High Speed Division

- Use High Speed Multiplier
- Follows Newton-Raphson iteration method
- Can find correct answer after few iterations

High Speed Divide

$$\frac{A}{B}$$

High Speed Divide

$$\frac{A \times f_0}{B \times f_0}$$

High Speed Divide

$$\frac{A \times f_0}{B \times f_0}$$

let $B = 1 - x$, then

$x = 1 - B$, and let

$f_0 = 1 + x = 2 - B$

High Speed Divide

$$\frac{A \times f_0}{B \times f_0}$$

let $B = 1 - x$, then

$x = 1 - B$, and let

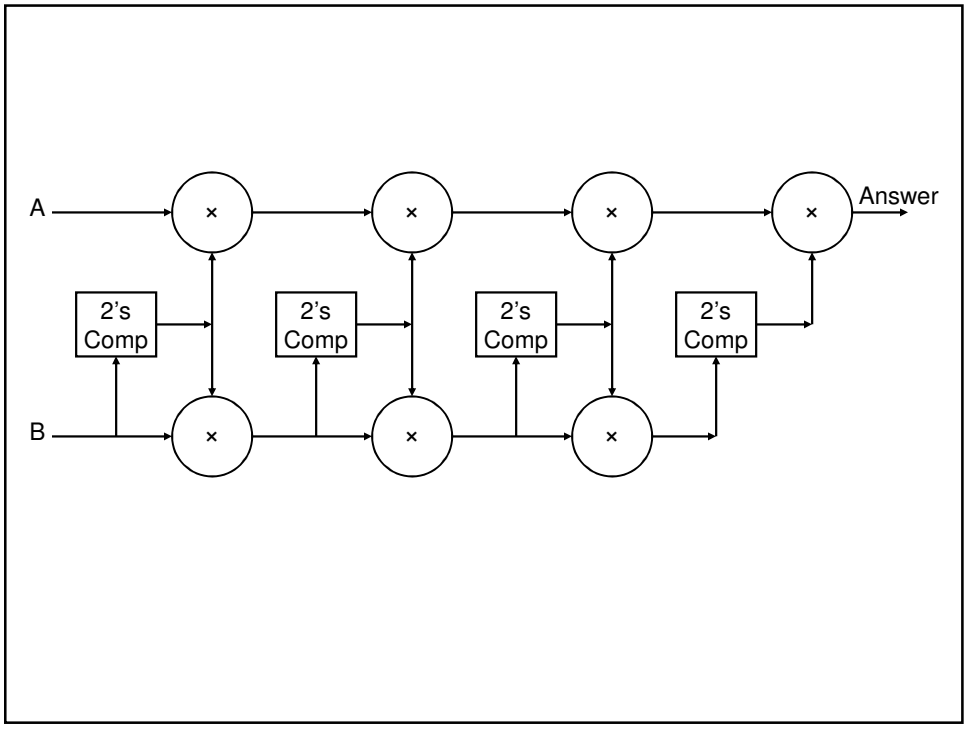
$f_0 = 1 + x = 2 - B$

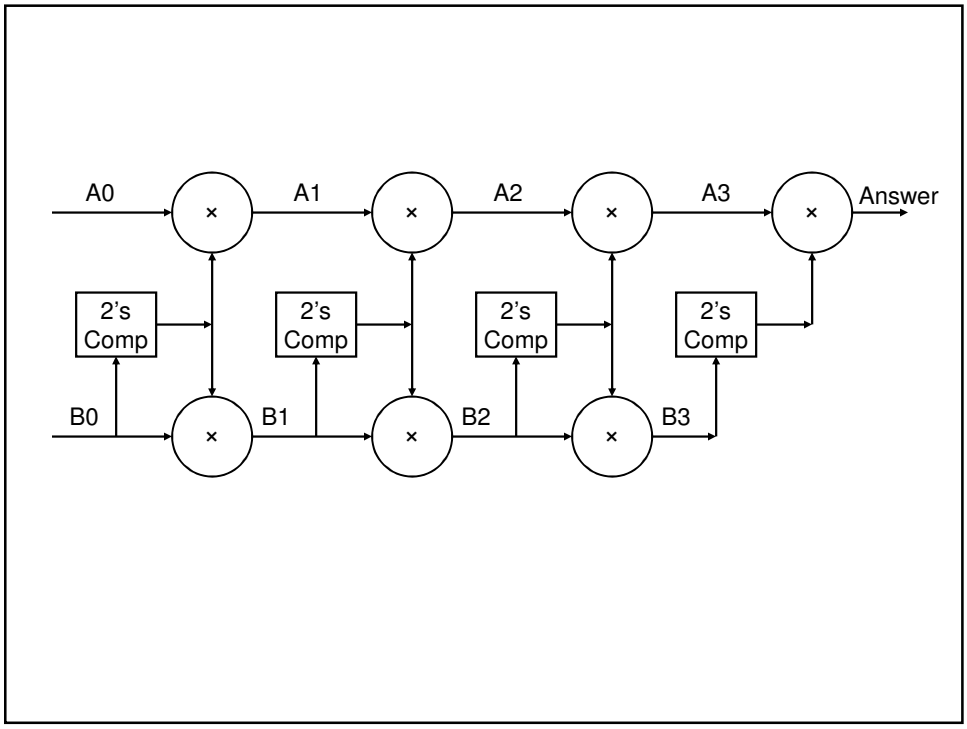
AND:

