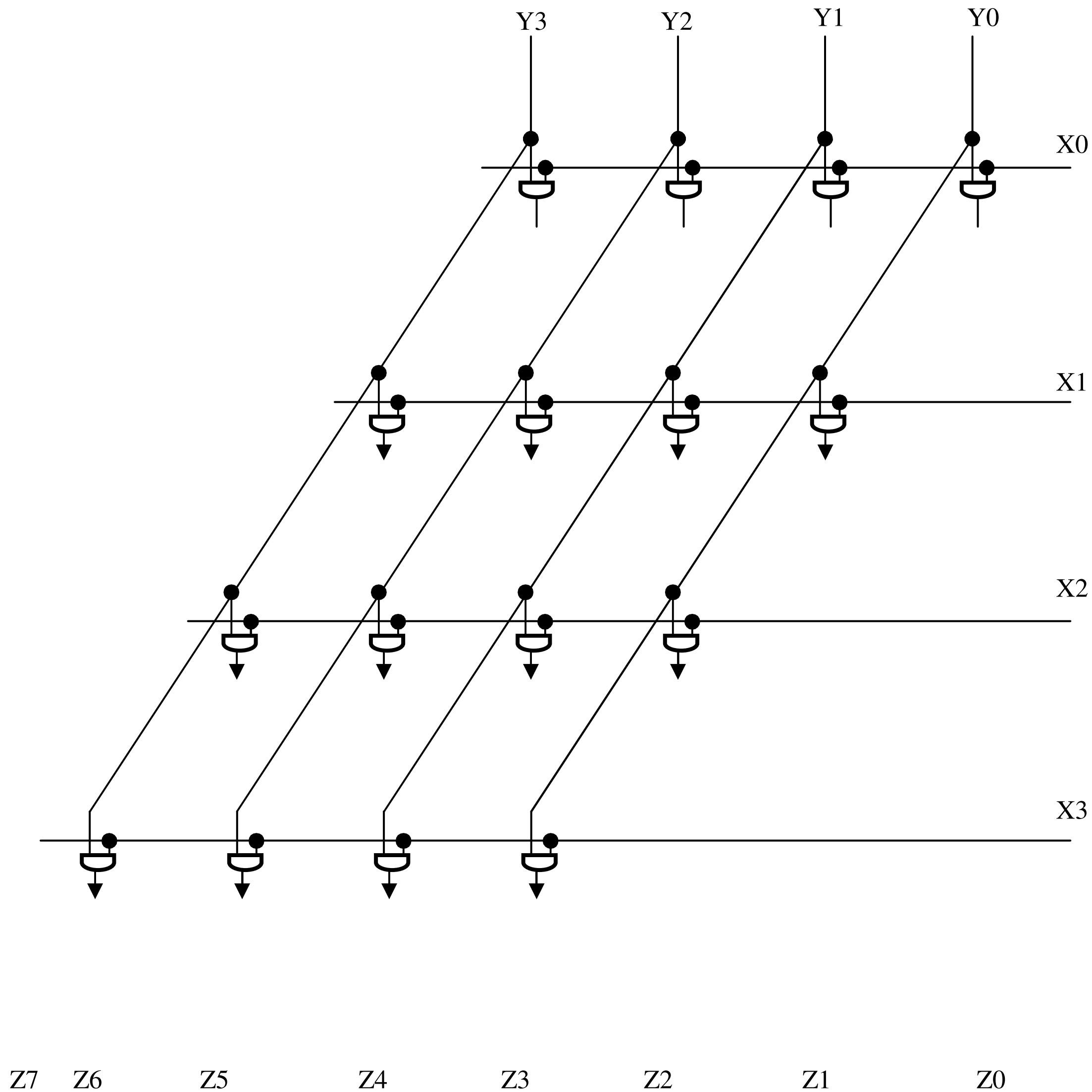
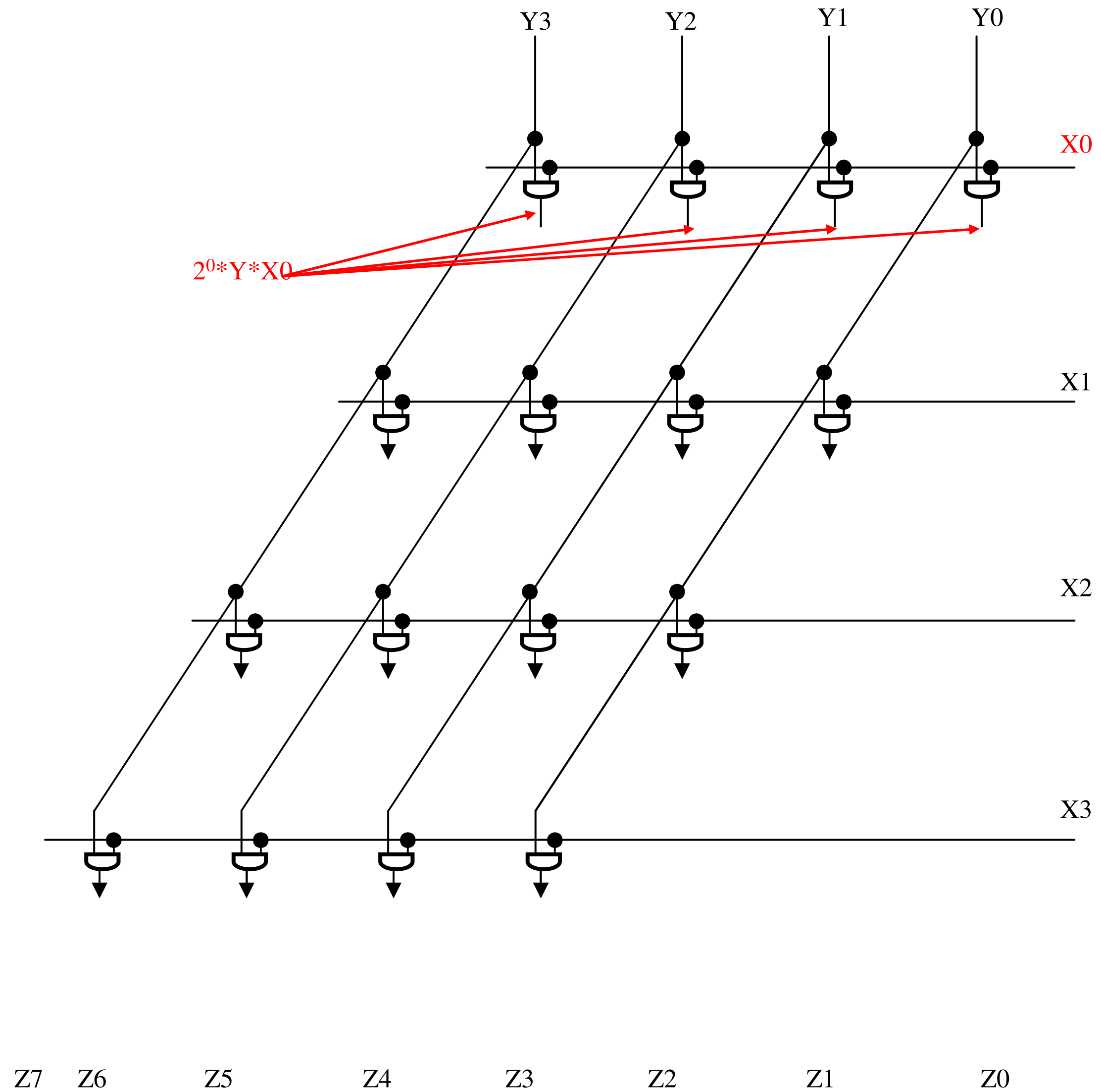
Array Multipliers

