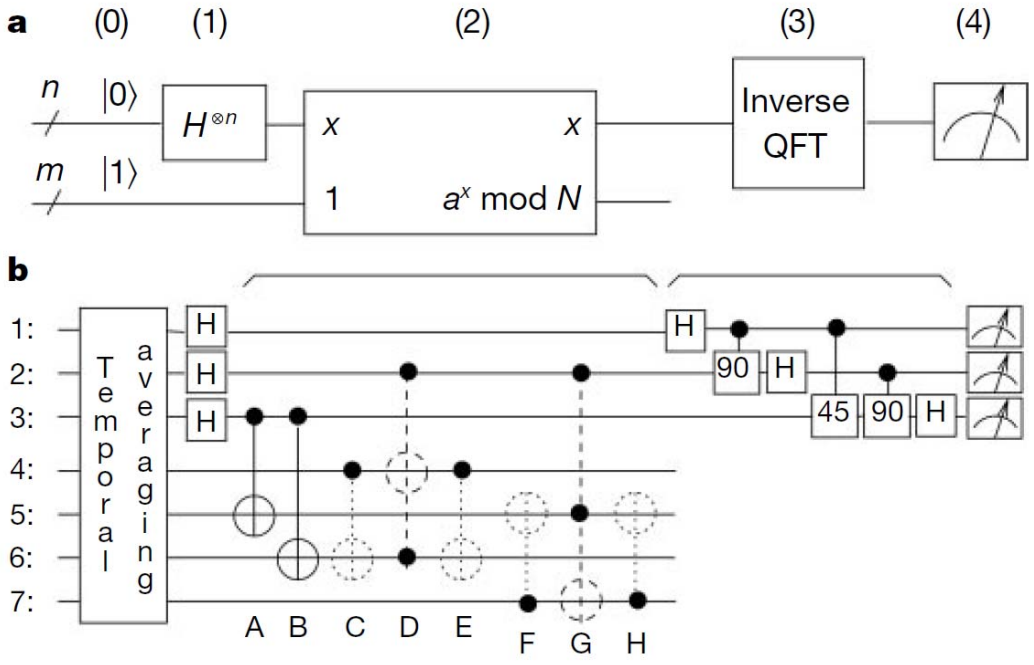
Switching Circuits & Logic Design

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

§18 Circuits for Arithmetic Operations



http://www.nature.com/nature/journal/v414/n6866/fig_tab/414883a_F1.html

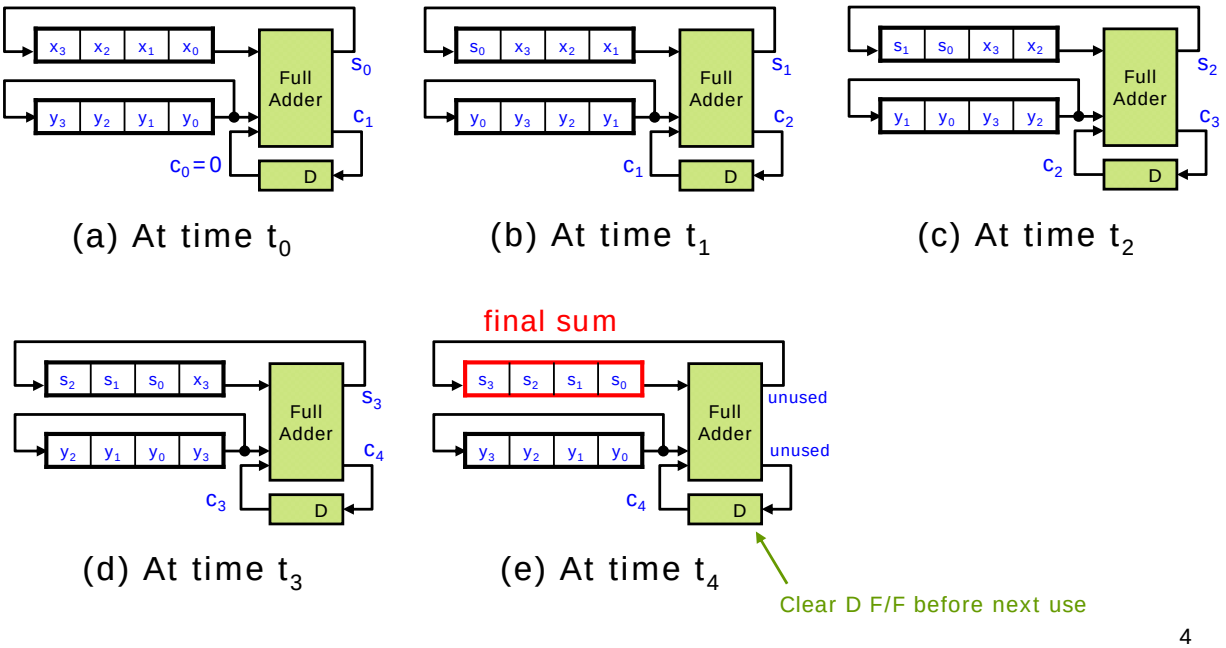
Outline

How to use a sequential circuit to control a sequence of operations in a digital system

- Serial adder with accumulator