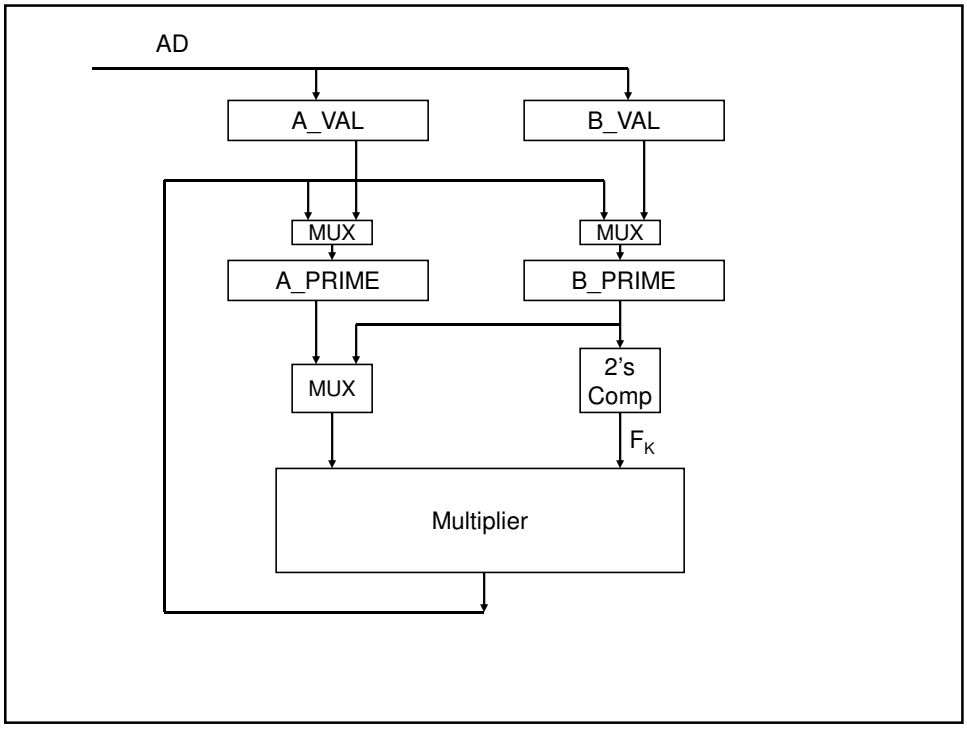
$$B \times f0 = (1 - x) \times (1 + x)$$
$$= 1 - x^2$$

