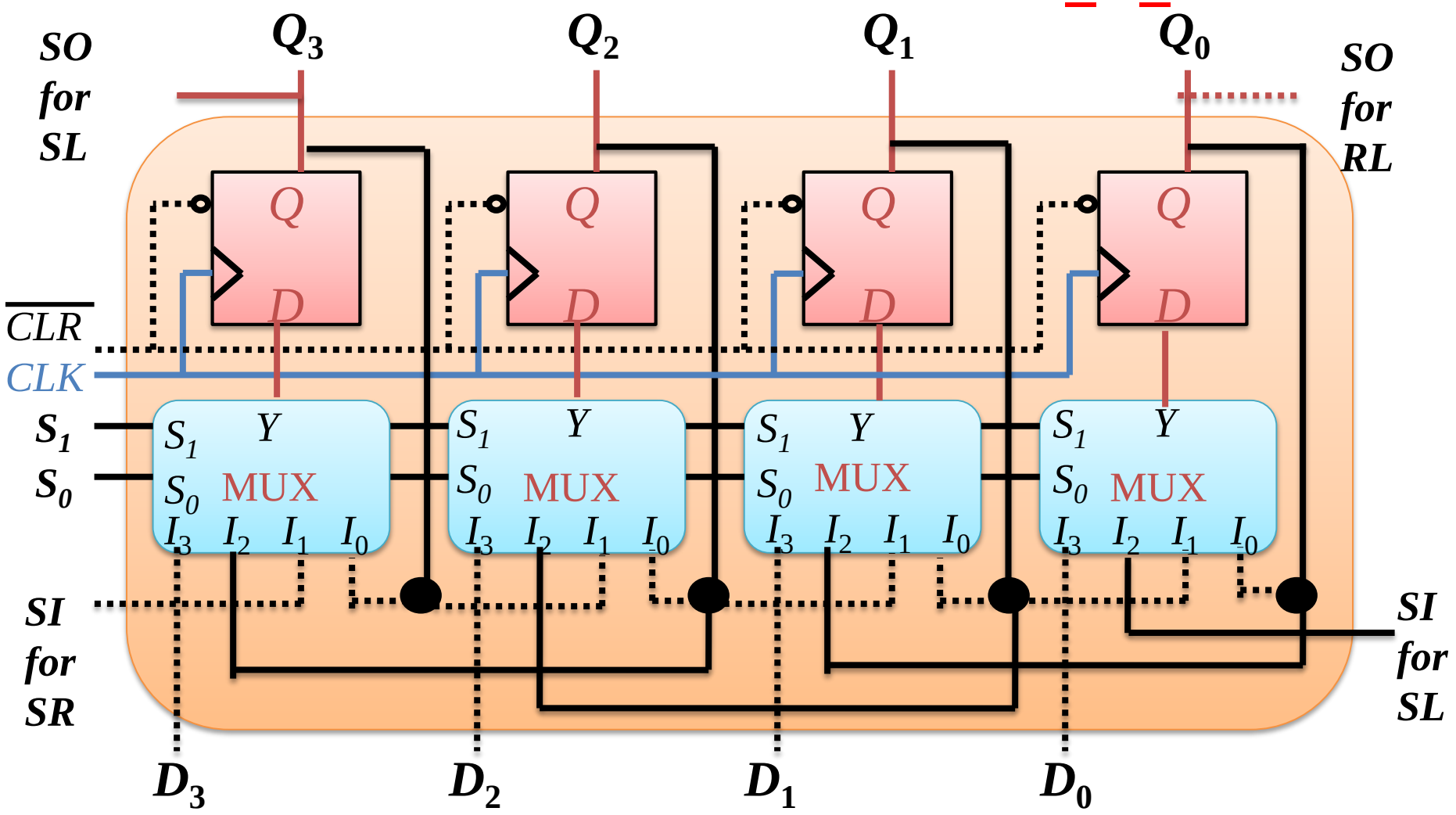
Universal Shift Register: S_1S_0=00



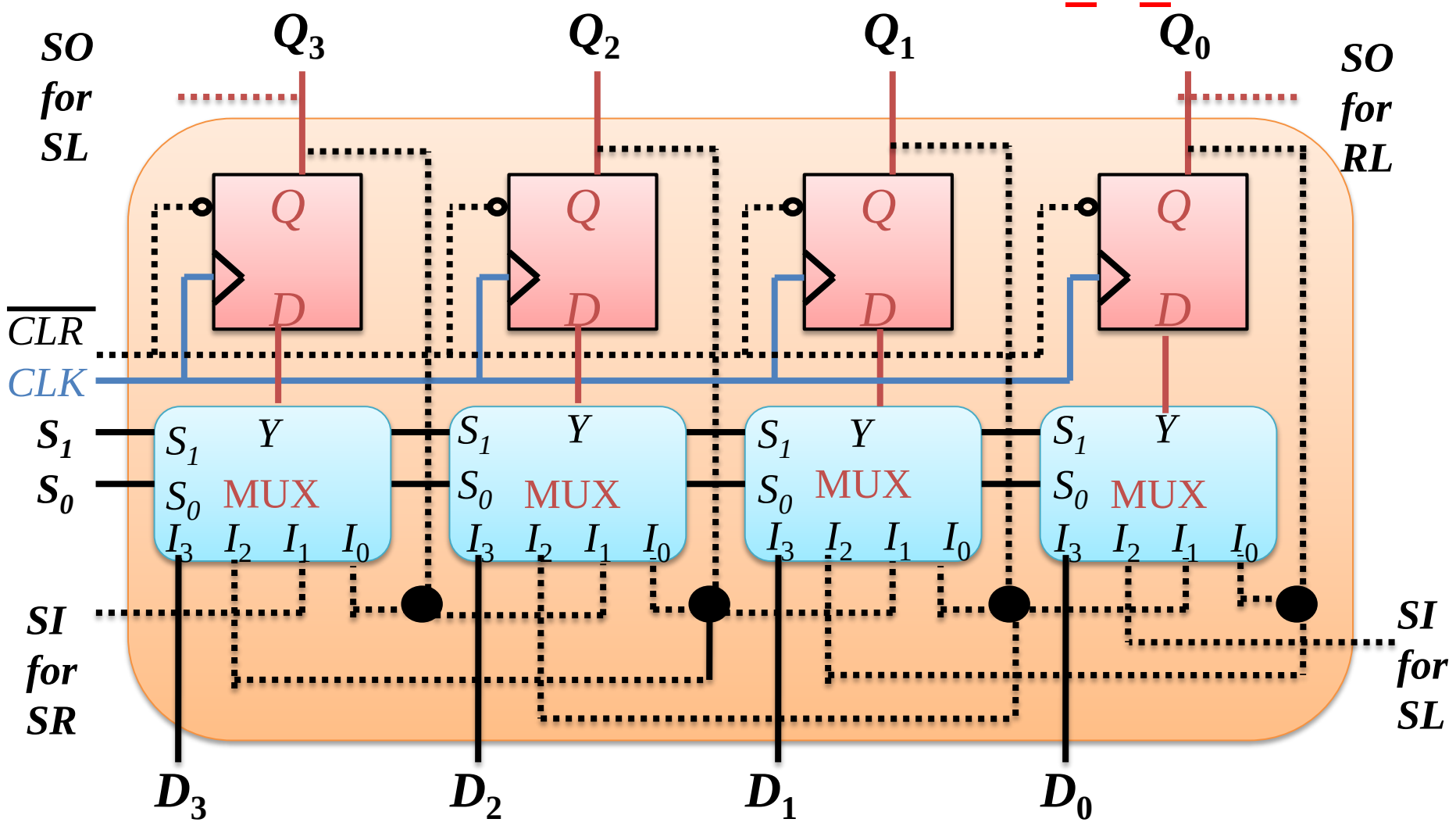
Universal Shift Register: S_1S_0=01



Universal Shift Register: $S_1S_0=10$



Universal Shift Register: $S_1S_0=11$



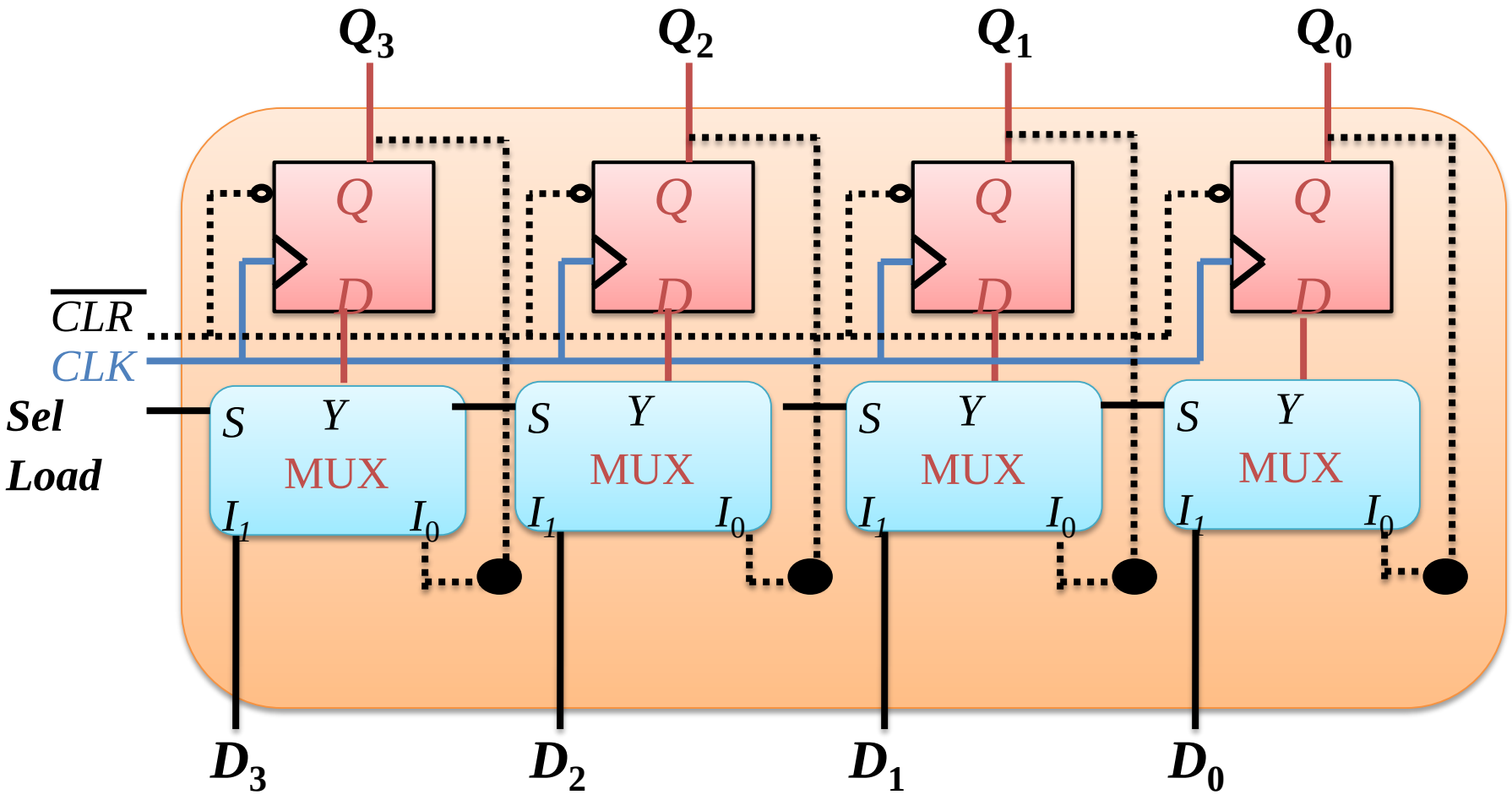
Design of Multifunction Register

- M function N bit Register
  - Last example : 4 functions and 4 bits register
- N number of $\boldsymbol{ceil[Log_2M]x2^{ceil[Log2M]}}$ size Multiplexor (Mux)
  - For 4 function 2x4 Mux
  - For 5 function 3x8 Mux
- Extra functions of Mux need to shunted (ignored) to do no work
  - For 5 function need a 3x8 Mux, we need to shunt 8-5=3 lines shunted (ignored) to do no work