- Design of a parallel multiplier

Serial Adder with Accumulator Operation

- 4-bit example

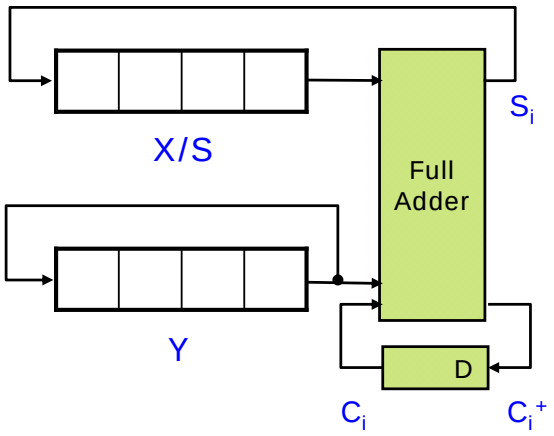


Serial Adder with Accumulator Operation

- 4-bit example

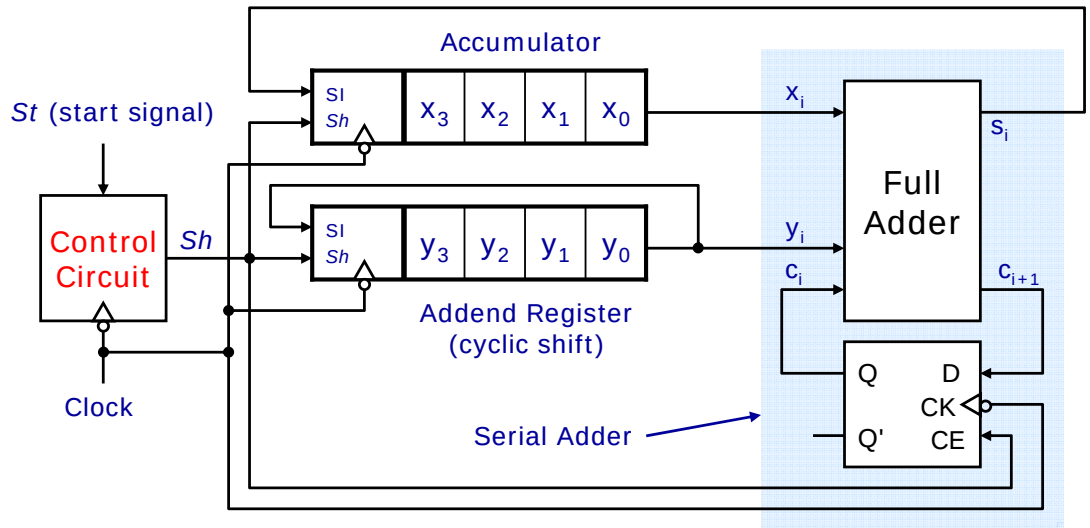
	X $(x_3x_2x_1x_0)$	Y $(y_3y_2y_1y_0)$	C_i	S_i	C_i^+
t_0	0101	0111	0	0	1
t_1	0010	1011	1	0	1
t_2	0001	1101	1	1	1
t_3	1000	1110	1	1	0
t_4	1100	0111	0	(1)	(0)

```
  0 1 0 1
+ 0 1 1 1
---------
  1 1 0 0
```