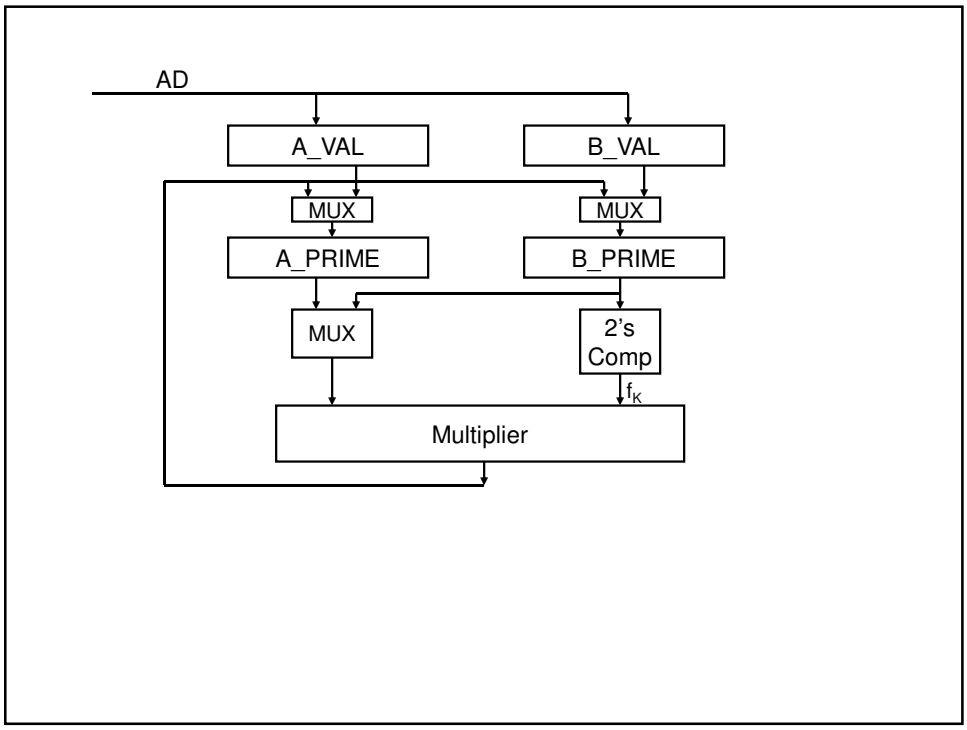
High Speed Divide

$$\frac{A \times f_0 \times f_1 \times f_2 \times f_3 \times f_4}{B \times f_0 \times f_1 \times f_2 \times f_3 \times f_4}$$


A
B
×
2's Comp
Answer


A0
A1
A2
A3
B0
B1
B2
B3
×
2's Comp
Answer


AD
A_VAL
B_VAL
MUX
MUX
A_PRIME
B_PRIME
MUX
2's
Comp
F_K
Multiplier


AD
A_VAL
B_VAL
MUX
MUX
A_PRIME
B_PRIME
MUX
2's
Comp
f_K
Multiplier

Step 1: A_VAL <= AD

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step B: A_PRIME <= A_PRIME × F_K ; increment CNT

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step B: A_PRIME <= A_PRIME × F_K ; increment CNT

Step C: B_PRIME <= B_PRIME × F_K

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step B: A_PRIME <= A_PRIME × F_K ; increment CNT

Step C: B_PRIME <= B_PRIME × F_K

Repeat B – C pair until CNT = 5

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step B: A_PRIME <= A_PRIME × F_K ; increment CNT

Step C: B_PRIME <= B_PRIME × F_K

Repeat B – C pair until CNT = 5

Note: At this point, Answer is:

(SignA ⊕ SignB) & (ExpA – ExpB) & A_PRIME

Step 1: A_VAL <= AD

Step 2: B_VAL <= AD

Step 3: Start algorithm – ask for AD <= divided value

Algorithm:

Step A: A_PRIME <= A_VAL ; B_PRIME <= B_VAL ; Clear CNT

Step B: A_PRIME <= A_PRIME × F_K ; increment CNT

Step C: B_PRIME <= B_PRIME × F_K

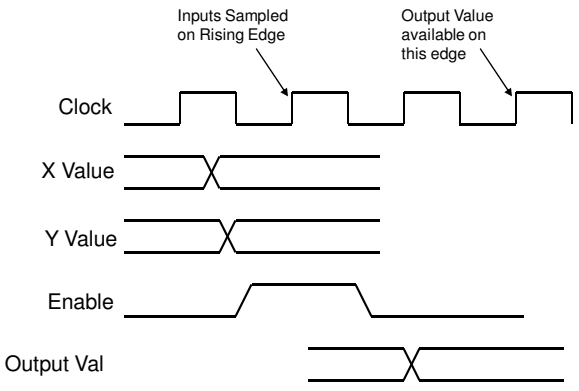
Repeat B – C pair until CNT = 5

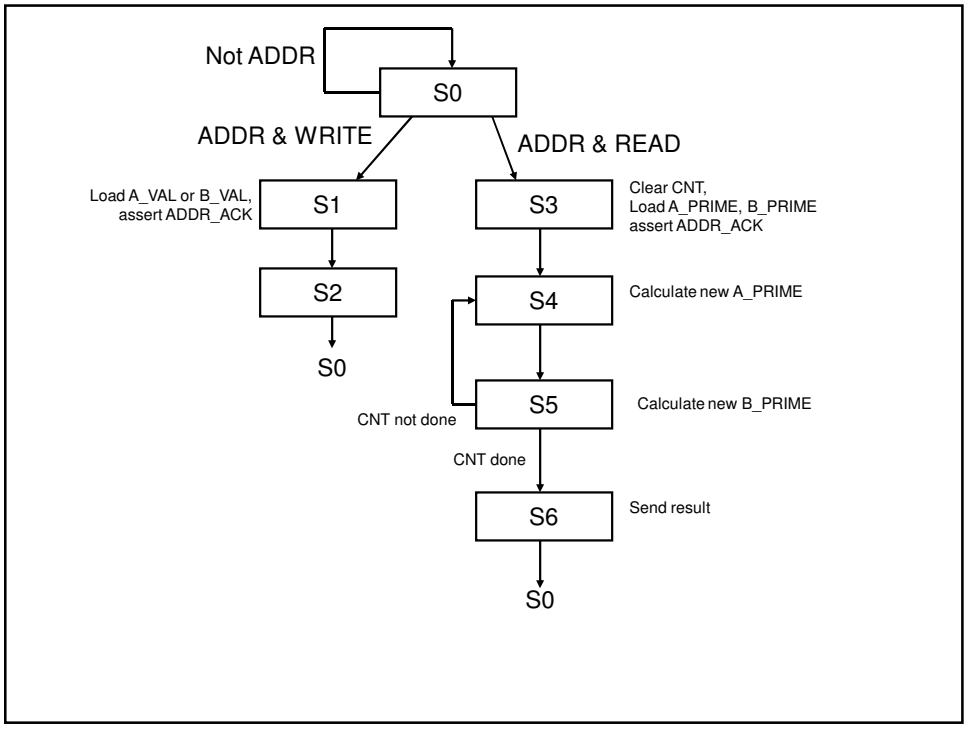
Note: At this point, Answer is:

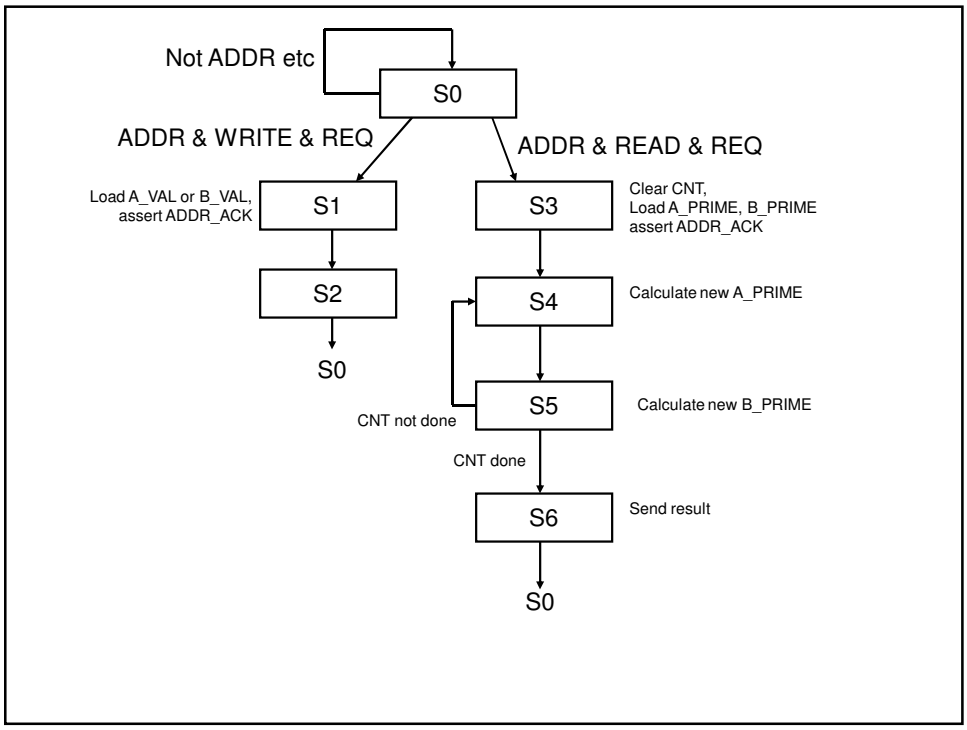
(SignA ⊕ SignB) & (ExpA – ExpB) & A_PRIME