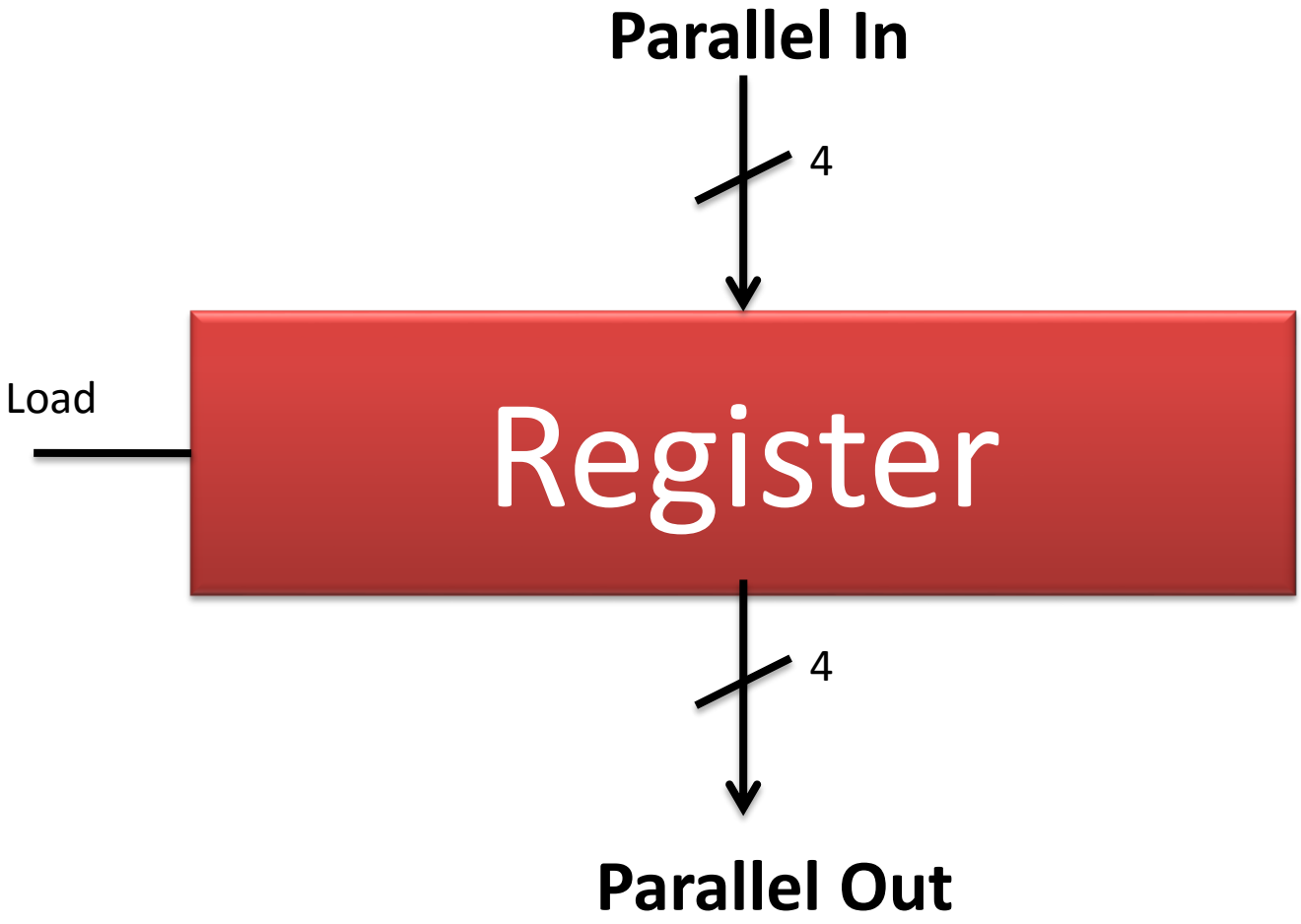
Memory

- Universal register : One register with many functionality
- Many registers clubbed with less functionality
  - No Shift
- Parallel load (In) or Parallel Out to a specific register
- Multiplexors used for selection of resisters

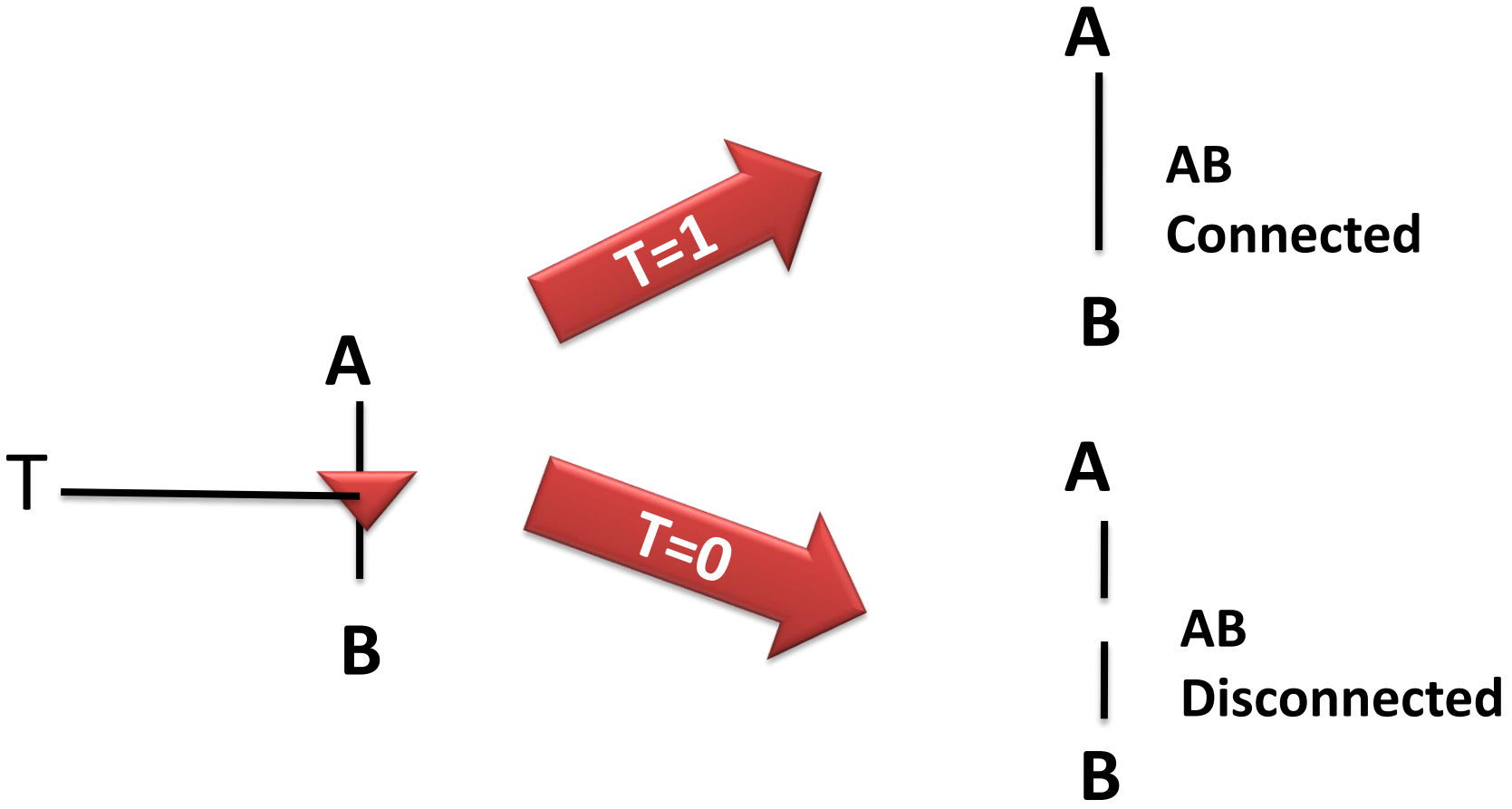
Register(PIPO)