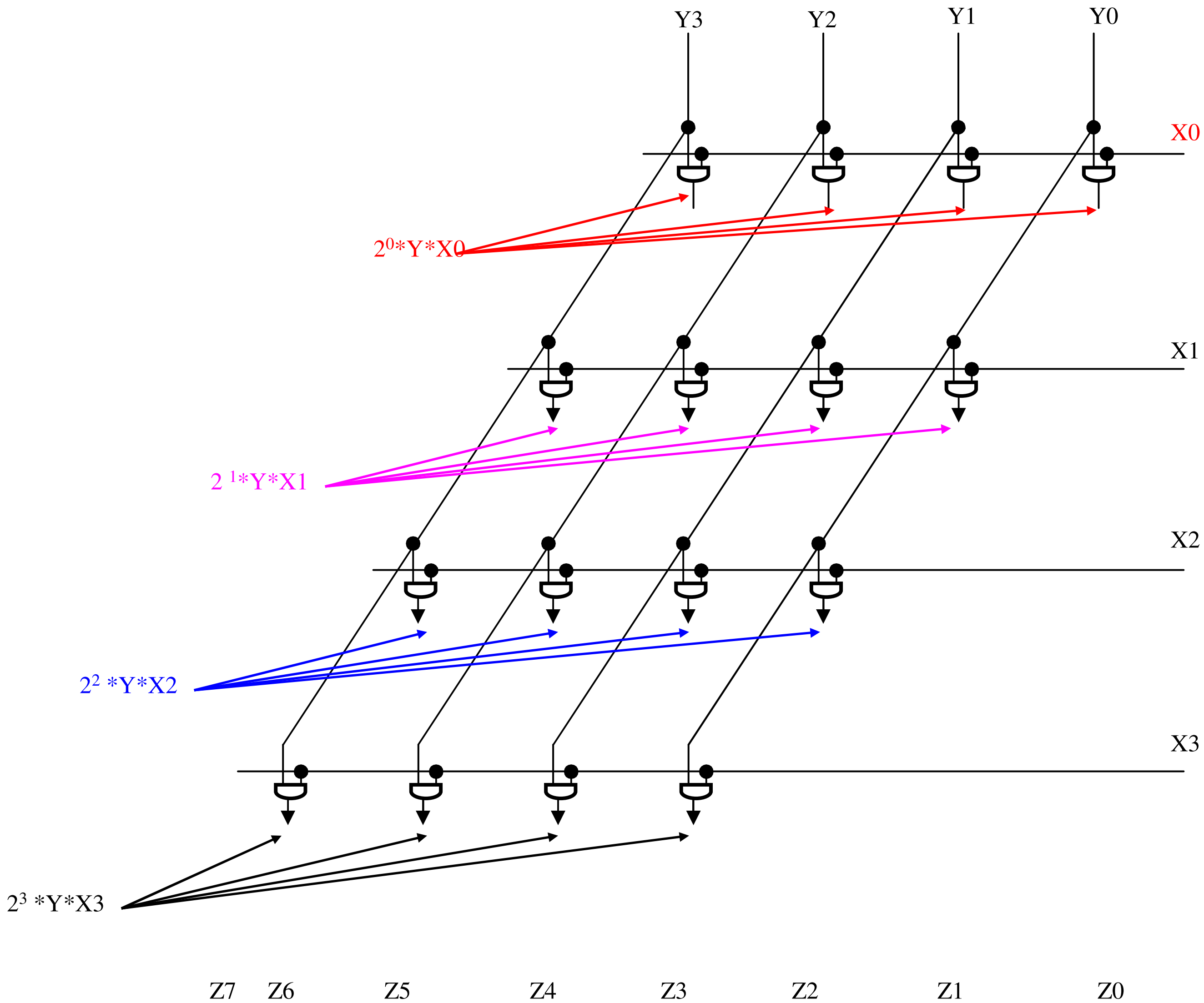
Array of AND gates



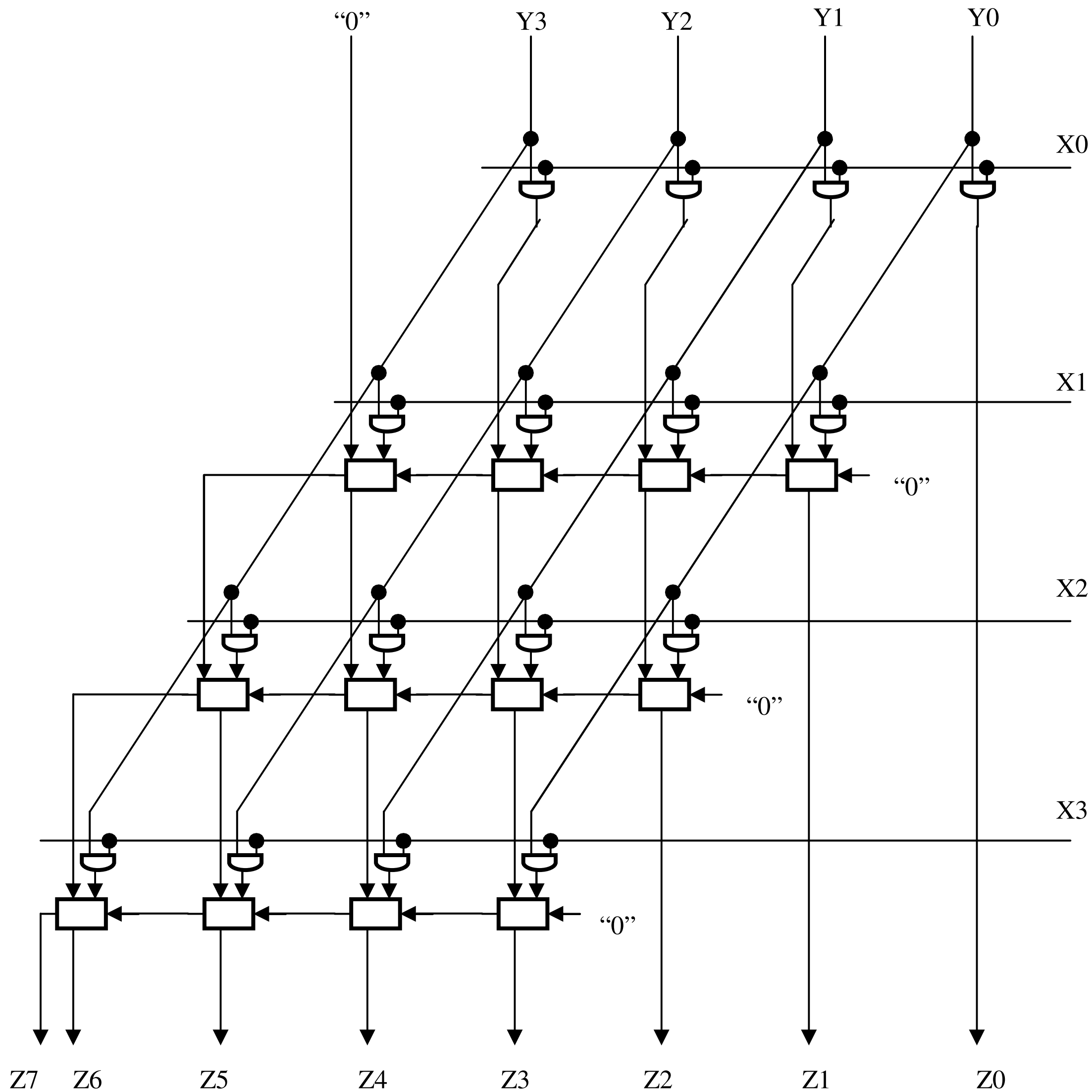
Array of AND gates



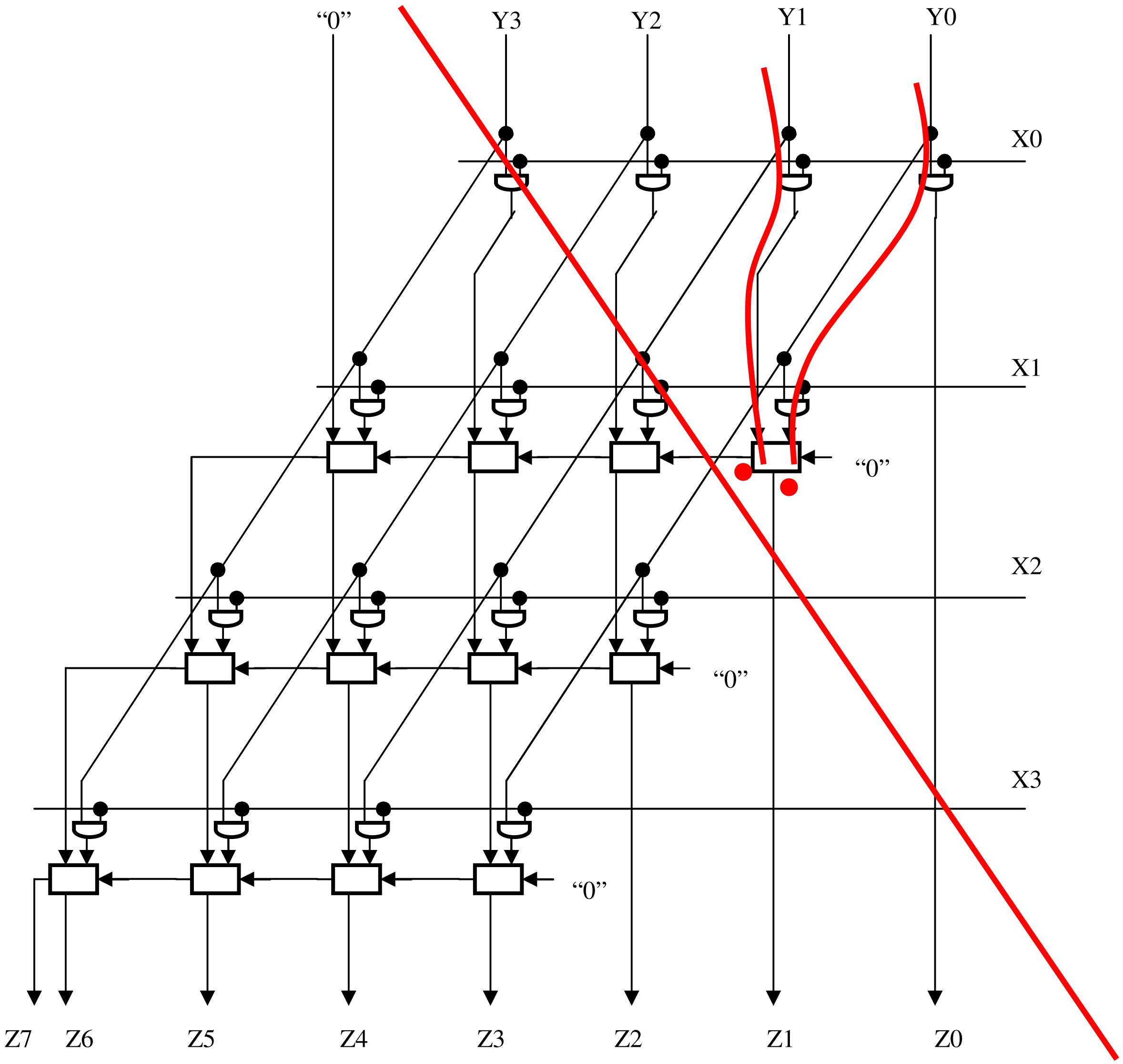
Array of AND gates



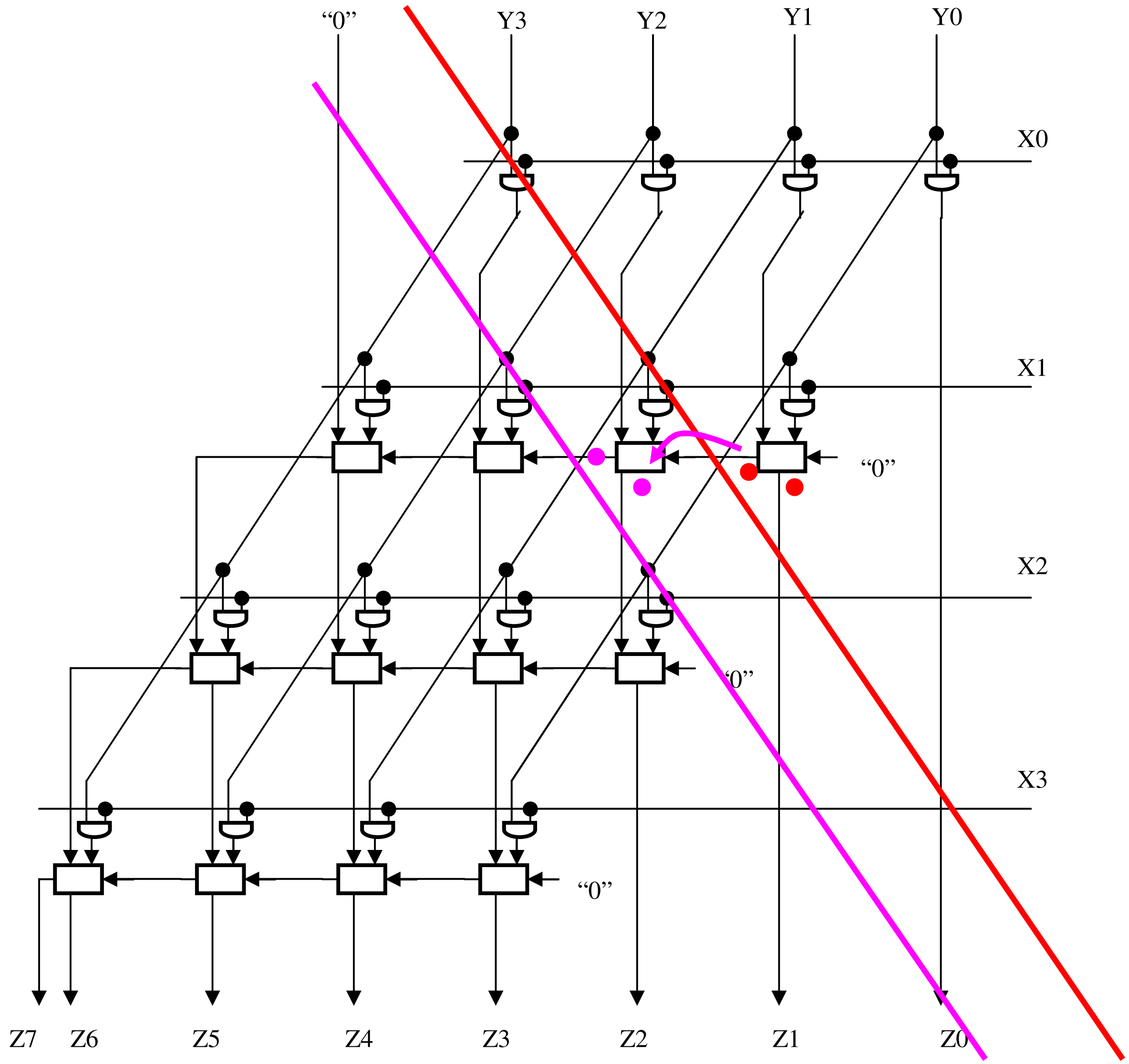
RCA with and gates



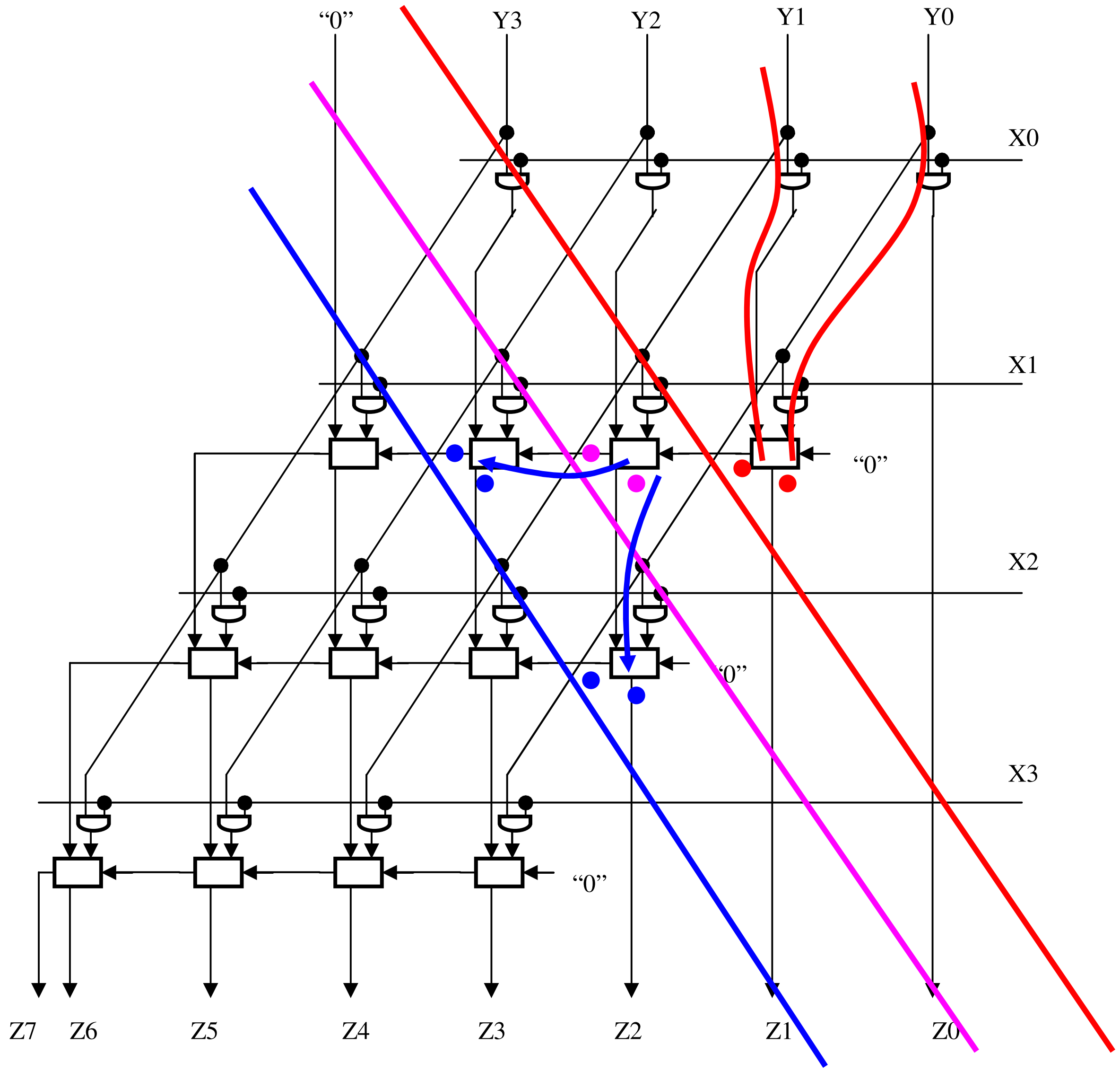
RCA with and gates