Serial Adder with Accumulator

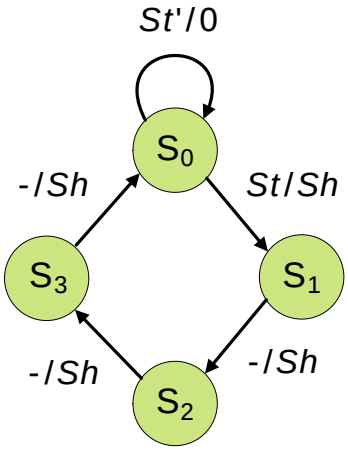
- Block diagram



SI: serial input
St: start signal
Sh: shift signal
} control signals

Serial Adder with Accumulator Control Circuit Design

- State graph and state table



	Next State		Sh	
	St = 0	1	St = 0	1
S_0	S_0	S_1	0	1
S_1	S_2	S_2	1	1
S_2	S_3	S_3	1	1
S_3	S_0	S_0	1	1

Shift 4 times after *St* is activated

Serial Adder with Accumulator Control Circuit Design

- Derivation of control circuit equations

Transition table

	AB	A^+B^+	
		St = 0	1
S_0	00	00	01
S_1	01	10	10
S_2	10	11	11
S_3	11	00	00

State Assignment

AB \ St	0	1
00	0	0
01	1	1
11	0	0
10	1	1

A^+

$D_A = A'B + AB' = A \oplus B$

AB \ St	0	1
00	0	1
01	0	0
11	0	0
10	1	1

B^+

$D_B = StB' + AB'$

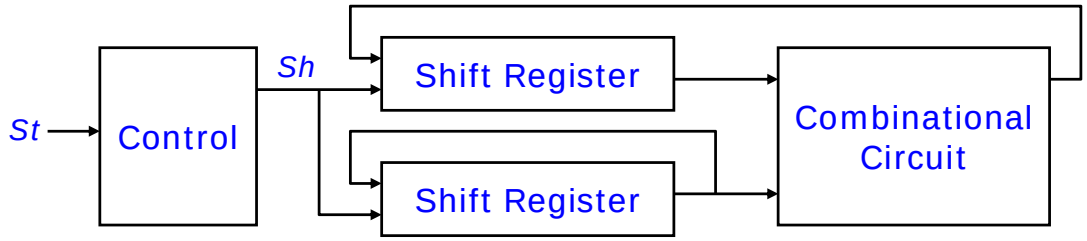
AB \ St	0	1
00	0	1
01	1	1
11	1	1
10	1	1

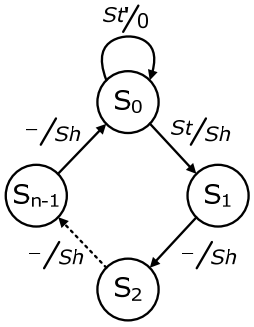
Sh

$Sh = St + A + B$

Serial Adder with Accumulator
Typical Serial Processing Unit

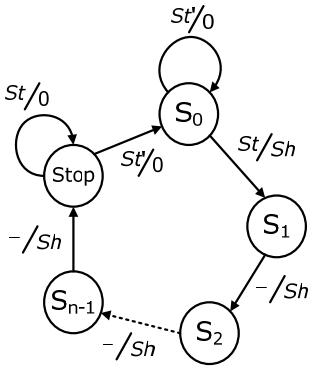
- Typical serial processing unit with n-bit shift registers