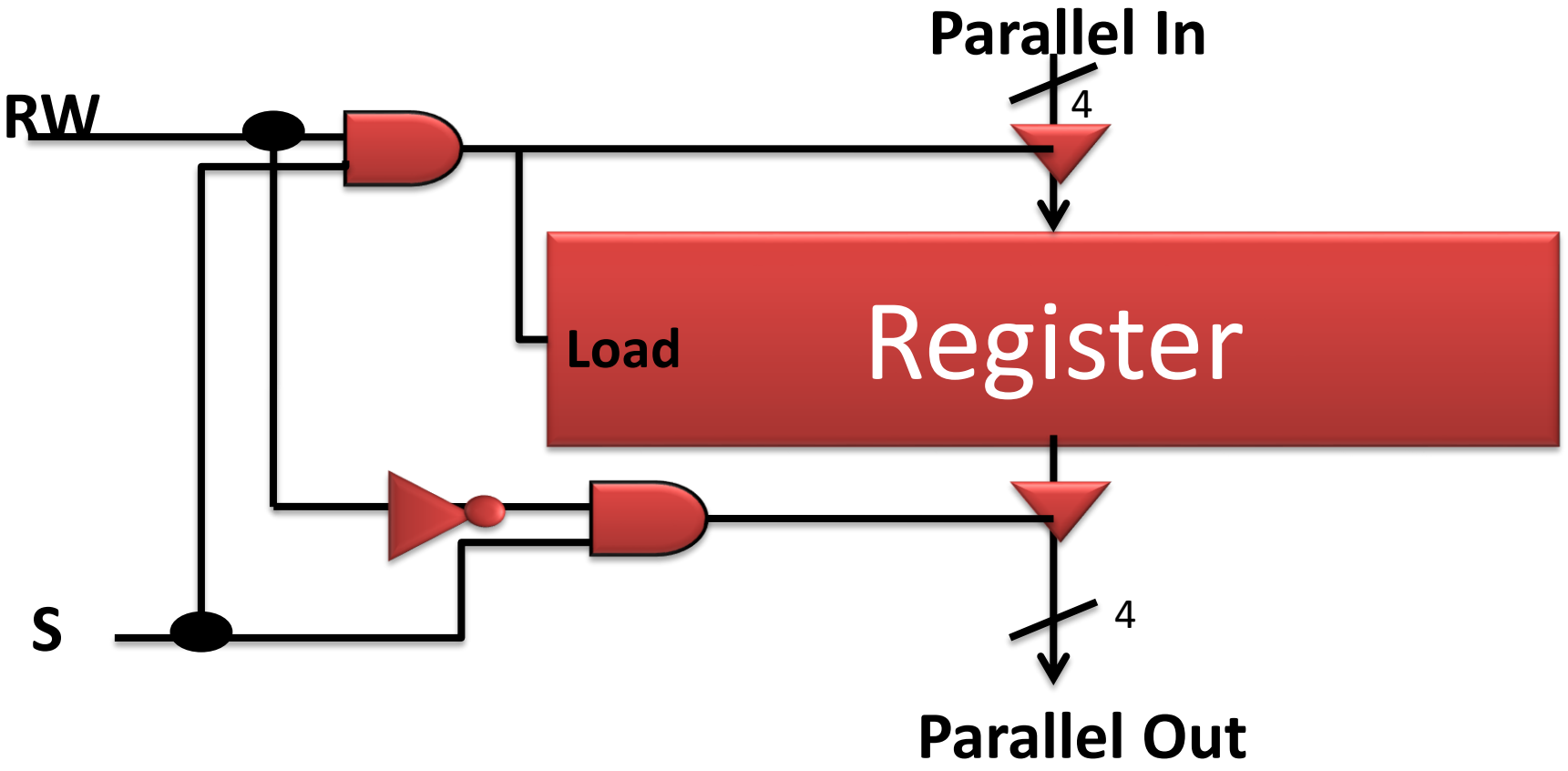
Register(PIPO)