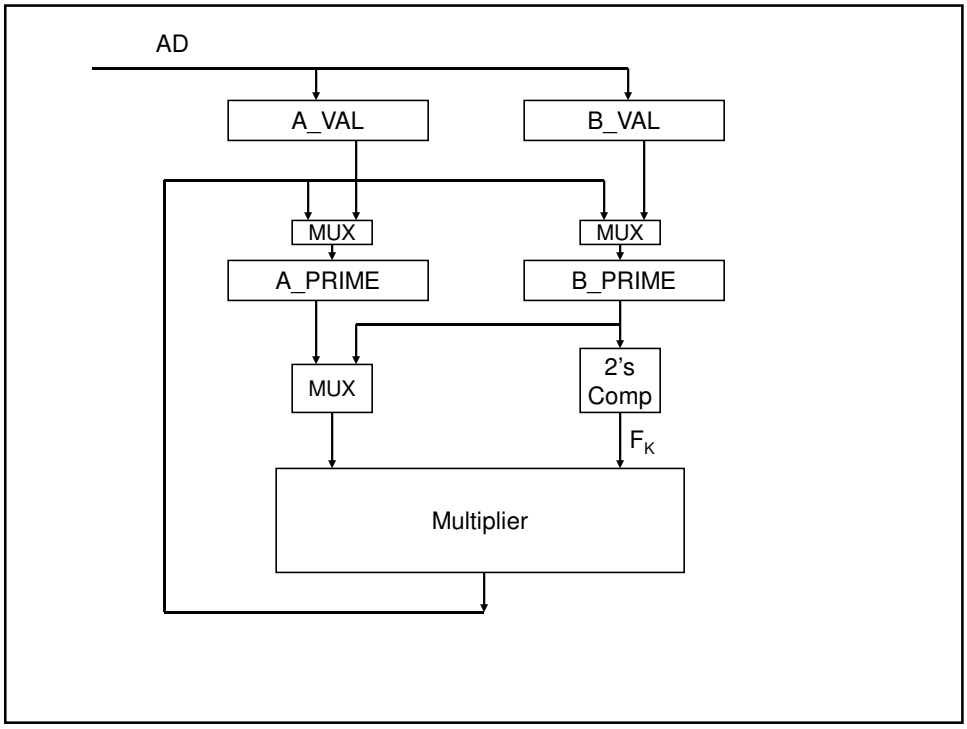
Step D: AD <= Answer ; assert DATA_H

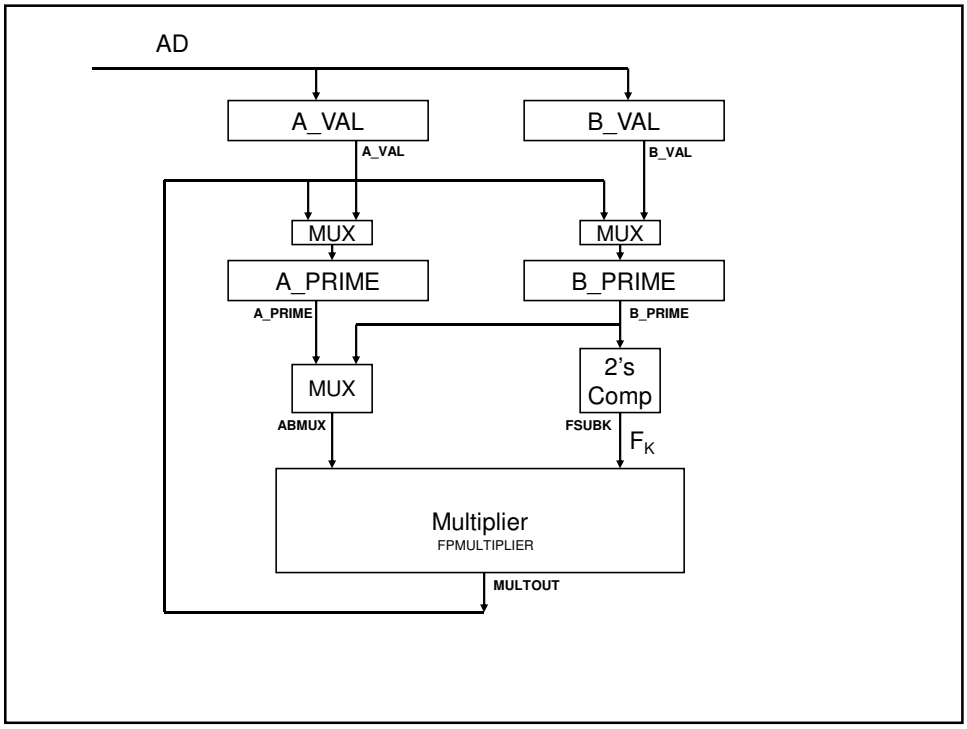
Step E: release AD ; release DATA_H

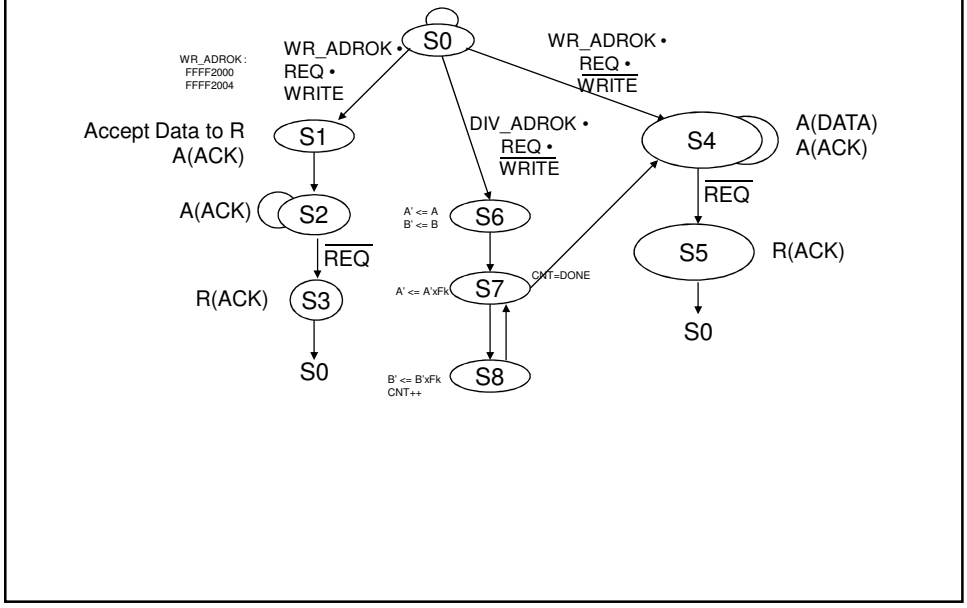


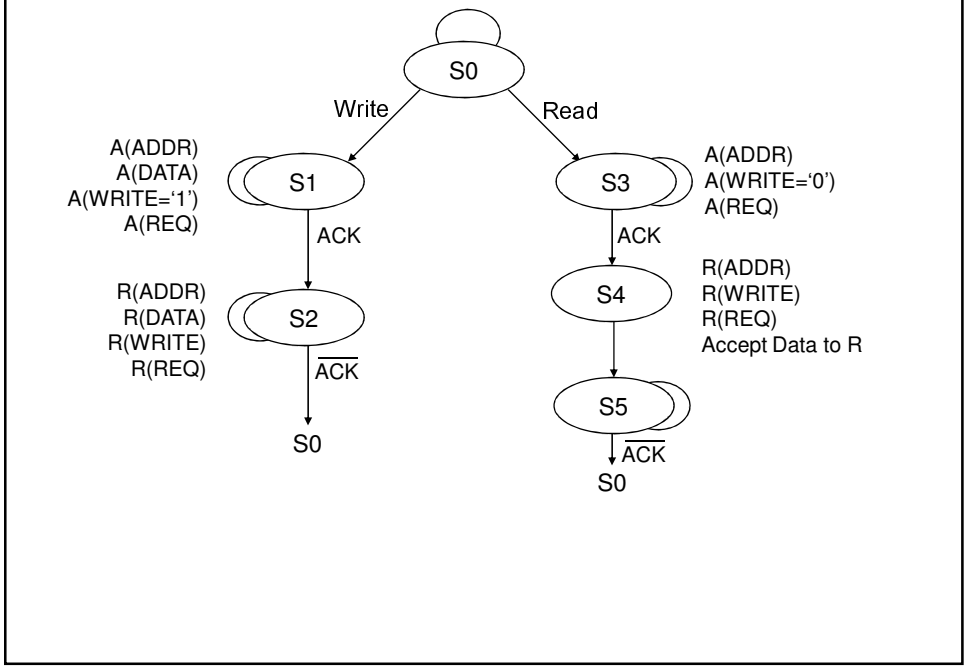

Not ADDR
S0
ADDR & WRITE
ADDR & READ
Load A_VAL or B_VAL,
assert ADDR_ACK
S1
S3
Clear CNT,
Load A_PRIME, B_PRIME
assert ADDR_ACK
S2
S4
Calculate new A_PRIME
S0
S5
Calculate new B_PRIME
CNT not done
CNT done
S6
Send result
S0


Not ADDR etc
S0
ADDR & WRITE & REQ
ADDR & READ & REQ
Load A_VAL or B_VAL,
assert ADDR_ACK
S1
S3
Clear CNT,
Load A_PRIME, B_PRIME
assert ADDR_ACK
S2
S4
Calculate new A_PRIME
S0
S5
Calculate new B_PRIME