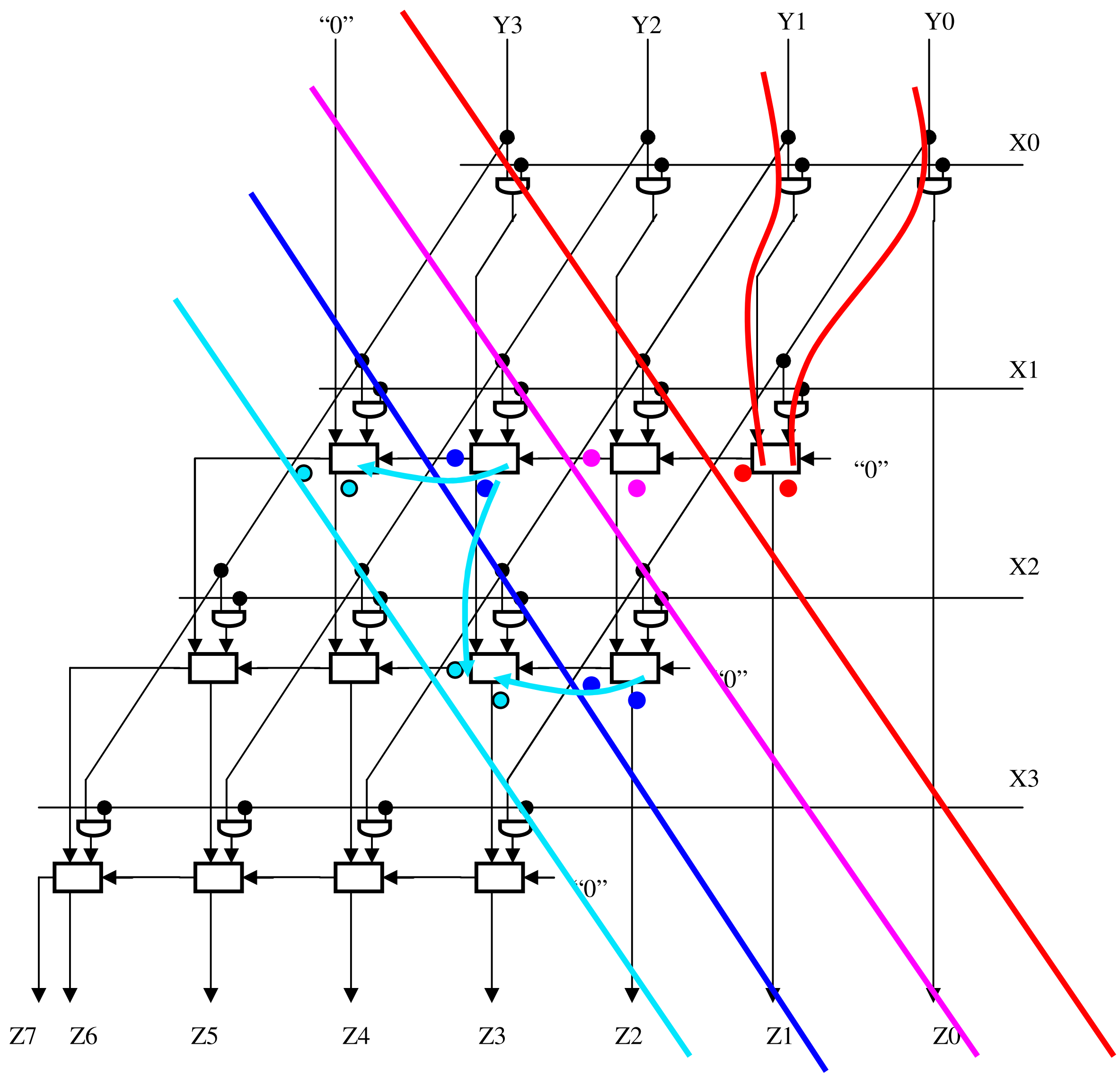
RCA with and gates



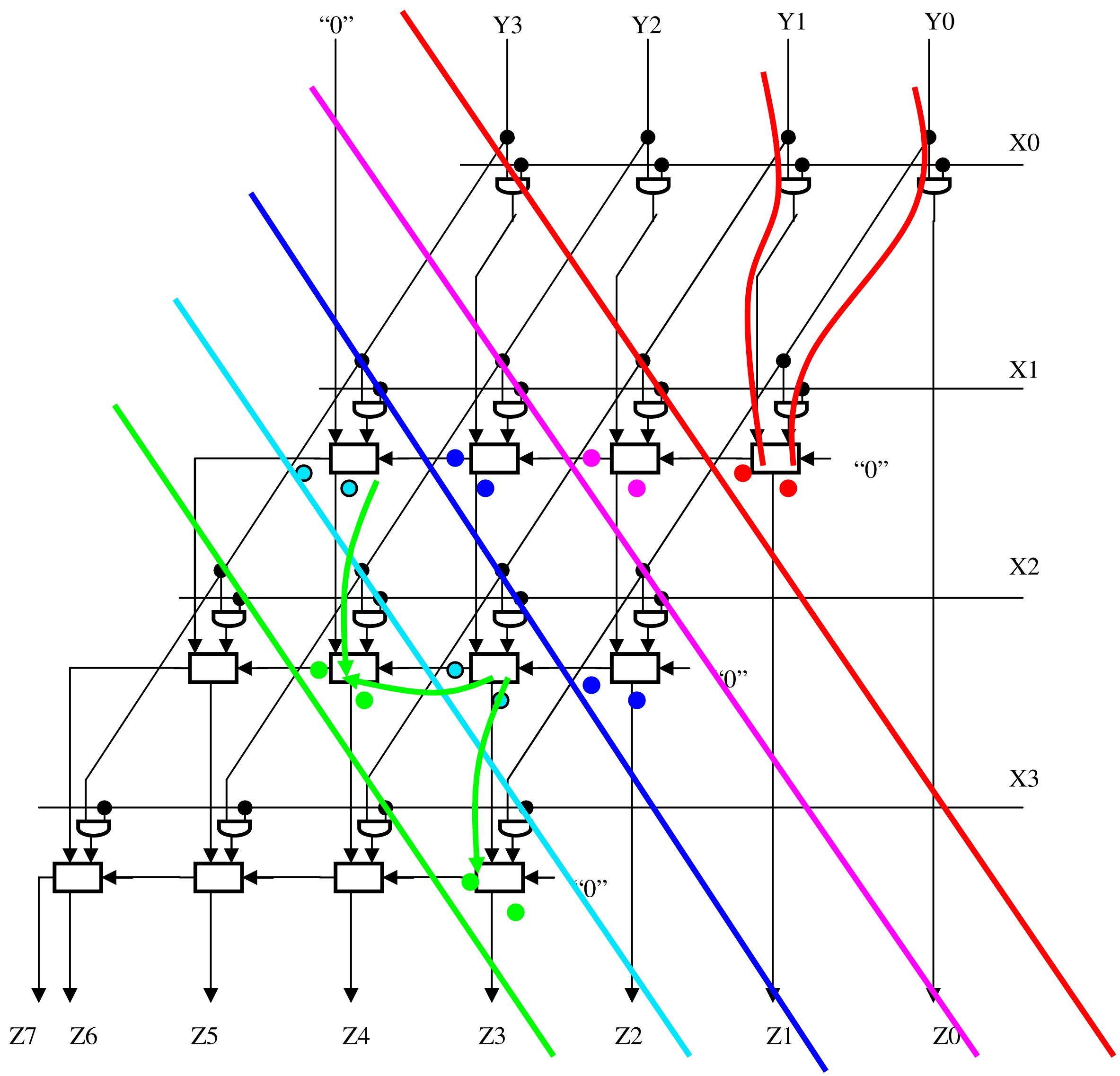
RCA with and gates



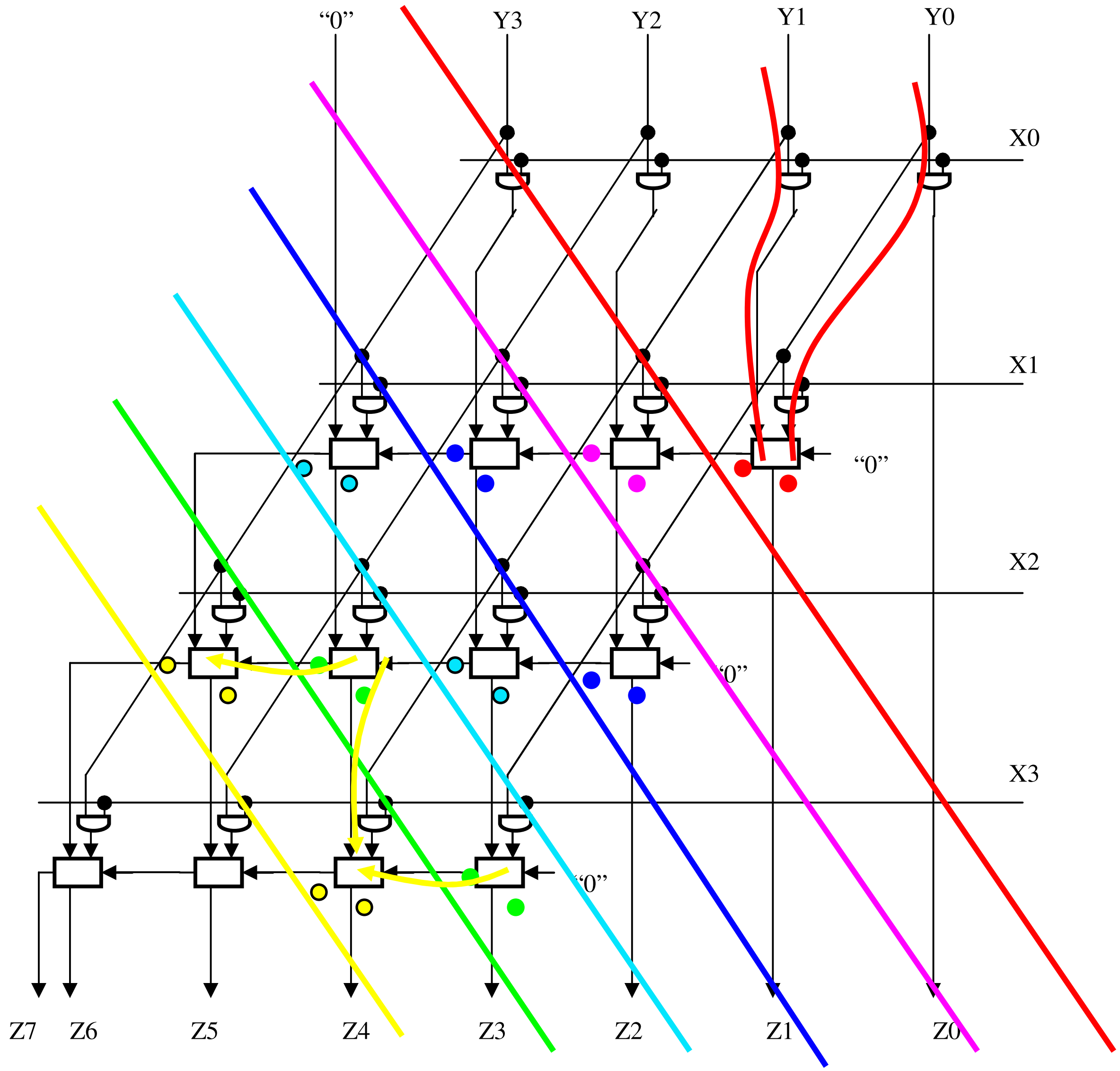
RCA with and gates



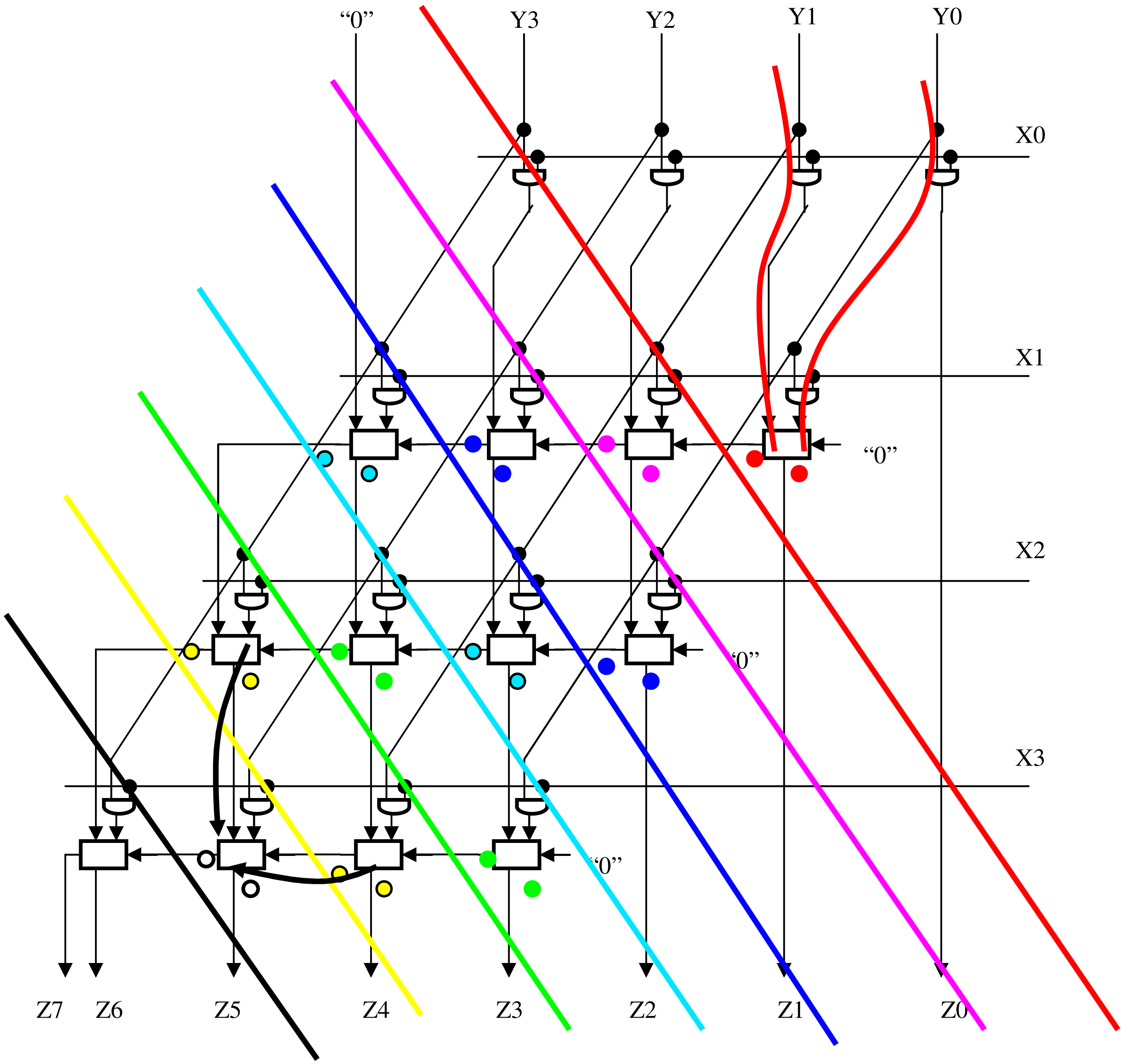
RCA with and gates



RCA with and gates



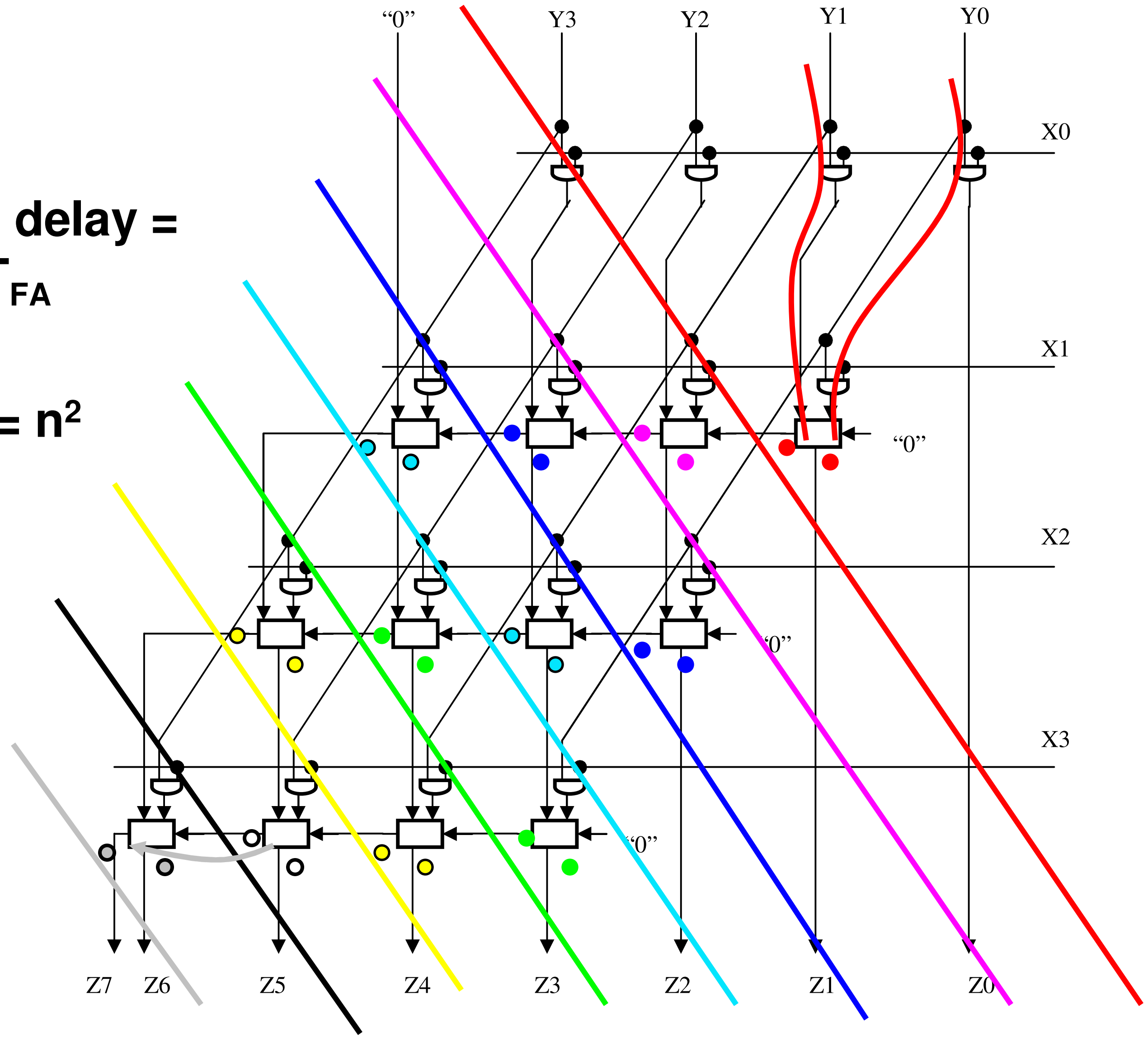