Tri-state Buffer



Register with Read and write control

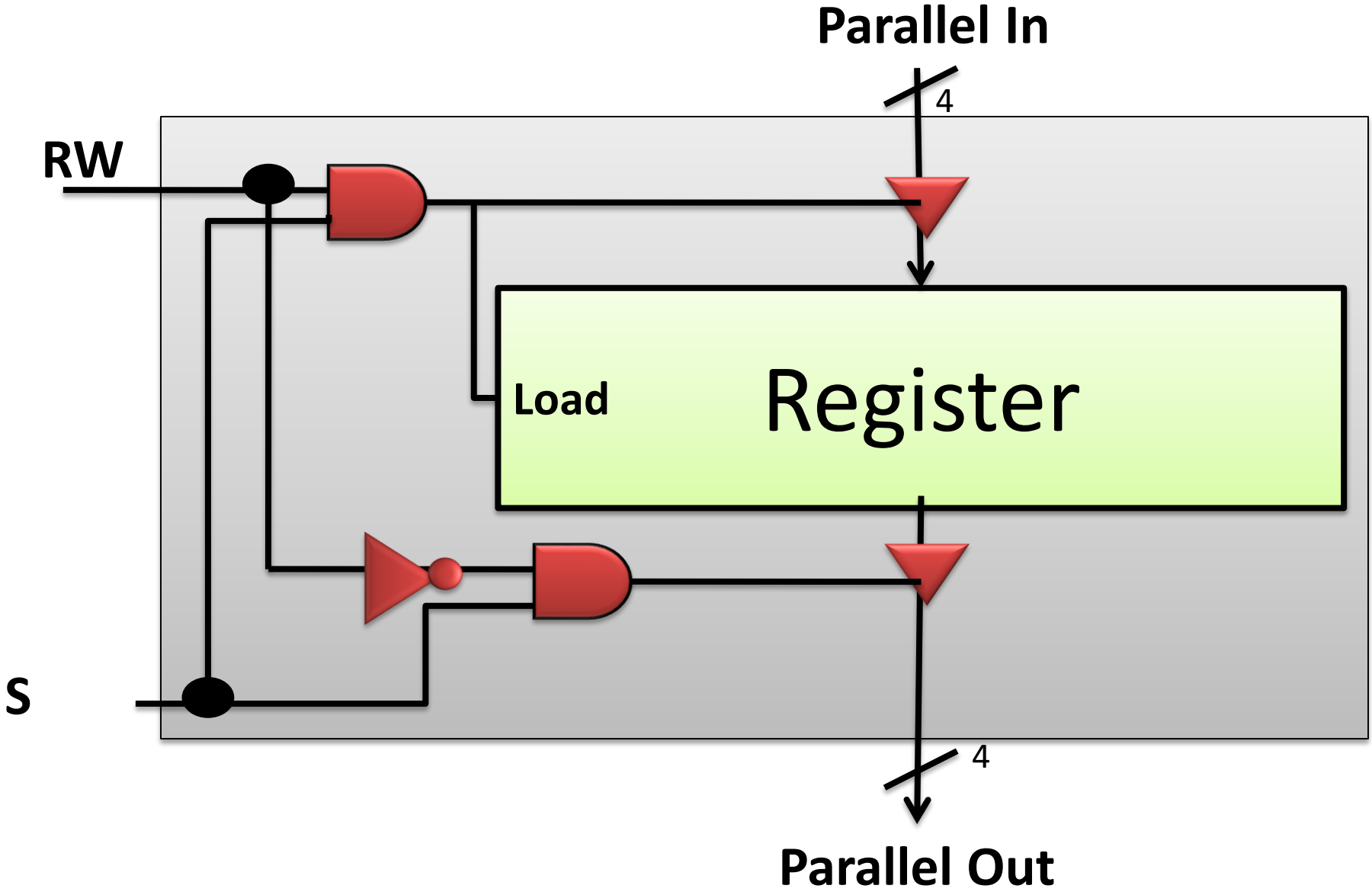


If S =0, TS1 and TS2 will be open

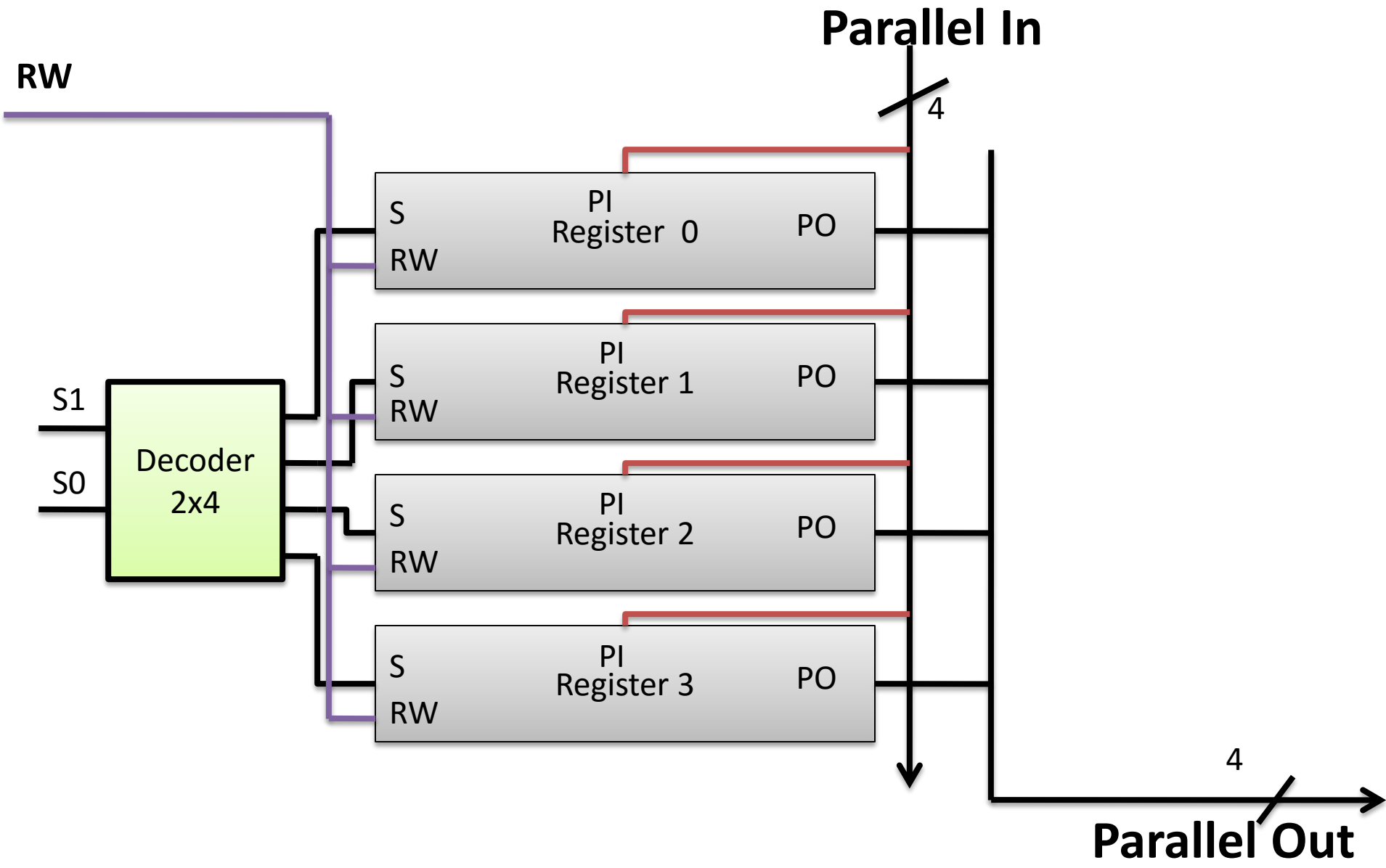
S=1, RW=1 , Write to Register 4 bit of data