CNT not done
CNT done
S6
Send result
S0


AD
A_VAL
B_VAL
MUX
MUX
A_PRIME
B_PRIME
MUX
2's Comp
F_K
Multiplier


AD
A_VAL
B_VAL
A_VAL
B_VAL
MUX
MUX
A_PRIME
B_PRIME
A_PRIME
B_PRIME
MUX
2's Comp
ABMUX
FSUBK
F_K
Multiplier
FPMULTIPLIER
MULTOUT


WR_ADROK :
FFFF2000
FFFF2004
S0
WR_ADROK •
REQ •
WRITE
WR_ADROK •
REQ •
WRITE
Accept Data to R
A(ACK)
S1
DIV_ADROK •
REQ •
WRITE
S4
A(DATA)
A(ACK)
A(ACK)
S2
A' <= A
B' <= B
S6
REQ
S5
R(ACK)
R(ACK)
S3
A' <= A'xFk
S7
CNT=DONE
S0
S0
B' <= B'xFk
CNT++
S8


S0
Write
Read
A(ADDR)
A(DATA)
A(WRITE='1')
A(REQ)
S1
S3
A(ADDR)
A(WRITE='0')
A(REQ)
ACK
ACK
R(ADDR)
R(DATA)
R(WRITE)
R(REQ)
S2
S4
R(ADDR)
R(WRITE)
R(REQ)
Accept Data to R
ACK
S5
S0
ACK
S0