RCA with and gates



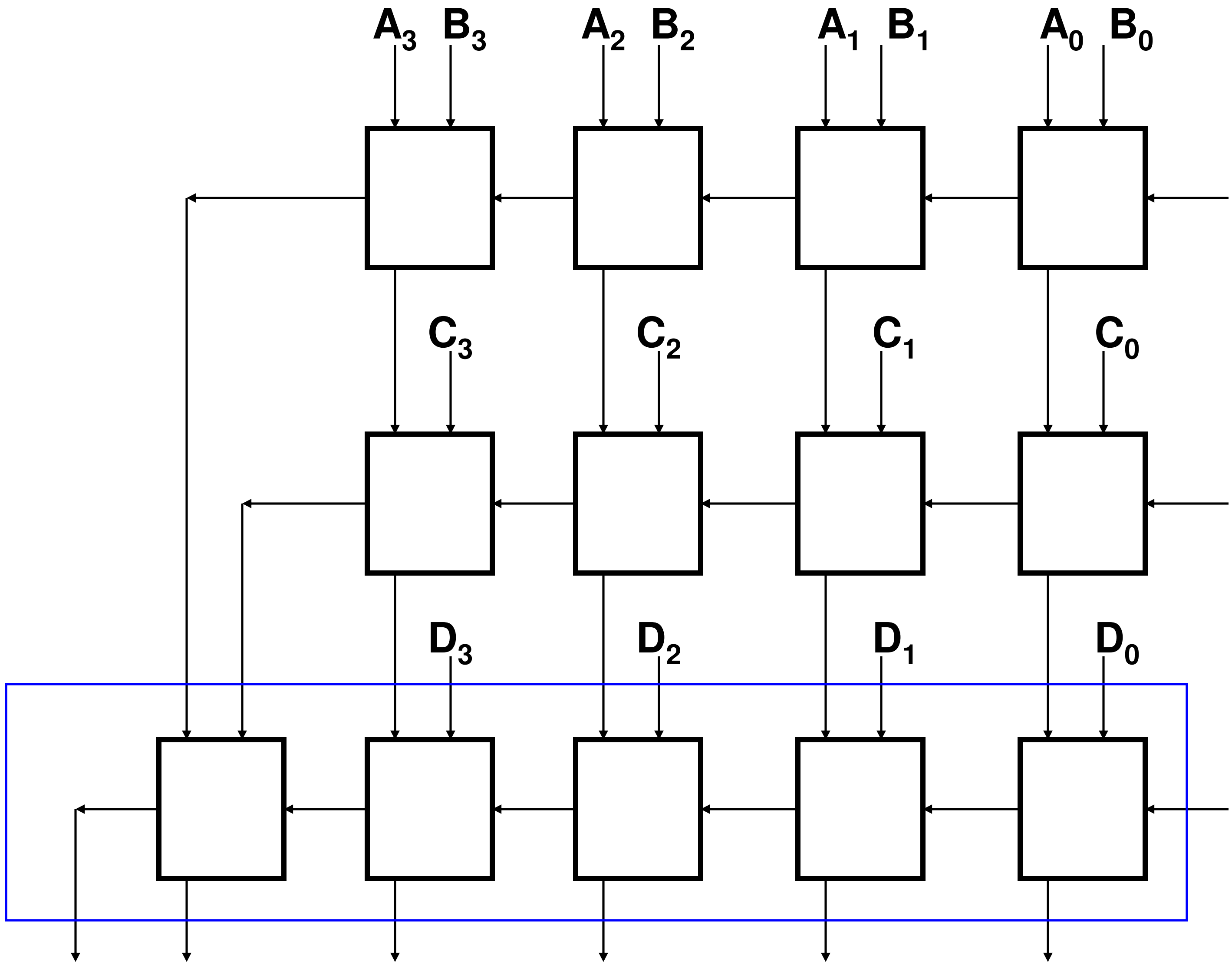
RCA with and gates

Total delay = $2*n*T_{FA}$

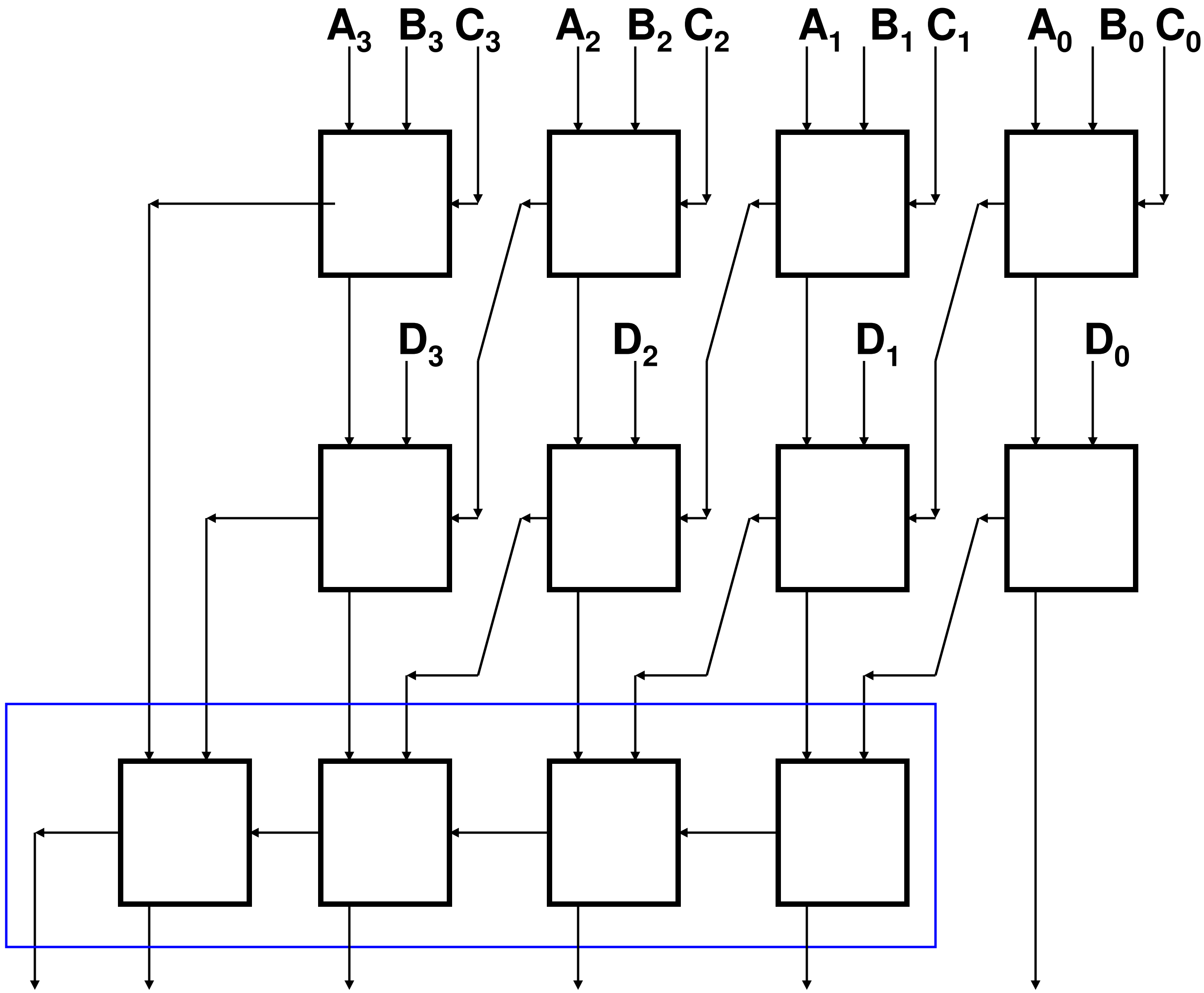
Area= n^2



Principle of CSA – add 4 numbers using a RCA



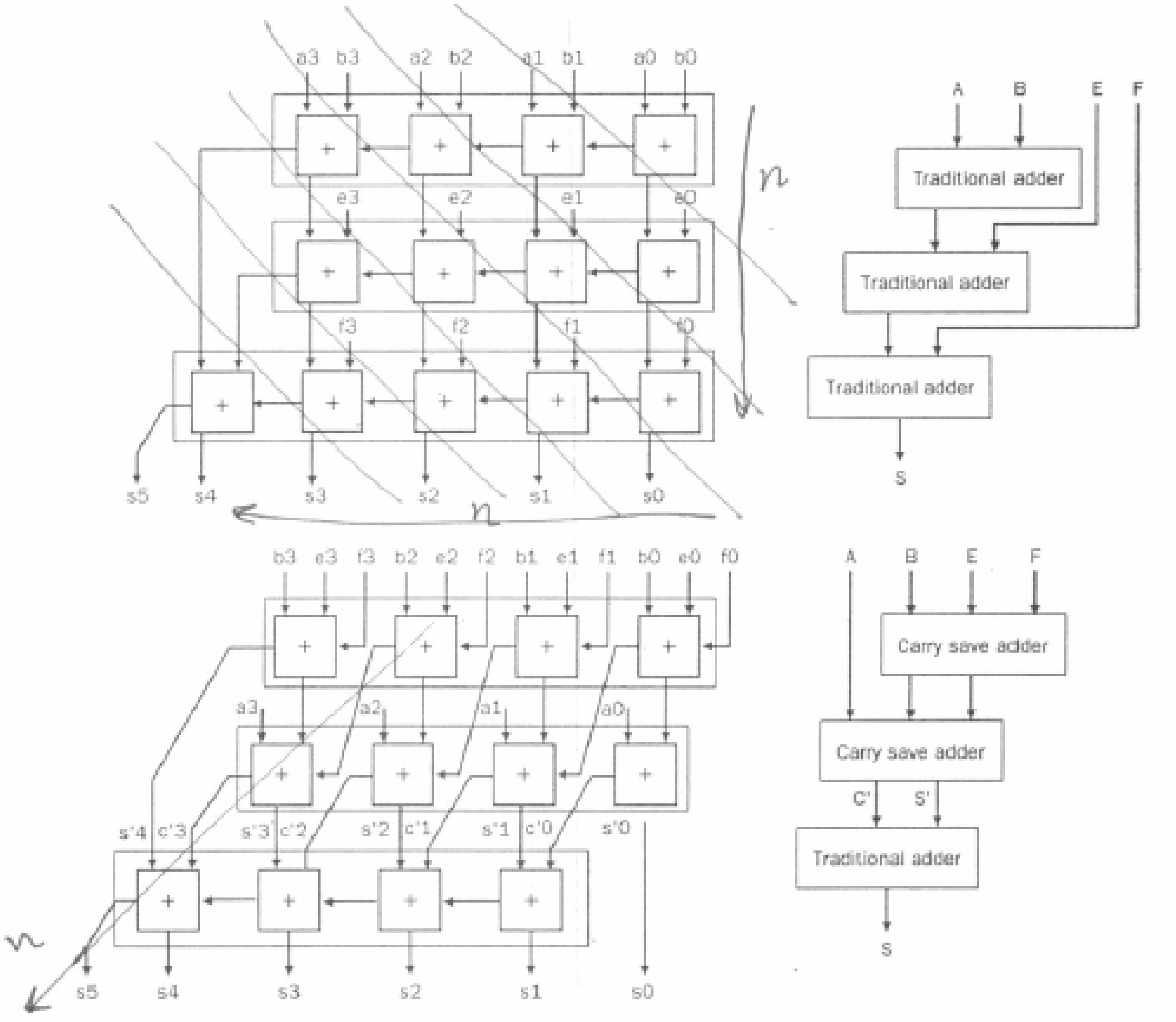
Principle of CSA – add 4 numbers using a CSA