If *St* is 1 for only one clock time

St resets to 0 automatically



If *St* remains 1 until after the shifting is completed

Make sure *St* resets to 0 before next start

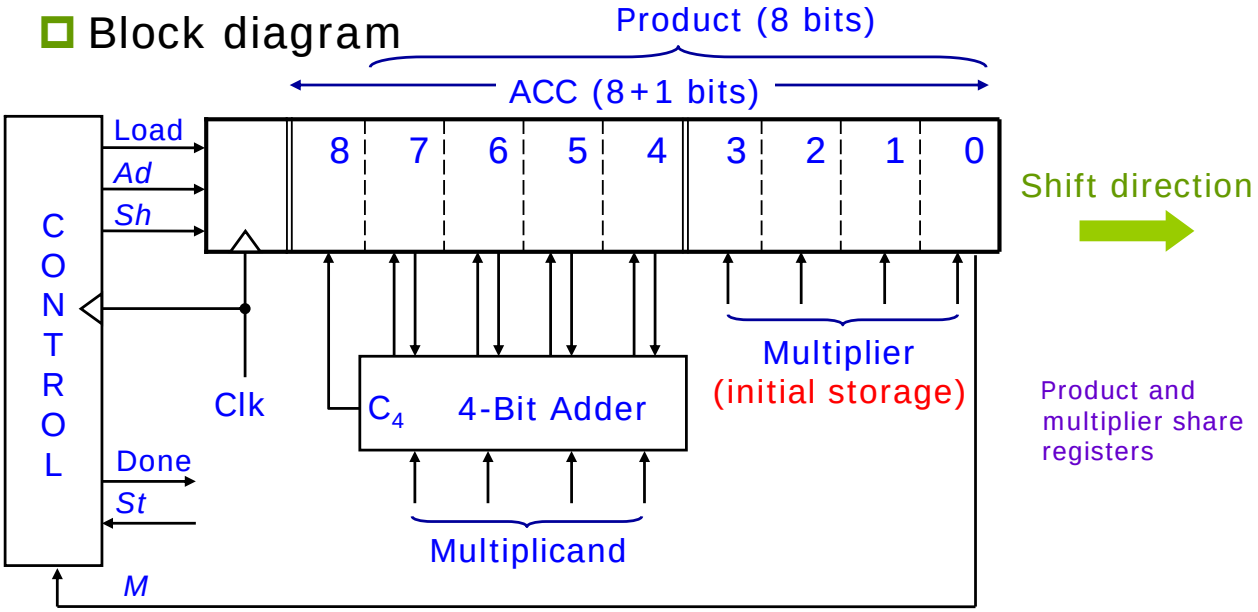
Design of a Parallel Multiplier

- Design a parallel adder for positive binary numbers
  - Require only shifting and adding
  - Add two binary numbers at a time

Multiplicand	→	1101	(13)
Multiplier	→	1011	(11)
Partial Products		1101	
		1101	
		100111	
		0000	
		100111	
		1101	
Product	→	10001111	(143)

Design of a Parallel Multiplier
Parallel Binary Multiplier

- Block diagram



Load: load multiplier to ACC[0:3] and clear ACC[4:8]
Ad: add signal (to store adder outputs in ACC[4:7] and C_4 in ACC[8])
Sh: shift the ACC contents one place to the right
M: current multiplier bit