Algorithm:

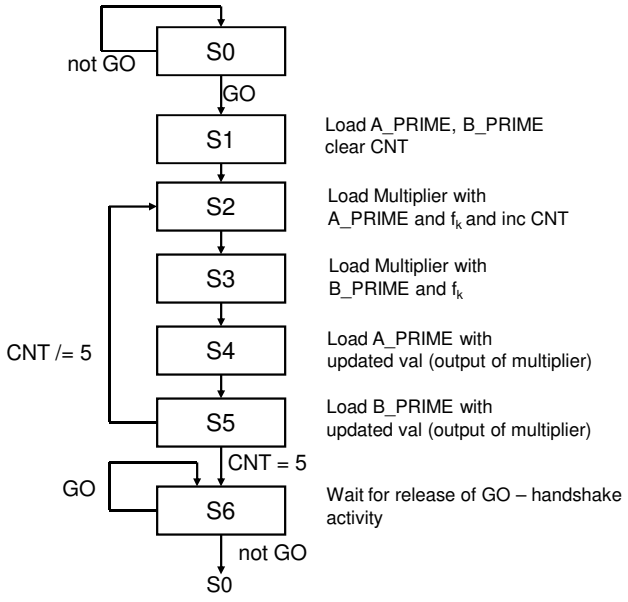
Step A: A_PRIME ← A_VAL ; B_PRIME ← B_VAL ; Clear CNT

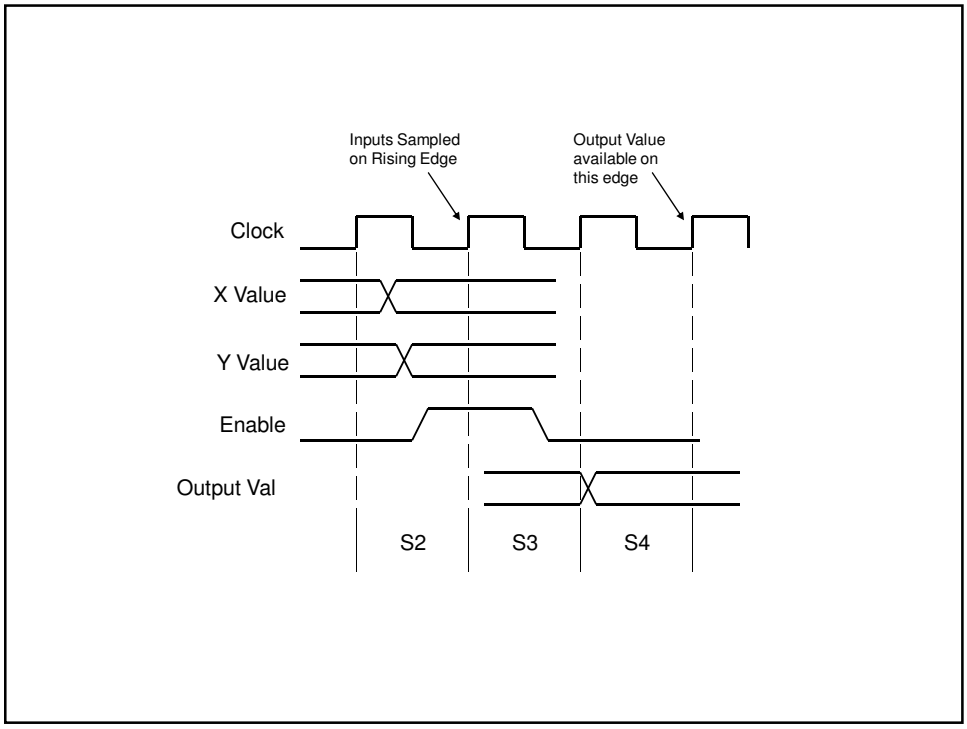
Step B: A_PRIME ← A_PRIME × F_K ; increment CNT