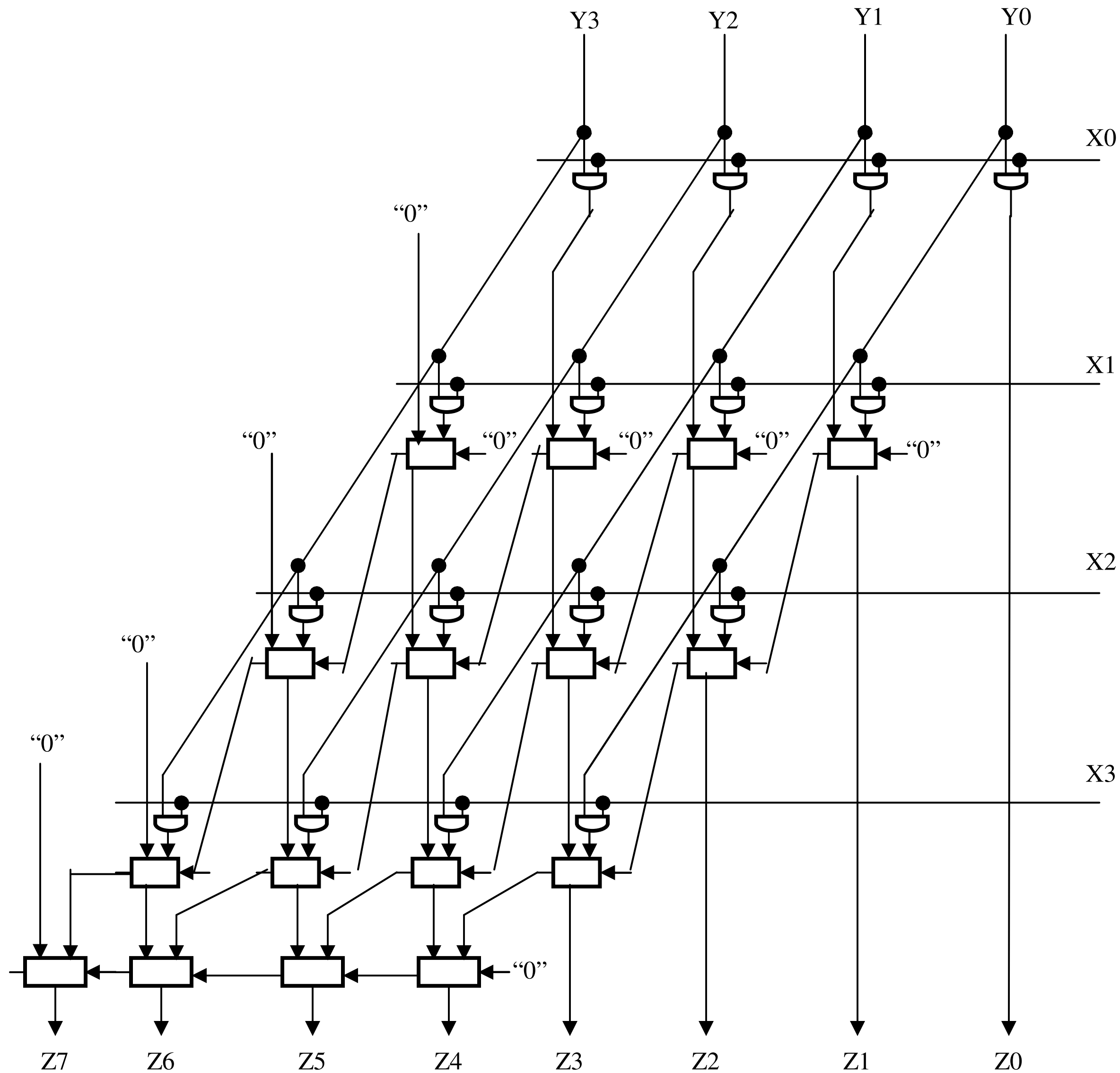
Principle of CSA

RCA delay =
$2*n*T_{FA}$

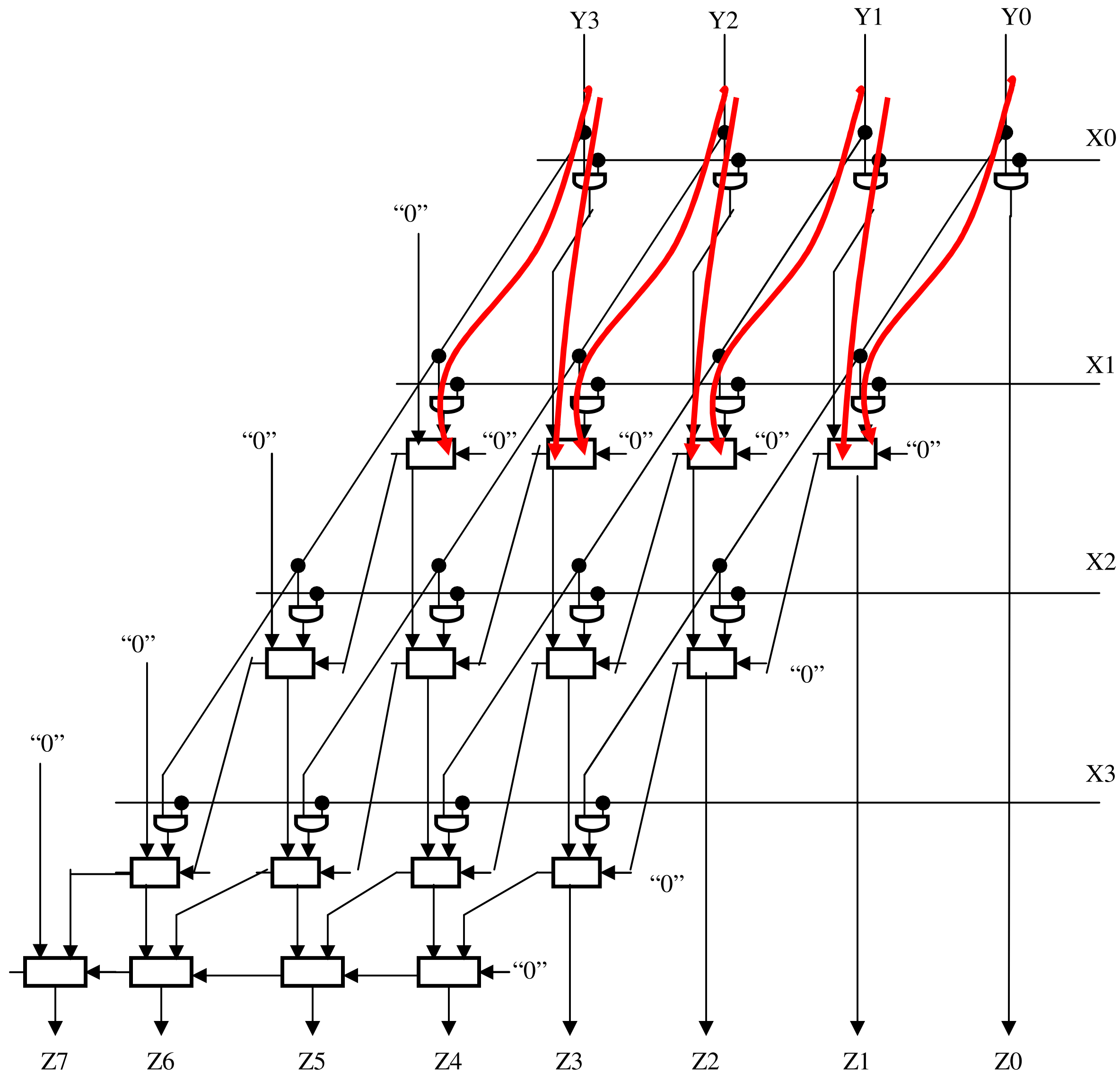
CSA delay =
$(n+lg_2 n)*T_{FA}$



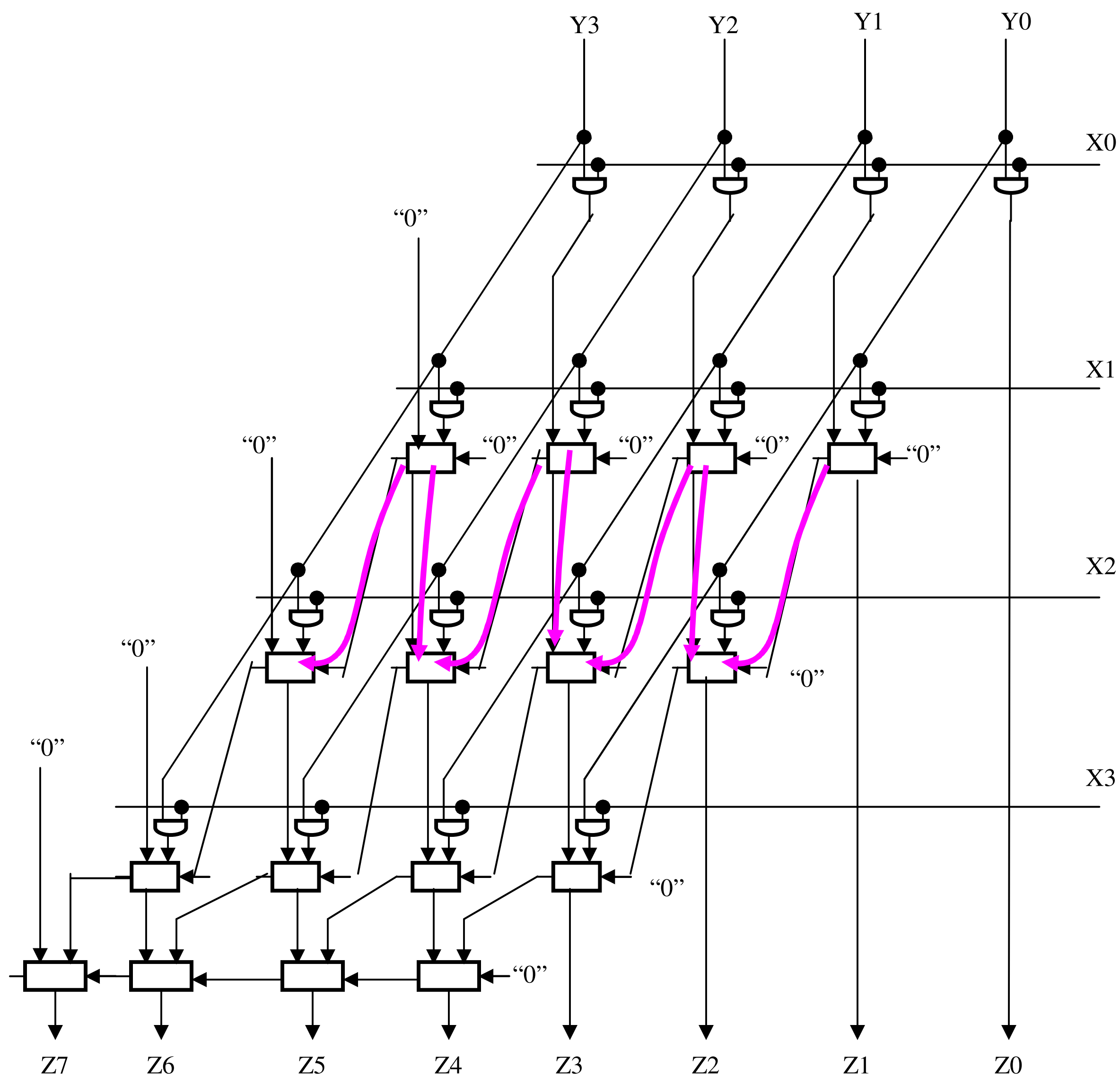
CSA based multiplier



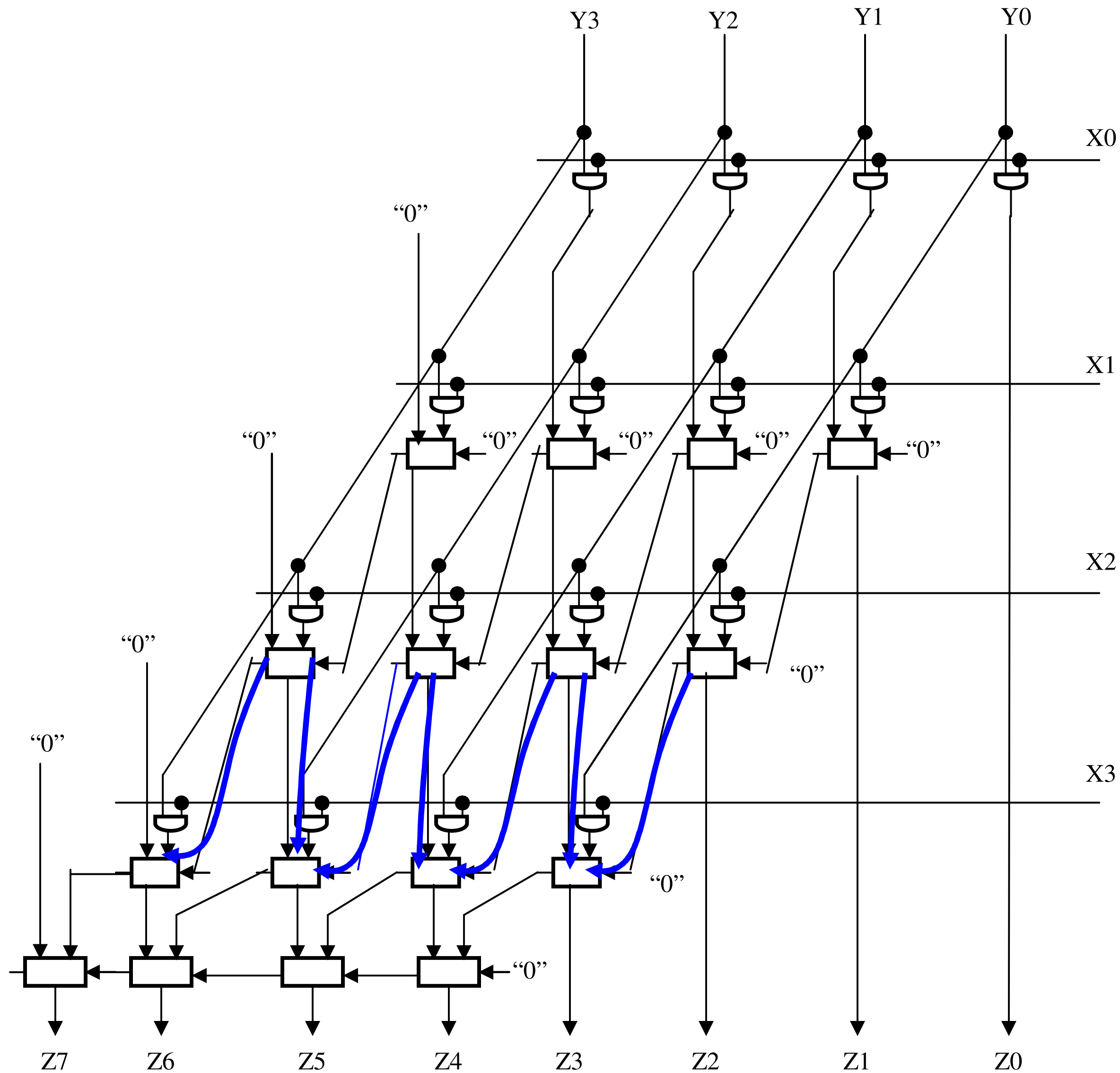
CSA based multiplier



CSA based multiplier



CSA based multiplier



CSA based multiplier

Total delay =

$(n-1)*T_{FA} + T_{RCA} =$

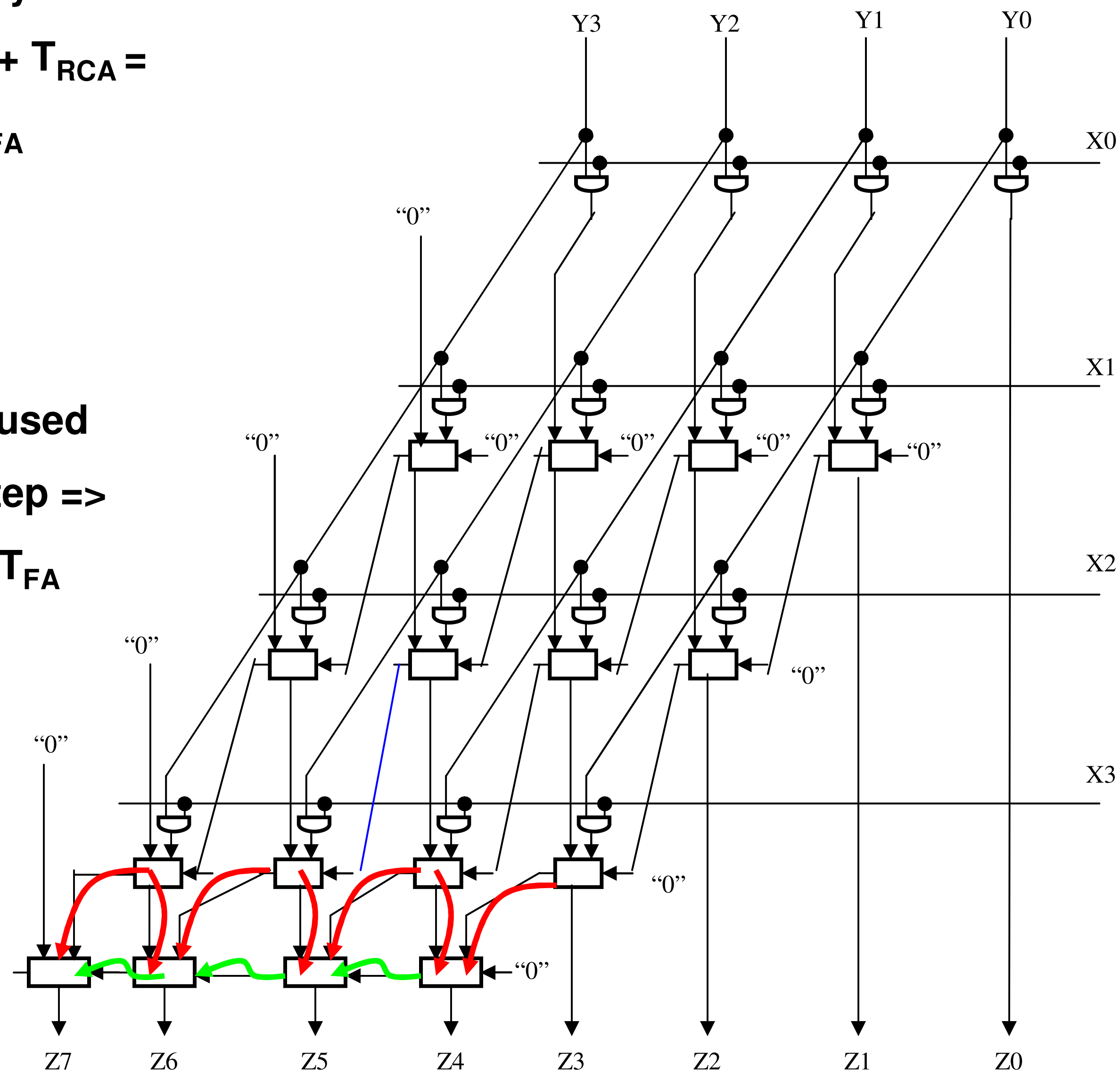
$(2*n-1)*T_{FA}$

Area= n^2

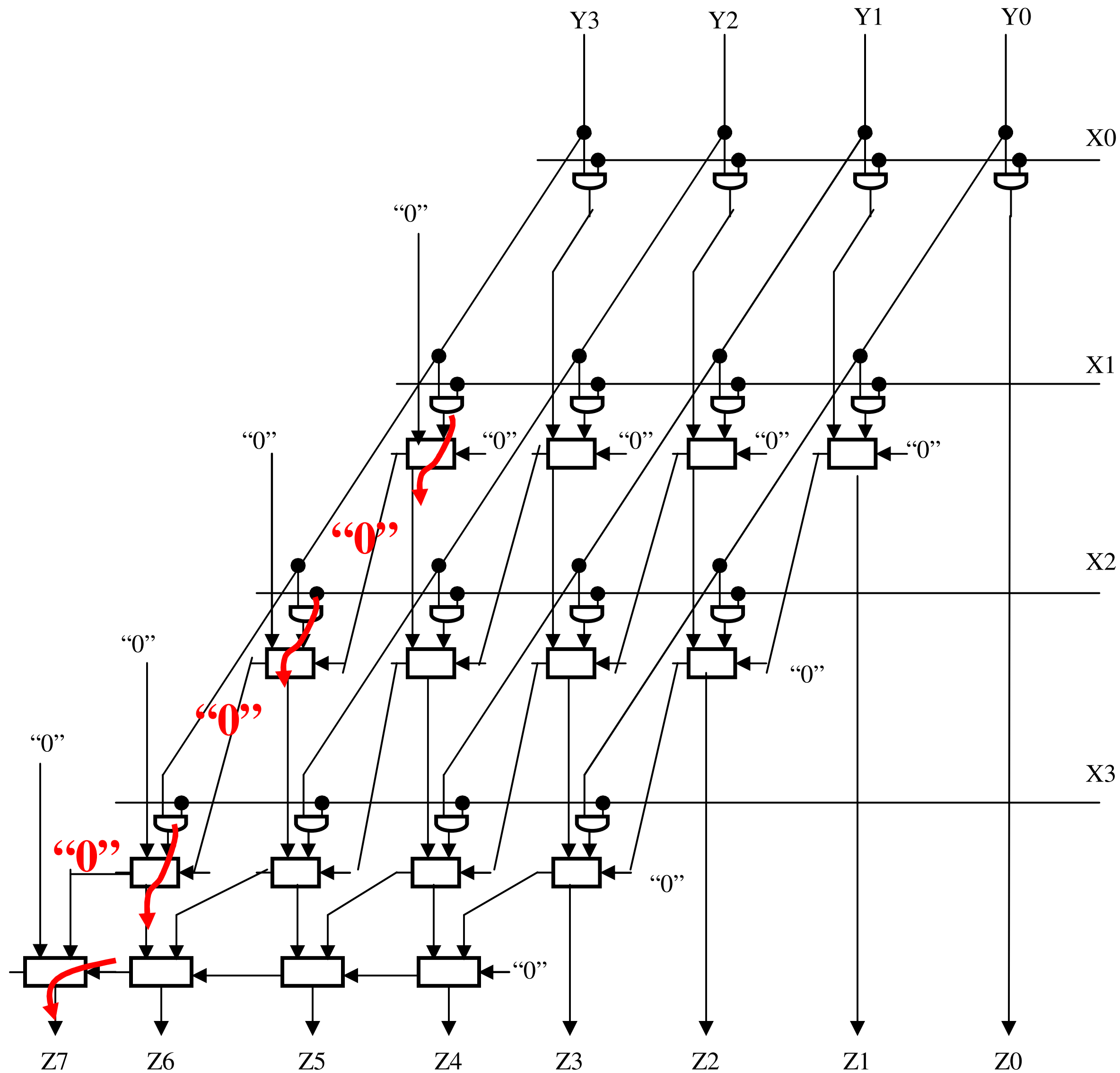
If CLA is used

for last step =>

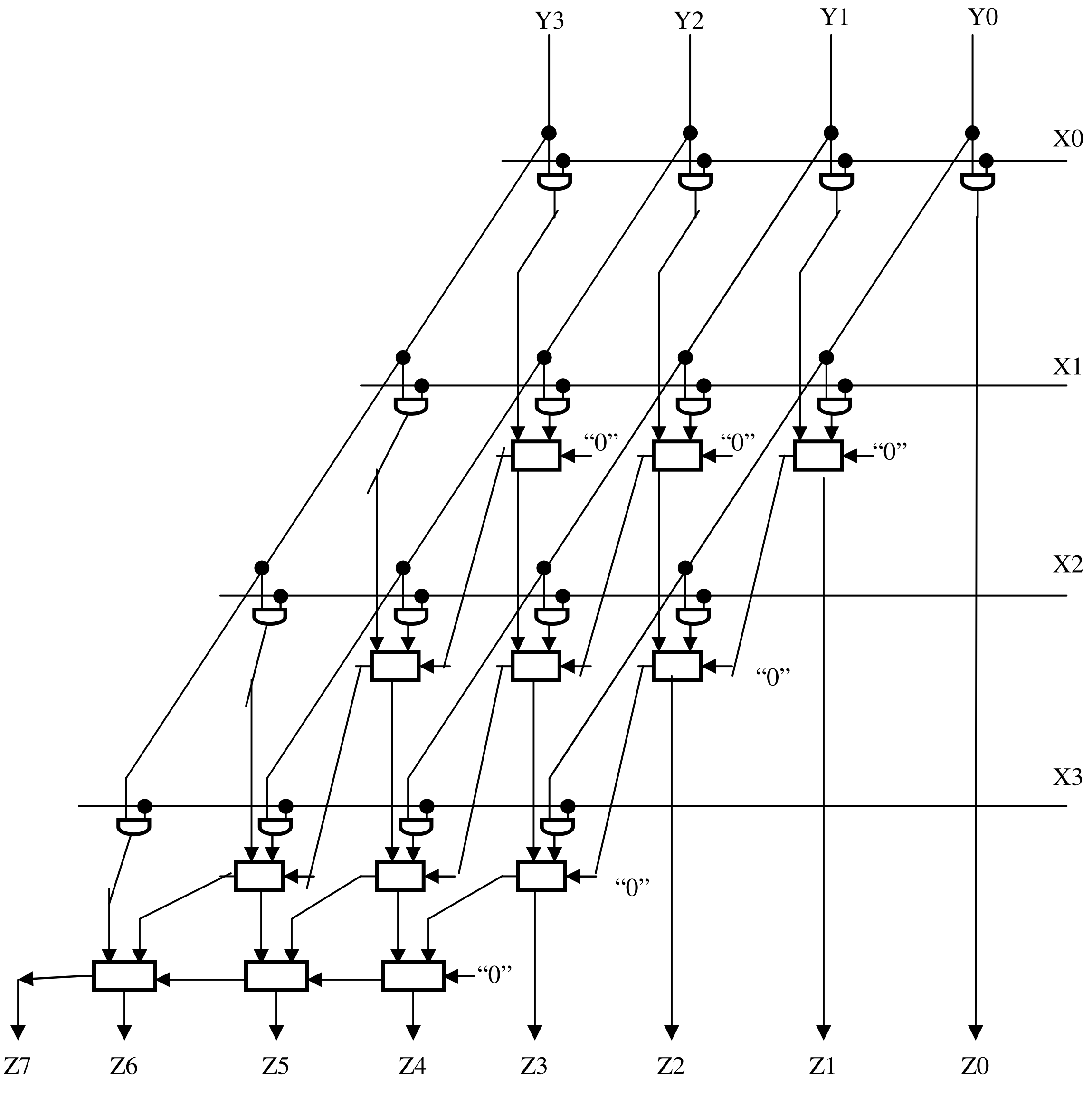
$(n+lg_2n)*T_{FA}$



CSA based multiplier



CSA based multiplier updated



Homework: O($(\log_2 n)^2$) using CLA

Build a multiplier that adds $Y*X_0 + 2*Y*X_1$ and $Y*X_2 + 2*Y*X_3$ and $Y*X_4 + 2*Y*X_5$ etc., using (n/2) CLAs with (n+1) bits each.