S=1, RW=0, Read data from Register

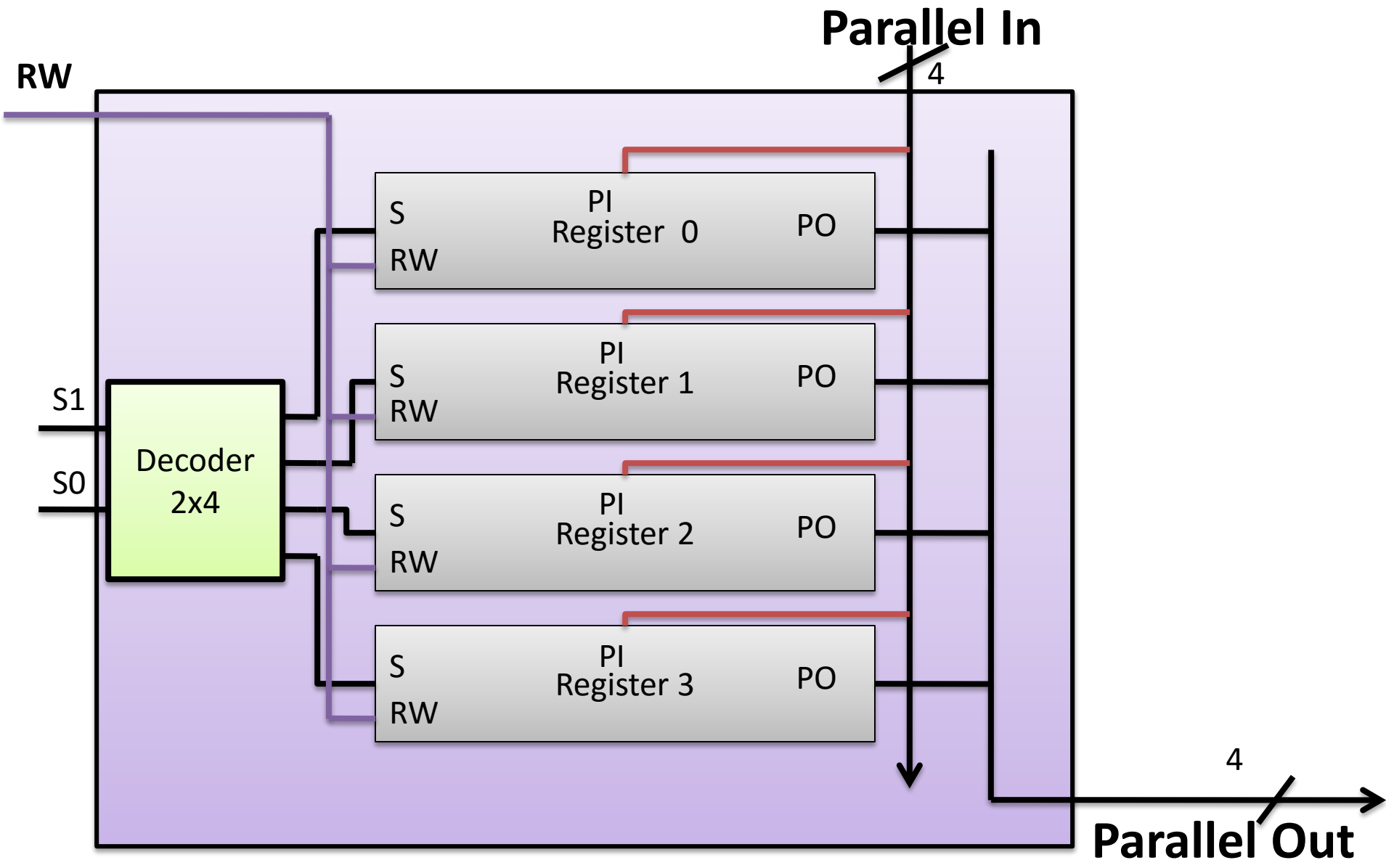
Register with Read and write control



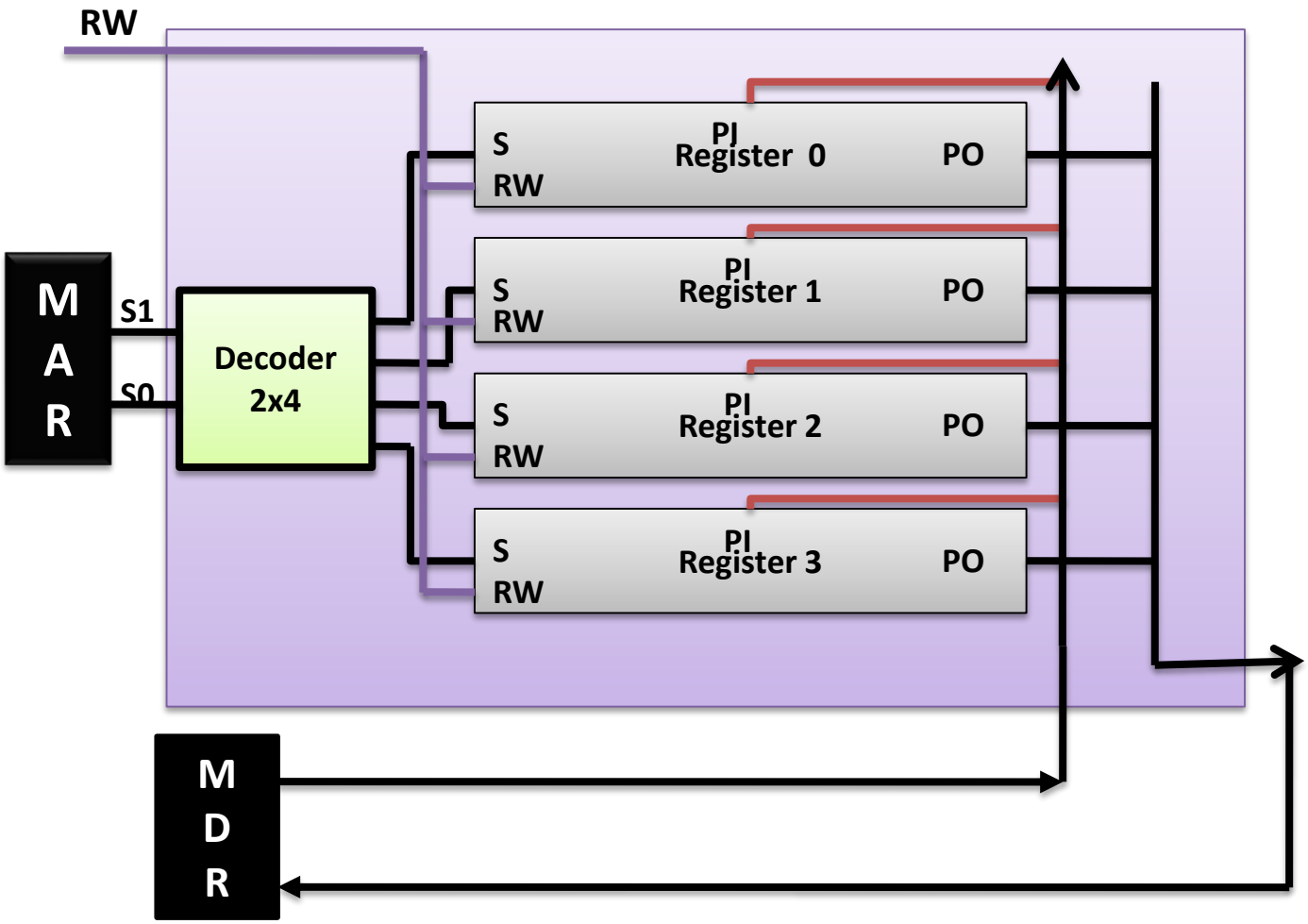
Many Register with RW control



Many Register with RW control



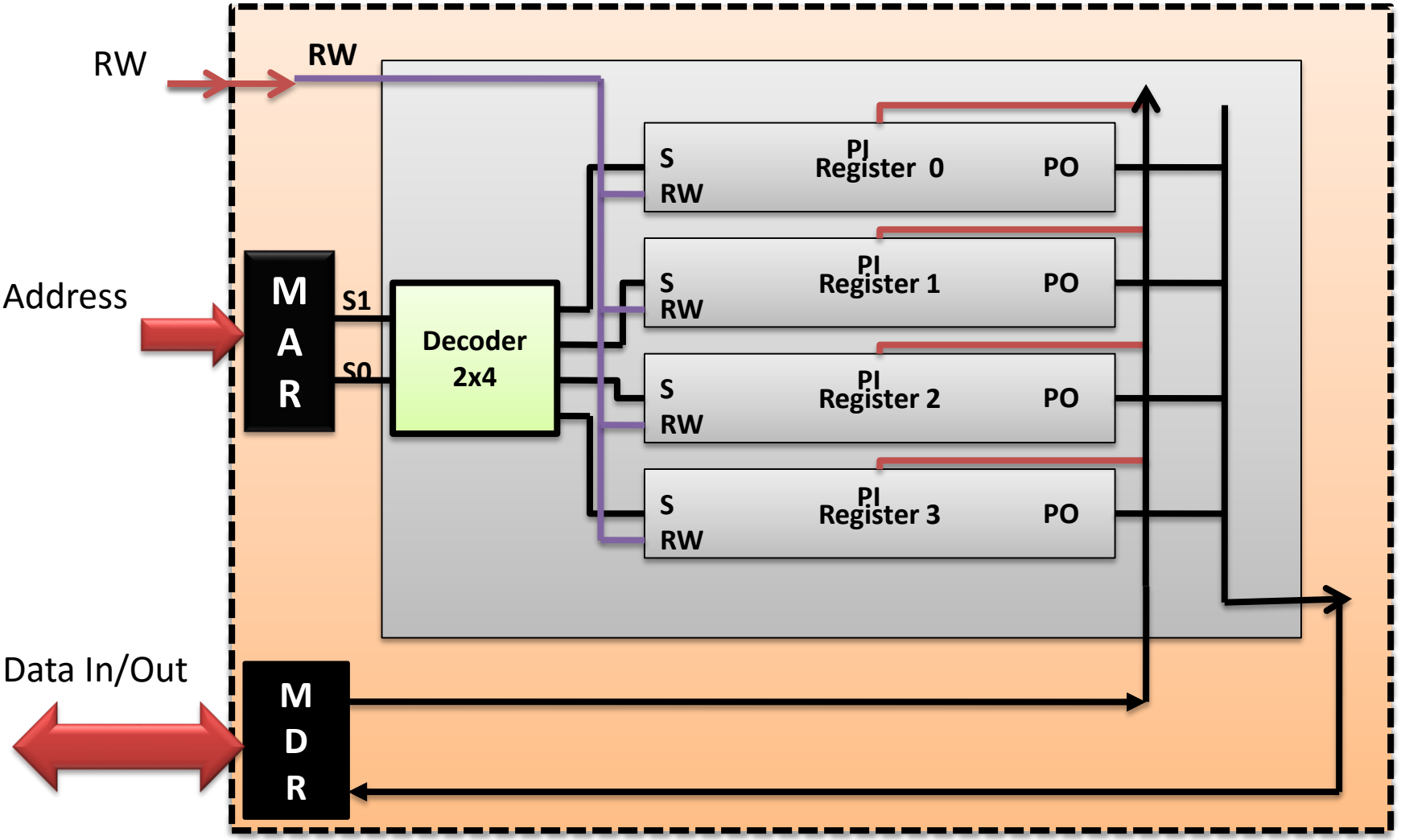