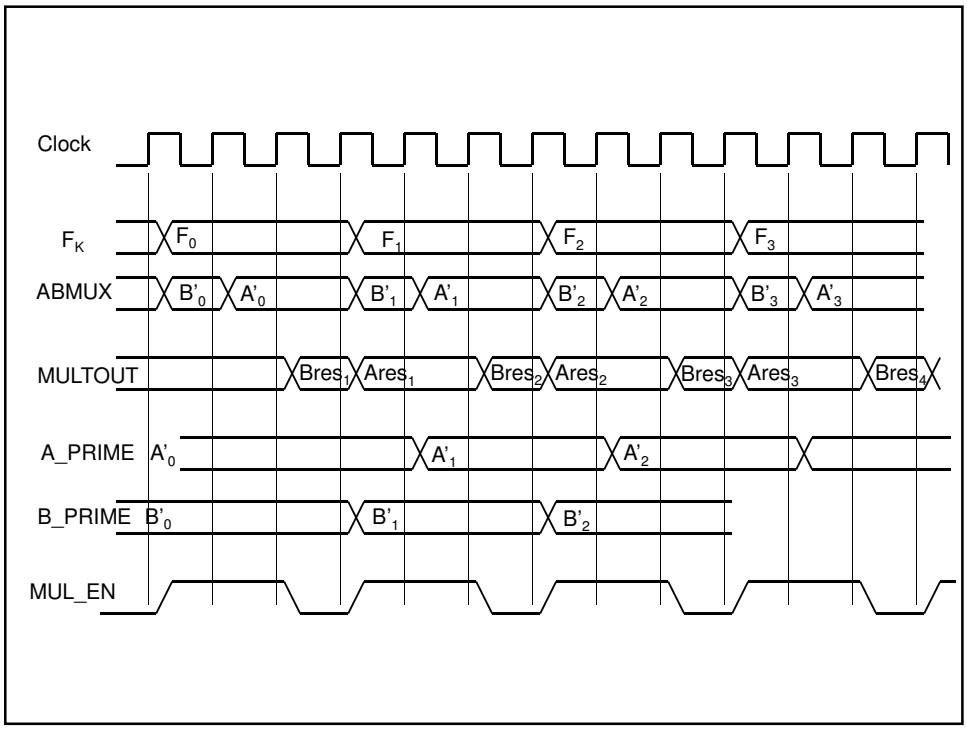
Step C: B_PRIME ← B_PRIME × F_K

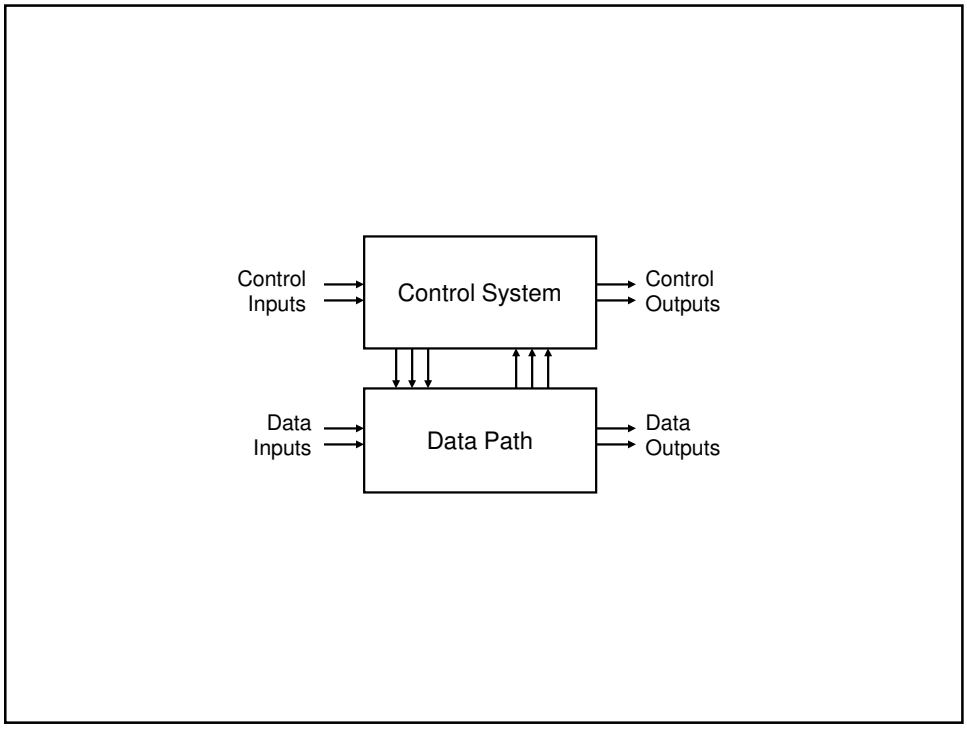
Repeat B – C pair until CNT = 5

Answer is in A_PRIME




Inputs Sampled
on Rising Edge
Output Value
available on
this edge
Clock
X Value
Y Value
Enable
Output Val
S2
S3
S4


Clock
F_K
F_0
F_1
F_2
F_3
ABMUX
B'_0
A'_0
B'_1
A'_1
B'_2
A'_2
B'_3
A'_3
MULTOUT
$Bres_1$
$Ares_1$
$Bres_2$
$Ares_2$
$Bres_3$
$Ares_3$
$Bres_4$
A_PRIME
A'_0
A'_1
A'_2
B_PRIME
B'_0
B'_1
B'_2
MUL_EN


Control
Inputs
Control System
Control
Outputs
Data
Inputs
Data Path
Data
Outputs