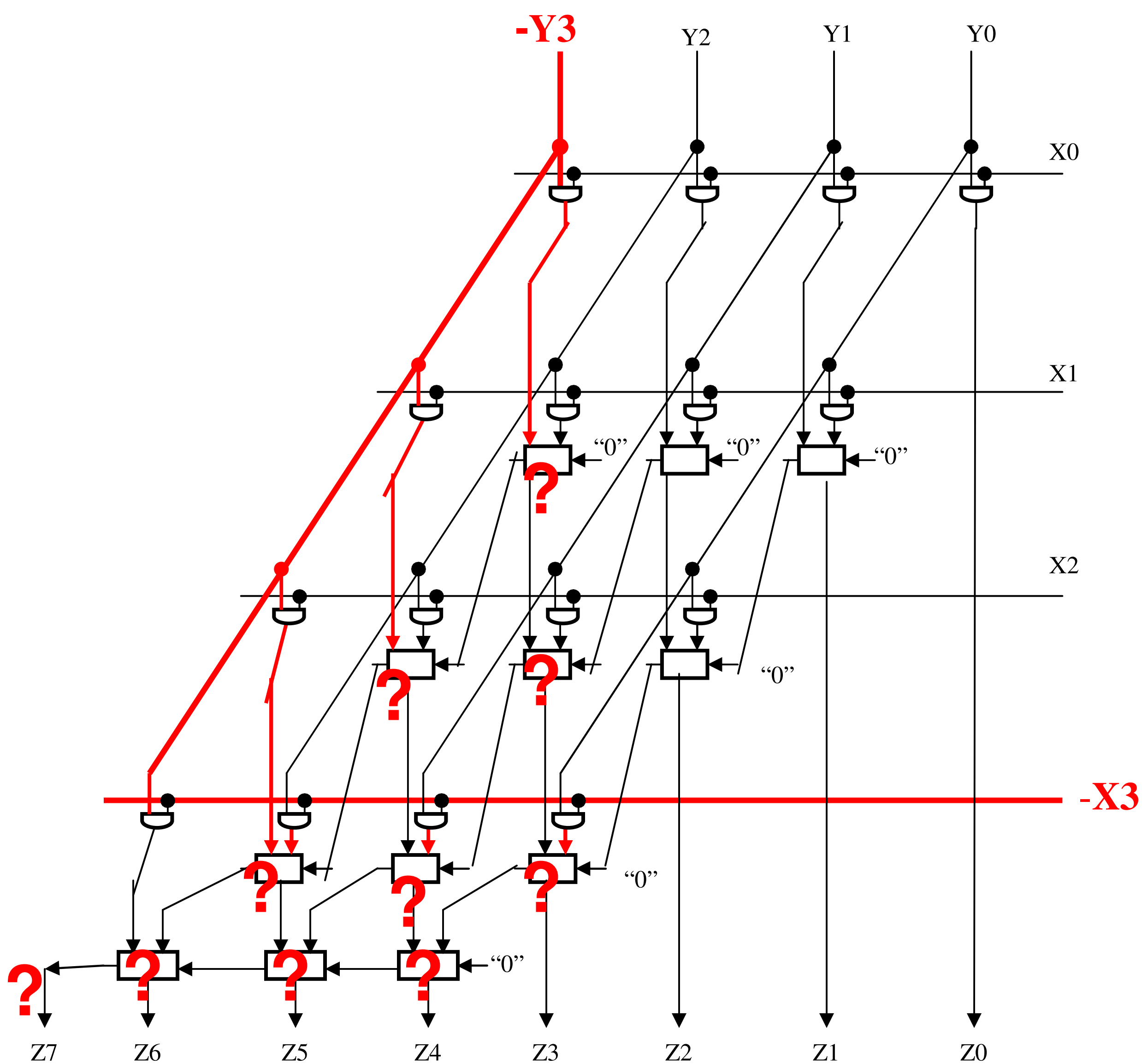
Then, add the (n/2) results using (n/4) CLAs with (n+3) bits each.

Continue till you need to add only 2 numbers.

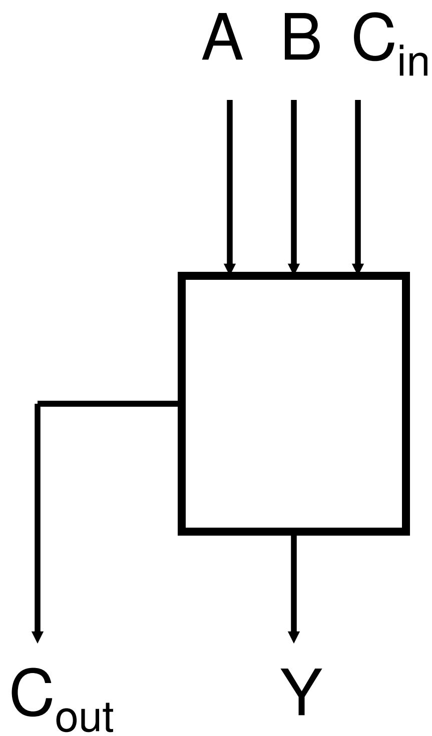
Calculate the delay expected in a CLA has a delay of $T*\log_2 n$