Design of a Parallel Multiplier
Parallel Binary Multiplier

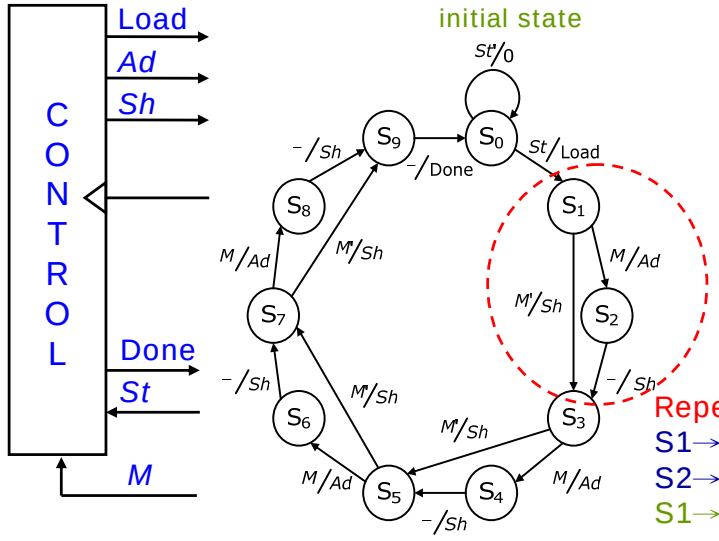
- Operation example

	{product, multiplier}	
initial contents of product register	0 0 0 0 0 1 0 1 **1** ←M	(11)
(add multiplicand because M = 1)	1 1 0 1	(13)
after addition	0 1 1 0 1 1 0 1 1	
after shift	0 0 1 1 0 1 1 0 **1** ←M	
(add multiplicand because M = 1)	1 1 0 1	
after addition	1 0 0 1 1 1 1 0 1	
after shift	0 1 0 0 1 1 1 1 **0** ←M	
(skip addition because M = 0)		
after shift	0 0 1 0 0 1 1 1 **1** ←M	
(add multiplicand because M = 1)	1 1 0 1	
after addition	1 0 0 0 1 1 1 1 1	
after shift (final answer)	0 1 0 0 0 1 1 1 1	(143)

Dividing line between product and multiplier

Design of a Parallel Multiplier
Control Circuit Design

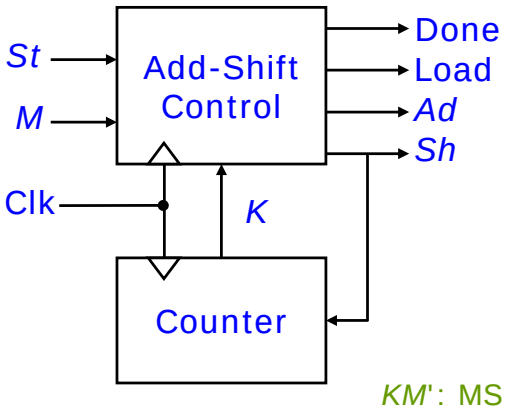
- Method 1 (direct implementation)



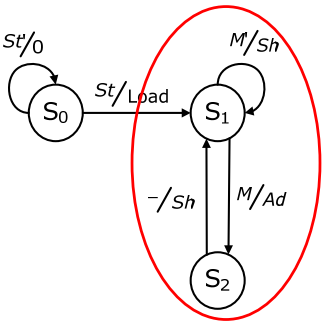
Repeating 1-bit multiplication:
S1→S2: M=1, add
S2→S3: right shift
S1→S3: M=0, right shift without adding

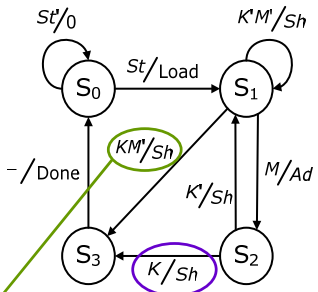
Design of a Parallel Multiplier
Control Circuit Design

- Method 2 (use counter, fewer states)
  - Introduce signal *K* for counting completion



1-bit multiplication





KM': MSB of multiplier is 0, shift and goto S_3

K: addition finished for MSB, shift and goto S_3

How many states in total (including add-shift control and counter)?

Design of a Parallel Multiplier
Control Circuit Design

Method 2 (cont'd)

Operation example for 1101×1011

Time	State	Counter	Product Register	*St*	*M*	*K*	Load	*Ad*	*Sh*	Done
t_0	S_0	00	000000000	0	0	0	0	0	0	0
t_1	S_0	00	000000000	1	0	0	1	0	0	0
t_2	S_1	00	000001011	0	1	0	0	1	0	0
t_3	S_2	00	011011011	0	1	0	0	0	1	0
t_4	S_1	01	001101101	0	1	0	0	1	0	0
t_5	S_2	01	100111101	0	1	0	0	0	1	0
t_6	S_1	10	010011110	0	0	0	0	0	1	0
t_7	S_1	11	001001111	0	1	1	0	1	0	0
t_8	S_2	11	100011111	0	1	1	0	0	1	0
t_9	S_3	00	010001111	0	1	0	0	0	0	1