Memory with MAR and MDR



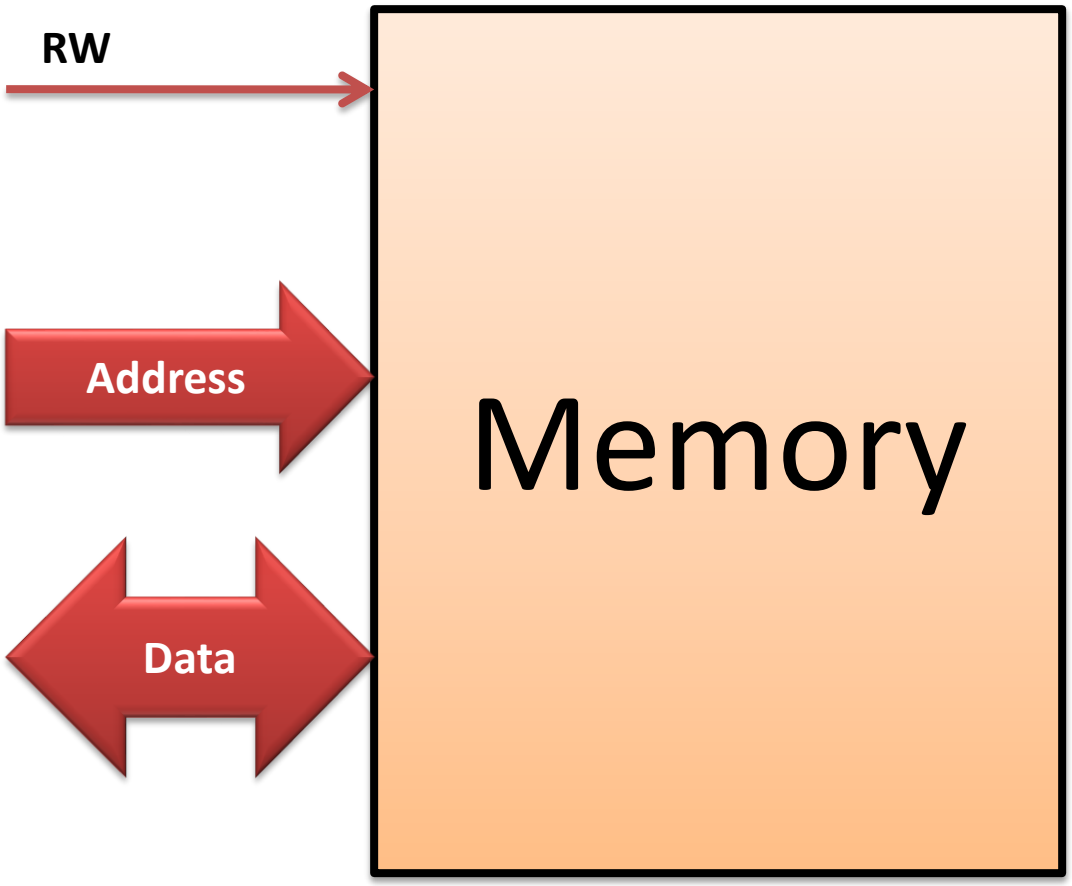
MAR: memory Address Register

MDR: Memory Data Register

Many Register with RW control: 4 four bit word Memory



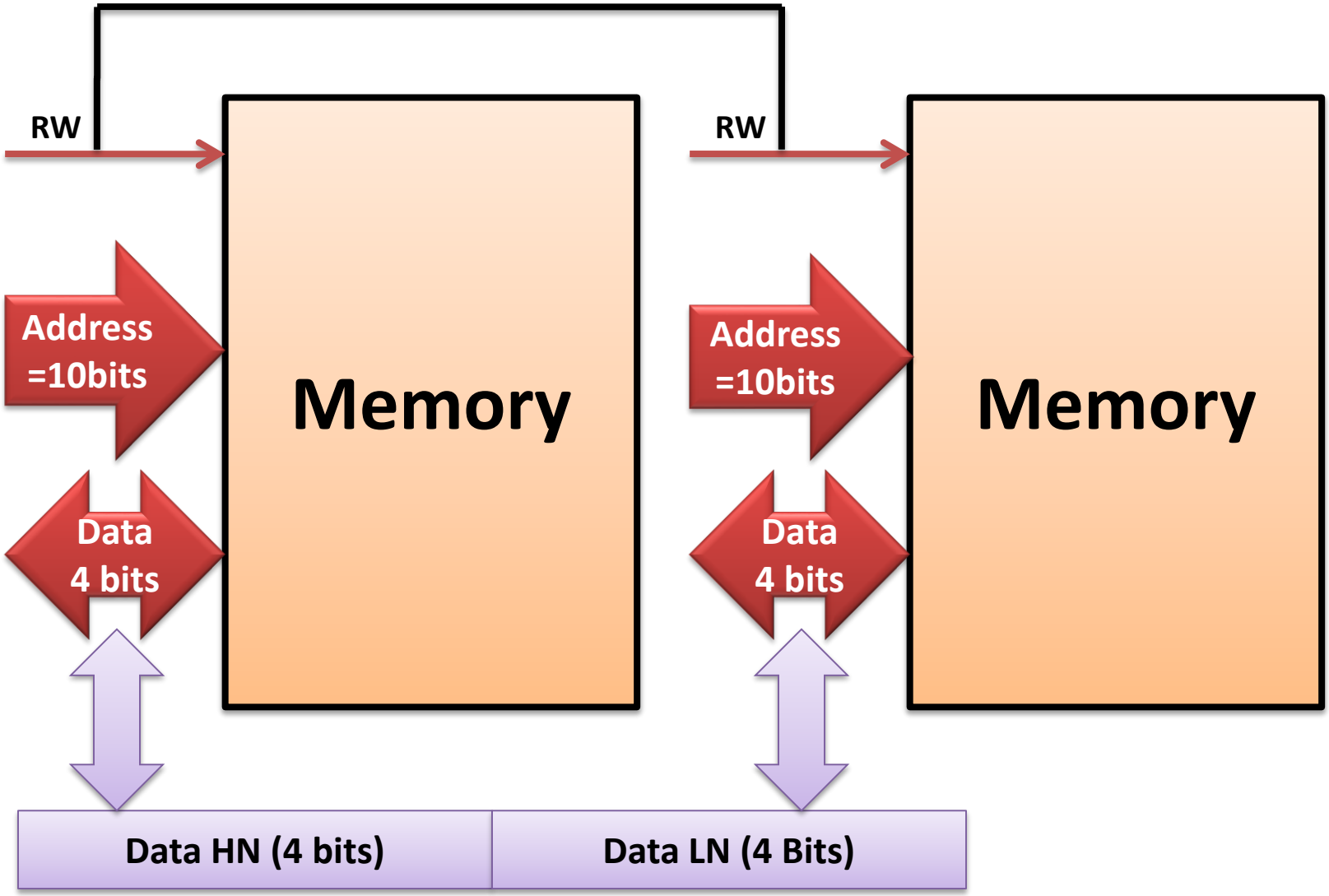
General Memory



Memory : Locations & Word size

- ADDRESS decides the number of memory location
- MDR size : decides the WordSize
- Number of registers : 2+NumLocations
  - $2 + 2^{\text{AddressSize}}$
  - 1 for MBR, 1 for MAR
- NumFF= MARsize+MDRsize+NumLoc*MDRSize

Designing 1KB Mem using two 1KN Mem



Thanks