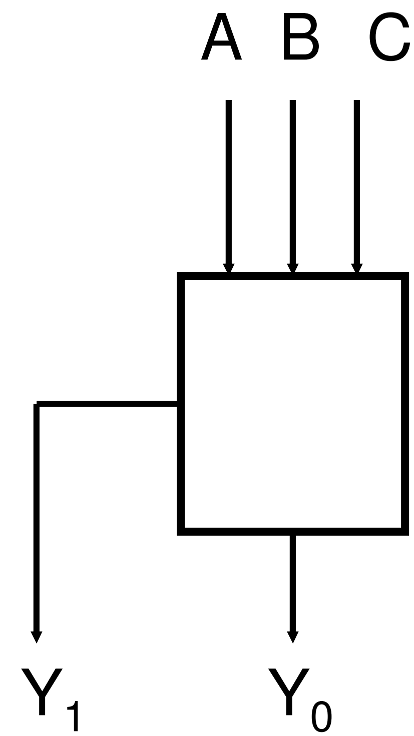
Signed array multiplier

CSA based signed multiplier



A full adder – a reminder

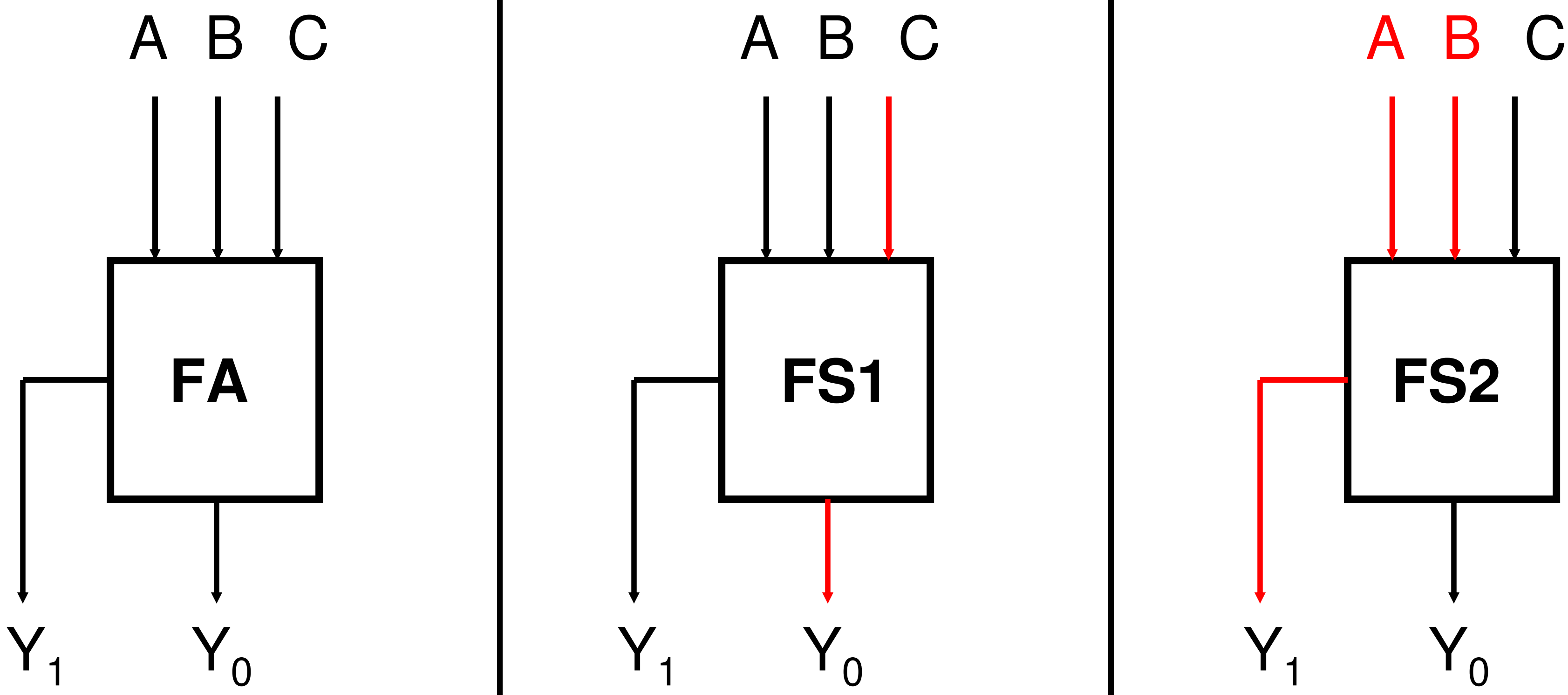




$2*C_{out}+Y = A + B + C_{in}$

$2*Y_1+Y_0 = A + B + C$

A full adder – a reminder

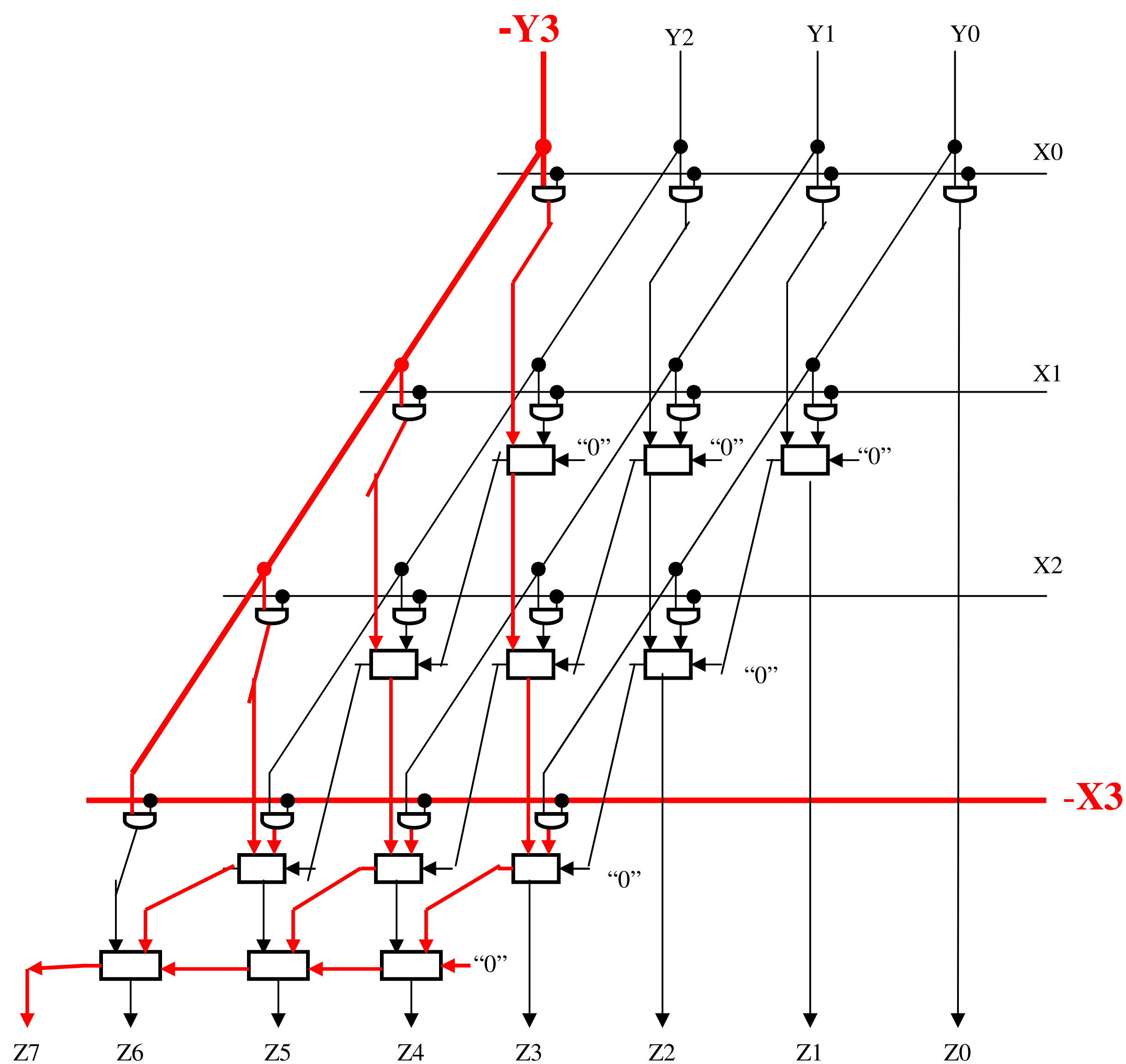


$2*Y_1+Y_0 = A+B+C$

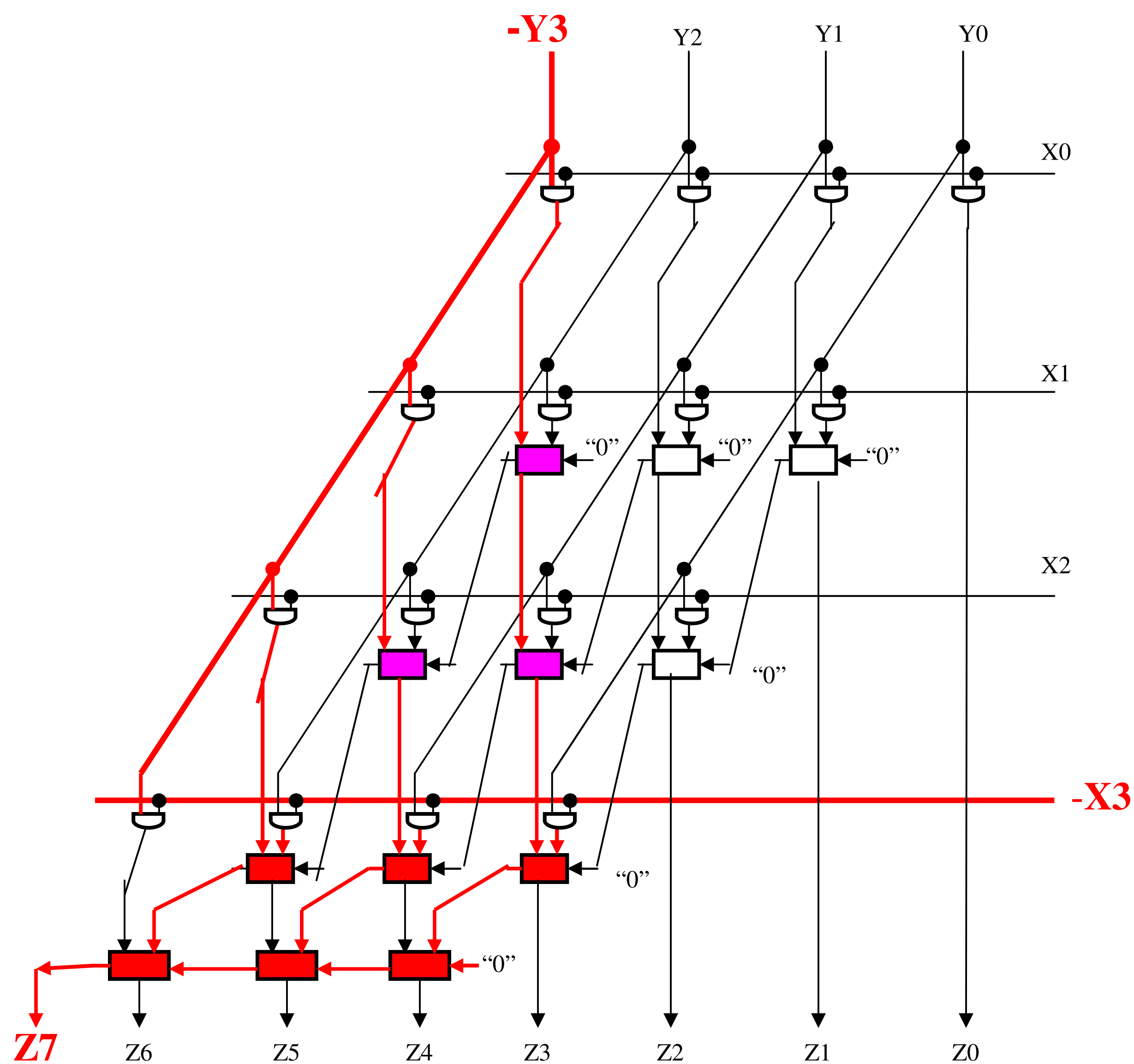
$2*Y_1-Y_0 = A+B-C$

$-2*Y_1+Y_0 = -A-B+C$

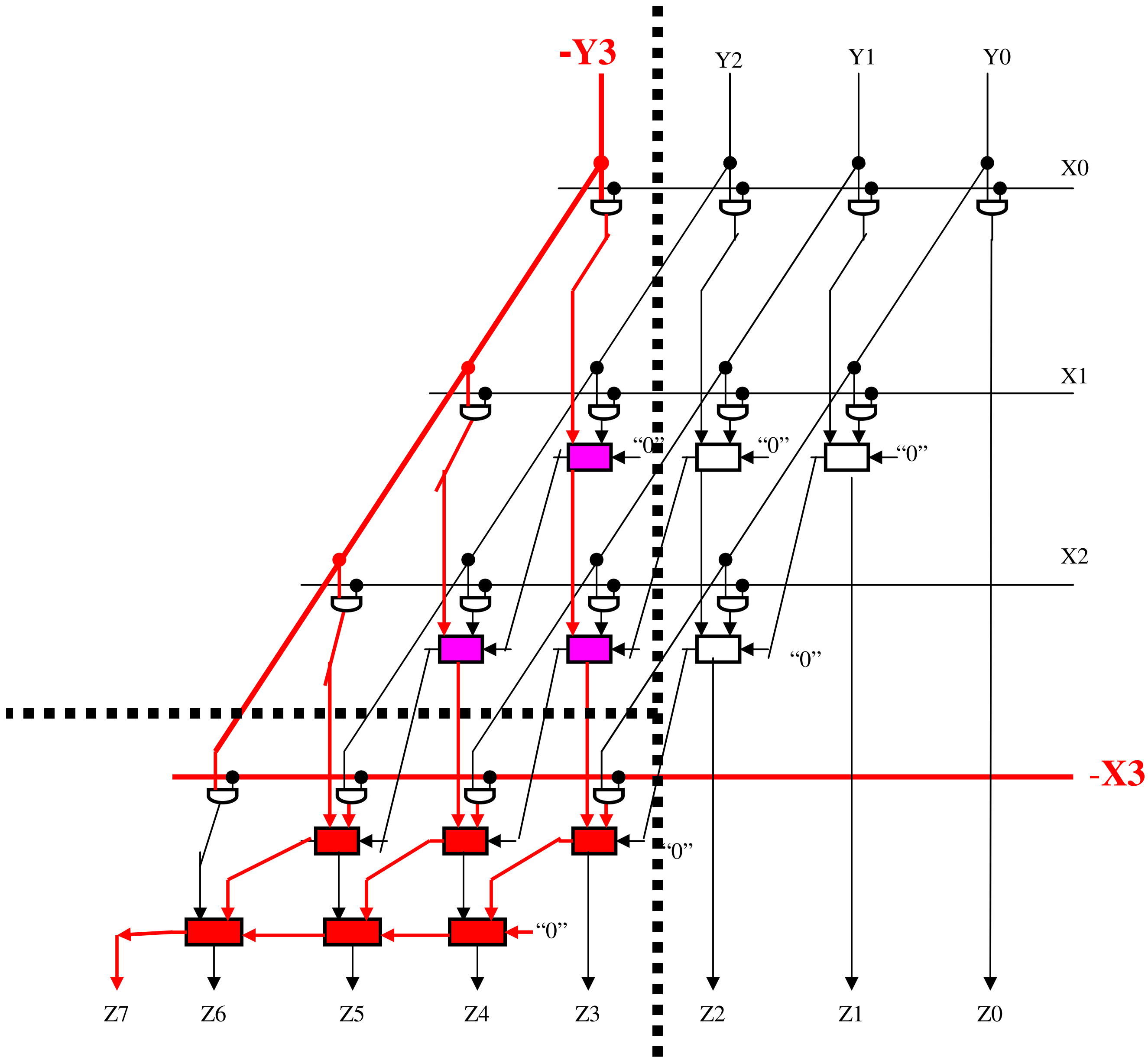
CSA based signed multiplier



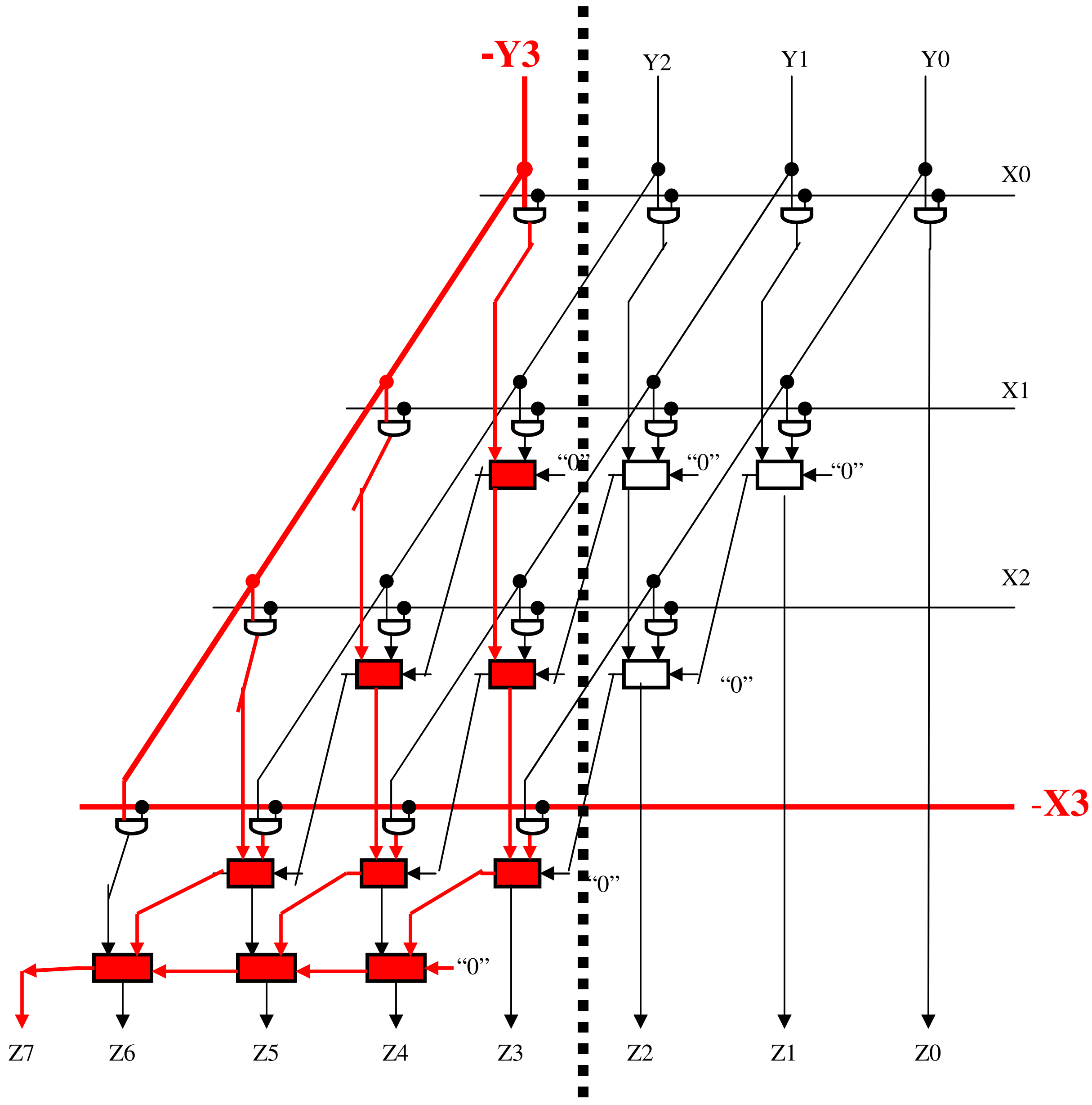
CSA based signed multiplier



CSA based signed multiplier



CSA based signed multiplier



other array multipliers

other array multipliers

1. **Odd/Even CSA multiplier** **$O(n/2 + \log_2 n)$**
2. **CLA binary tree multiplier** **$O([\log_2 n]^2)$**
3. **Wallace tree multiplier** **$O(\log_{1.5} n)$**
4. **CSA binary tree** **$O(\log_2 n)$**

And there are others!

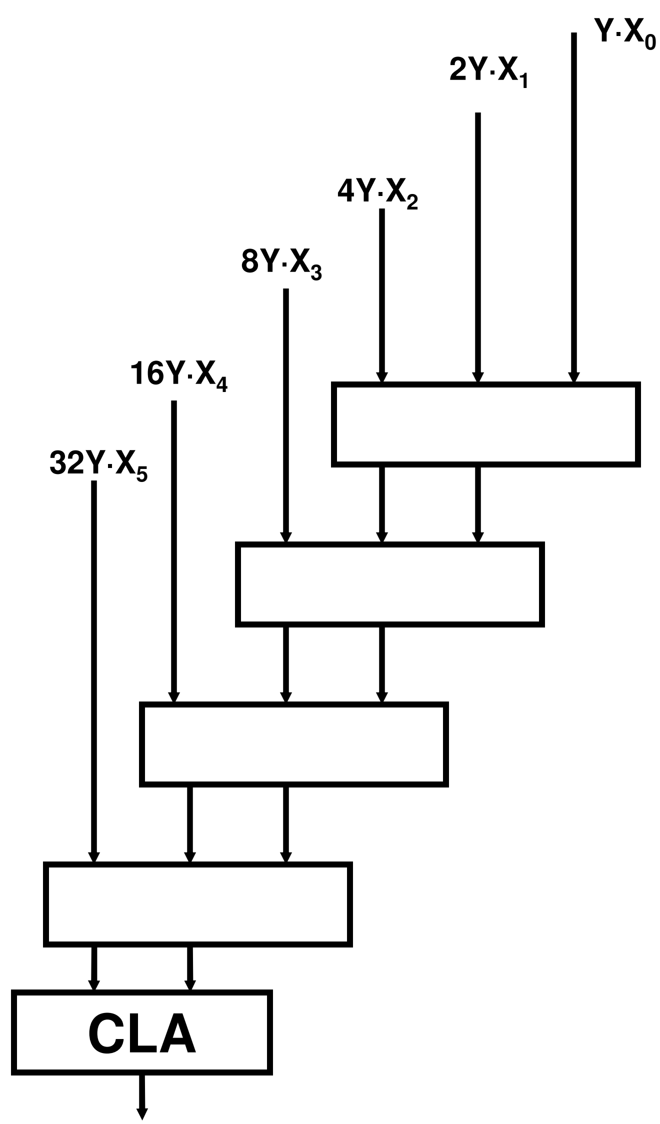
Note: in all of the above except the Booth, we demonstrate unsigned multiplication

1. Odd/Even CSA multiplier

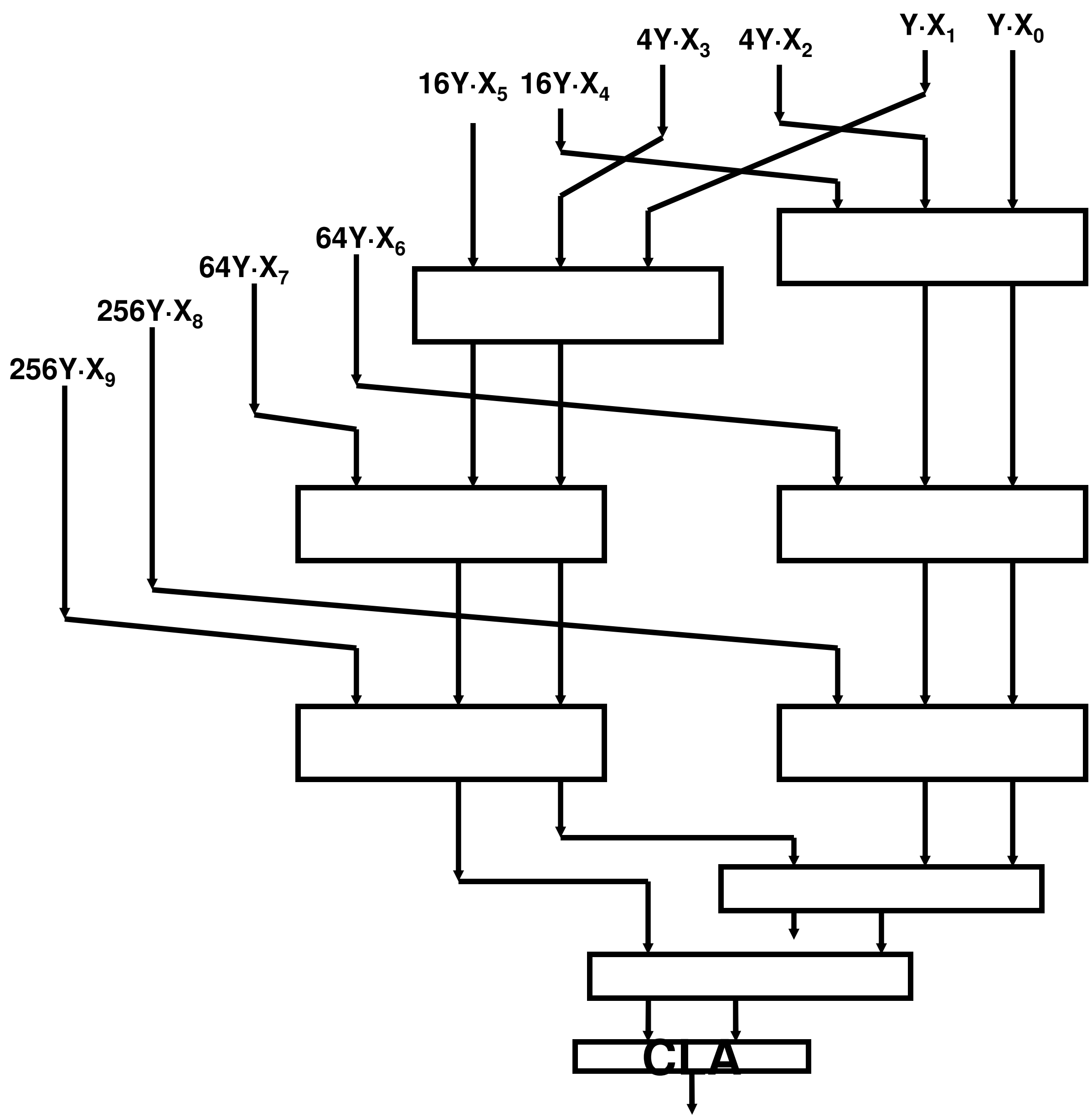
1. **Similar to the ”regular” CSA multiplier except that we add the “even” summands and the “odd” summands separately (and sum them together at the end)**

2. **This is done in parallel, so we get about half of the time**

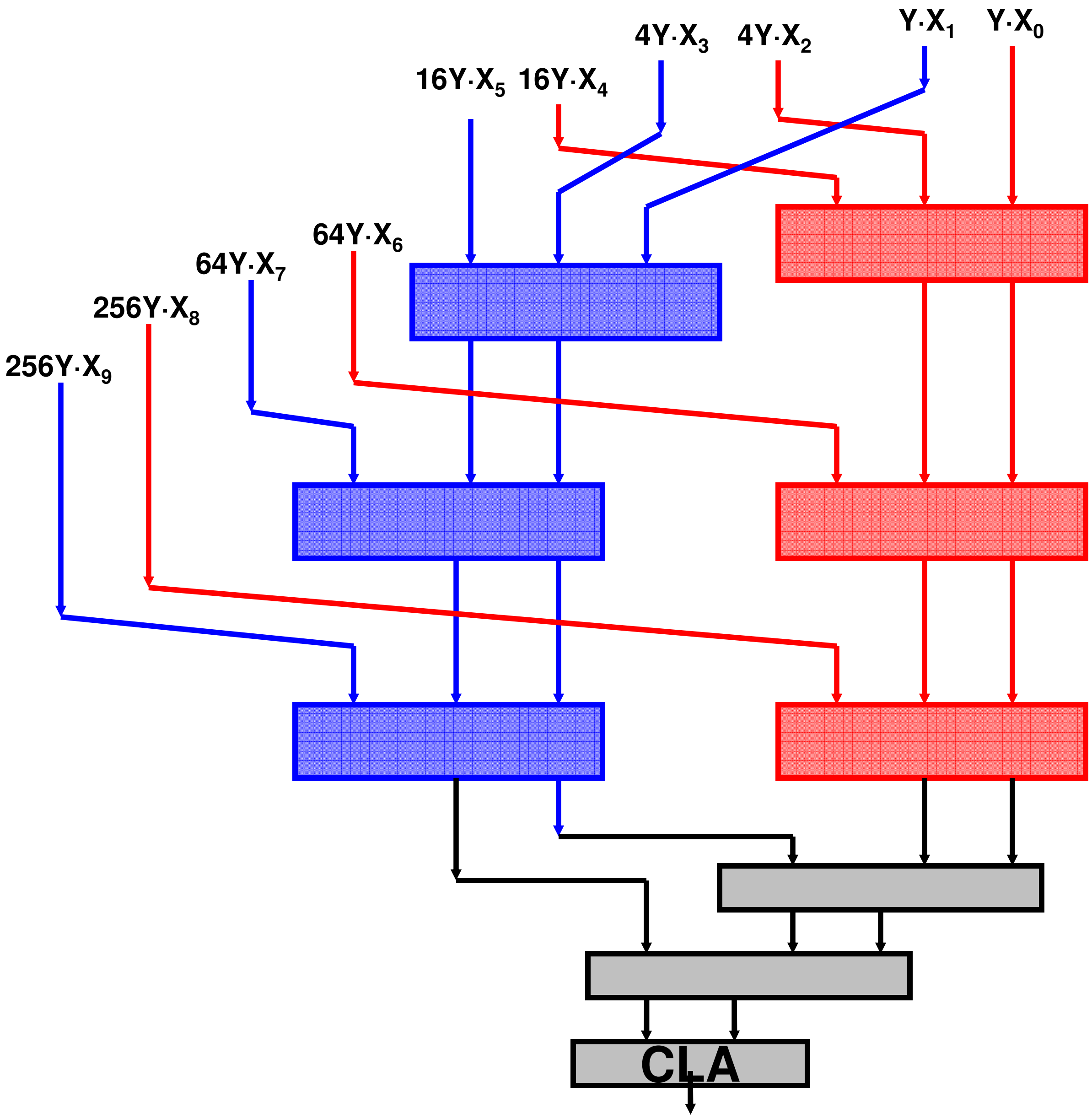
A regular CSA multiplier



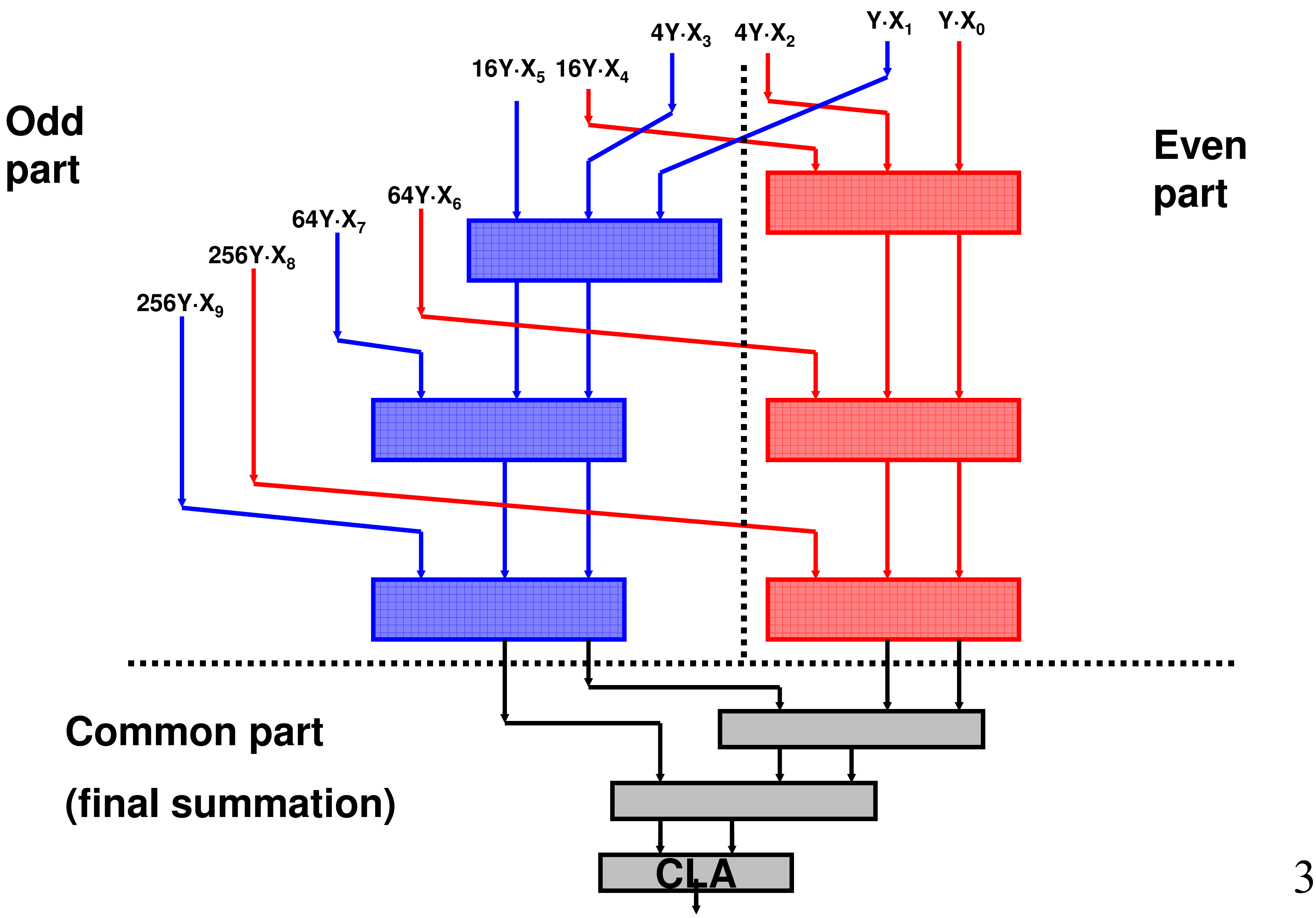
Odd/Even CSA multiplier



Odd/Even CSA multiplier – The parallel flow



Odd/Even CSA multiplier – The parallel flow



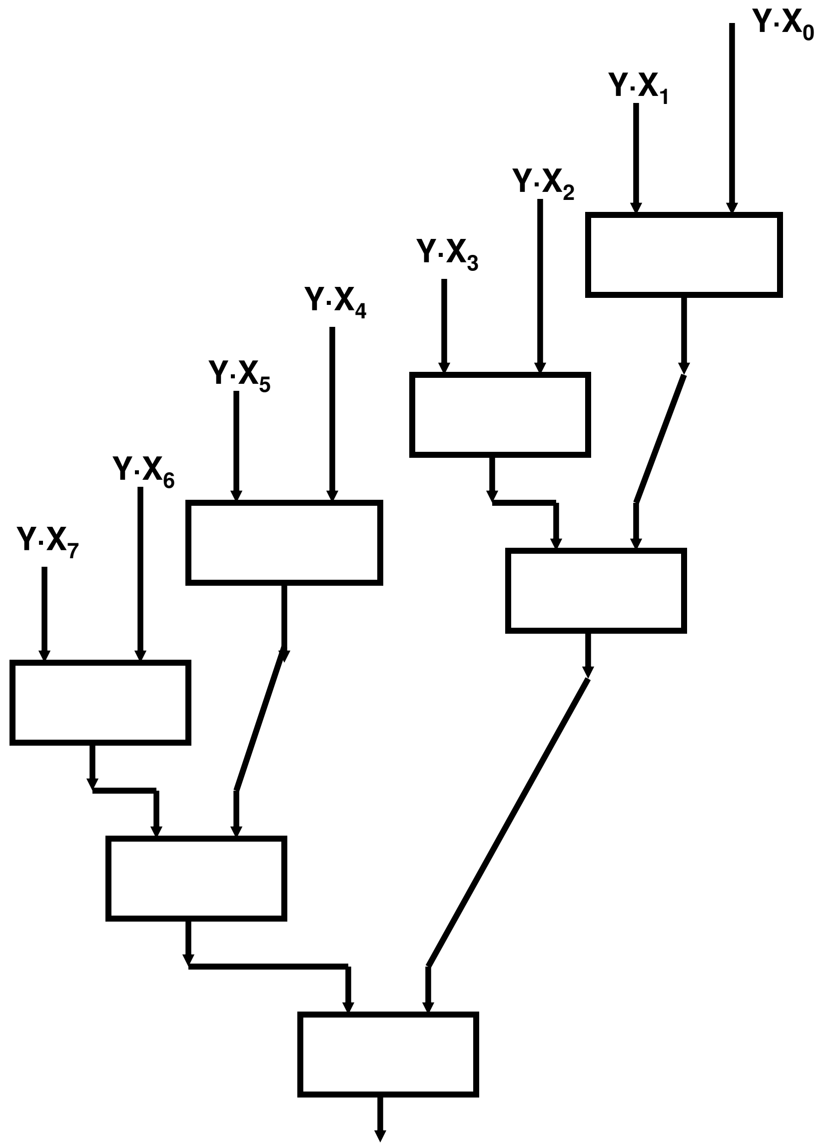
2. Binary tree with CLAs (See homework)

1. O($[\log_2 n]^2$)
2. Use (n/2) CLAs for adding $2Y_1+Y_0$, $2Y_3+Y_1$, $2Y_5+Y_4$, etc.
3. Use (n/4) wider CLAs for adding the (n/2) results
4. Continue in a tree like structure in the same manner
5. Delay is $T_{CLA} * \log_2 n$ since the tree depth is $\log_2 n$
6. Since $T_{CLA} = \log_2 n$ we get O($[\log_2 n]^2$)
7. Cost is $C_{CLA} * n * (1/2+1/4+\ldots+1/2^{n-1}) = n*C_{CLA}$ (with average of 1.5n bits??) CHECK!!!

Binary tree with CLAs

All adders are CLAs

Size of adders is increasing



Binary tree with CLAs

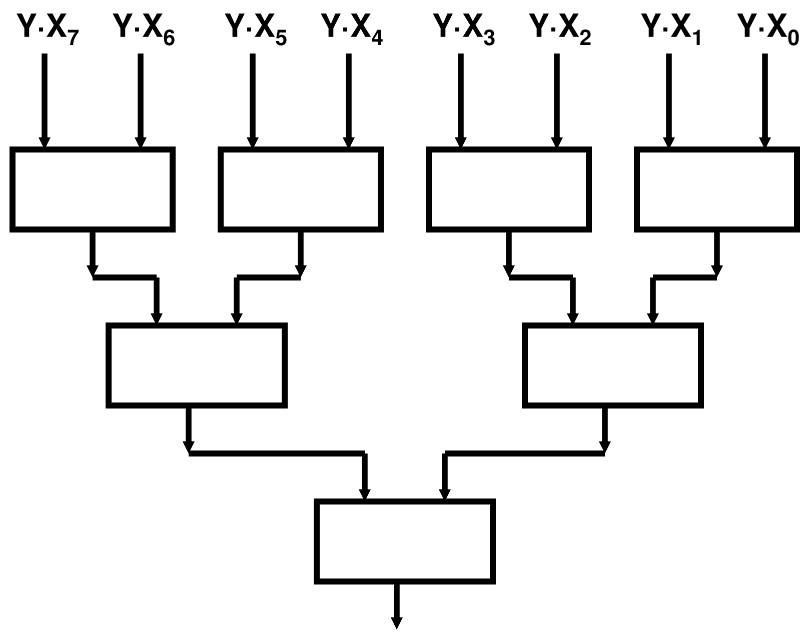
All adders
are CLAs

Delay is
T_{CLA}*$\log_2 n$

= T_{FA}*$(\log_2 n)^2$

Cost <

$(n-1)$*C_{CLA}*2



3. Wallace tree = 3>2 reduction

(See Guy Even' lectures, Patterson & Hennessy Quantitative approach Fig. A.30, I. Koren, pp 88-89)

1. Delay is $O(\log_{1.5}n)$
2. Use (n/3) CSAs for reducing the number of summands to (2/3)n
3. Use (2/3)n/3 2 bit wider CSAs for adding the (2/3)n results
4. Continue in a Wallace tree structure in the same manner
5. Delay is $T_{CSA} * \log_{1.5}n$ since the tree depth is $\log_{1.5}n$
6. Since $T_{CSA} = T_{FA}$ we get delay of $O(\log_{1.5}n)$
7. Cost is C_{CSA}* $[n/3+ (2/3)n/3+(2/3)^2n/3+\ldots] = O(n^2)$ CHECK!!!

Wallace tree – an example – 8 summands

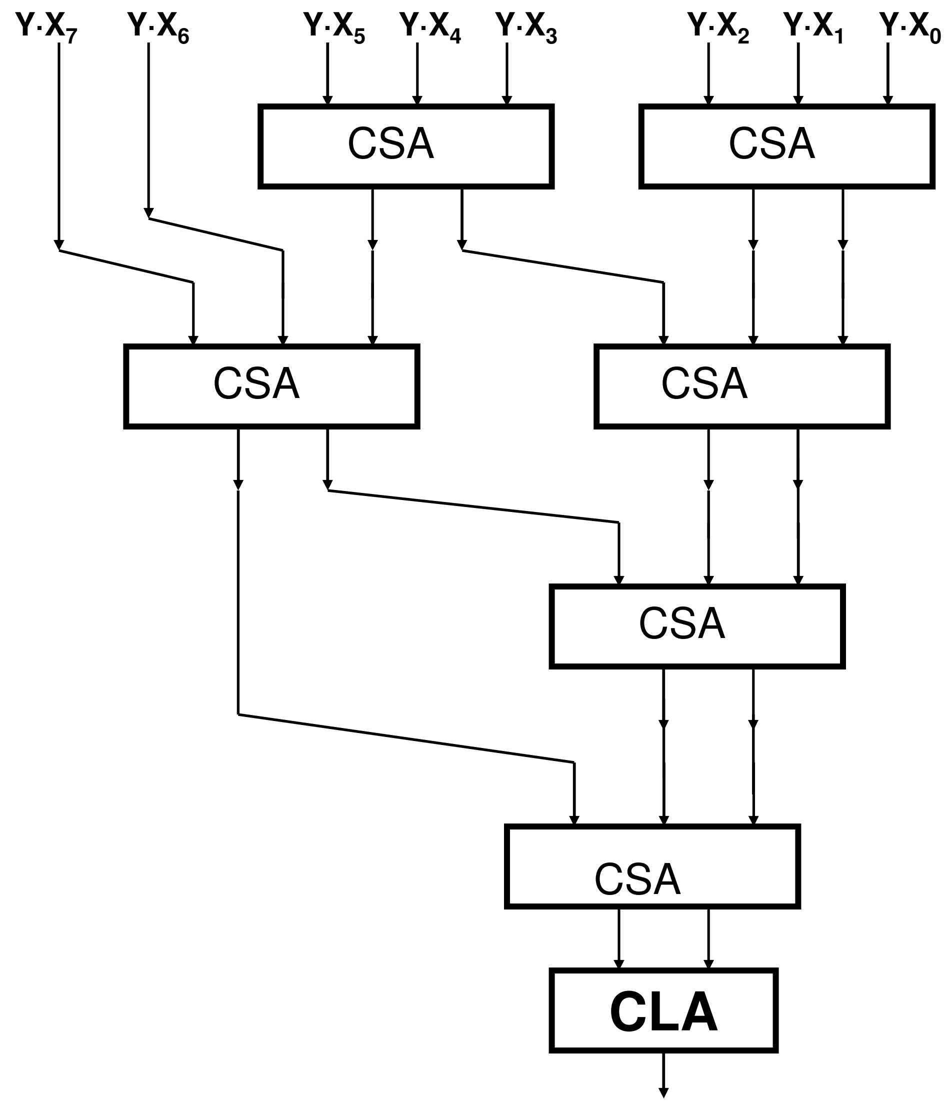
summands => levels:

3=>1, 4=>2, 5-6=>3,

7-9=>4, 10-13=>5,

14 -19=>6, 20-28=>7,

29-42=>8, 43-63=>9



A note:

The ratio 2/3 comes from the fact that a FA has inputs and 2 outputs

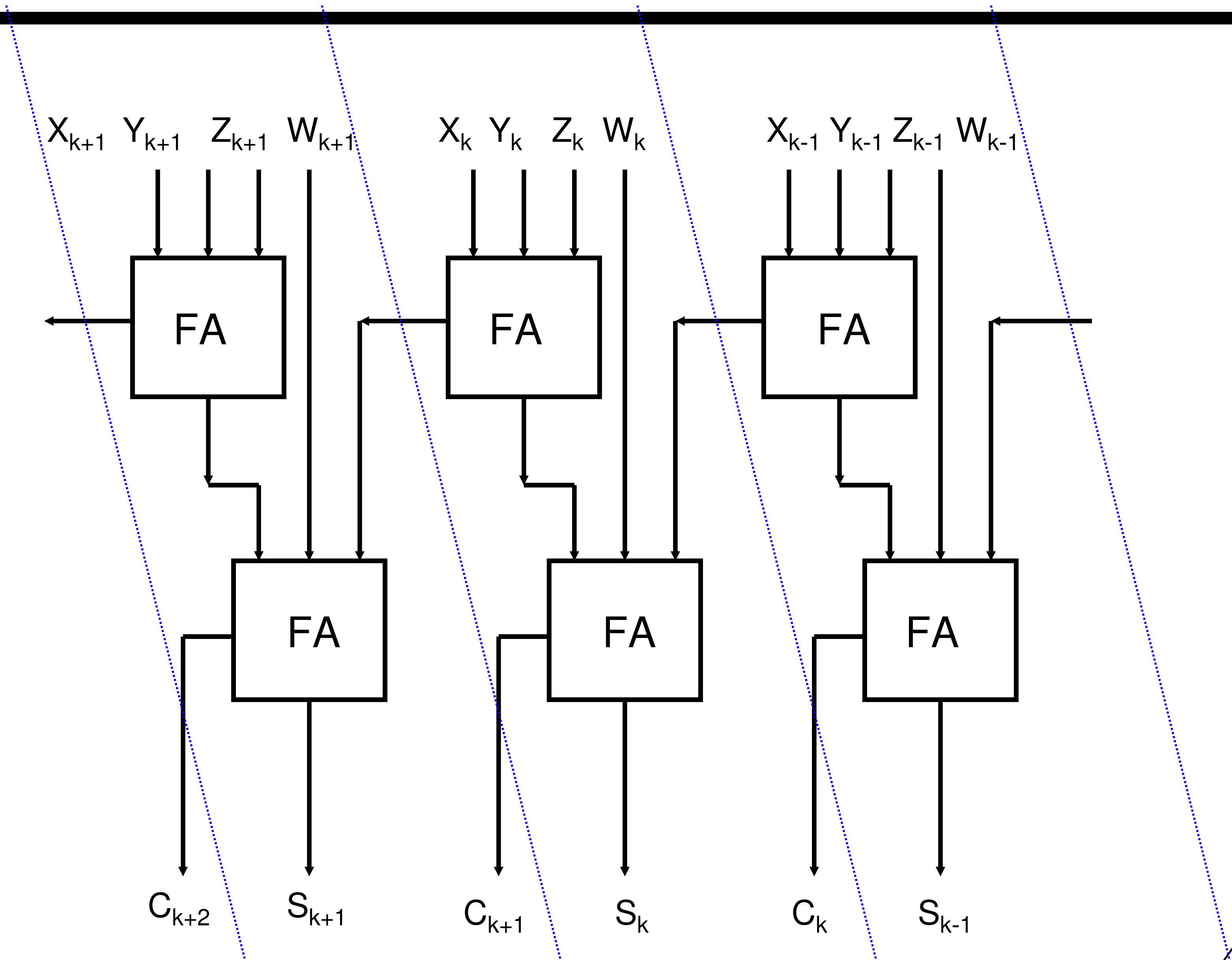
(looks like the last part of the Odd/Even CSA multiplier)

4. Binary tree with CSAs

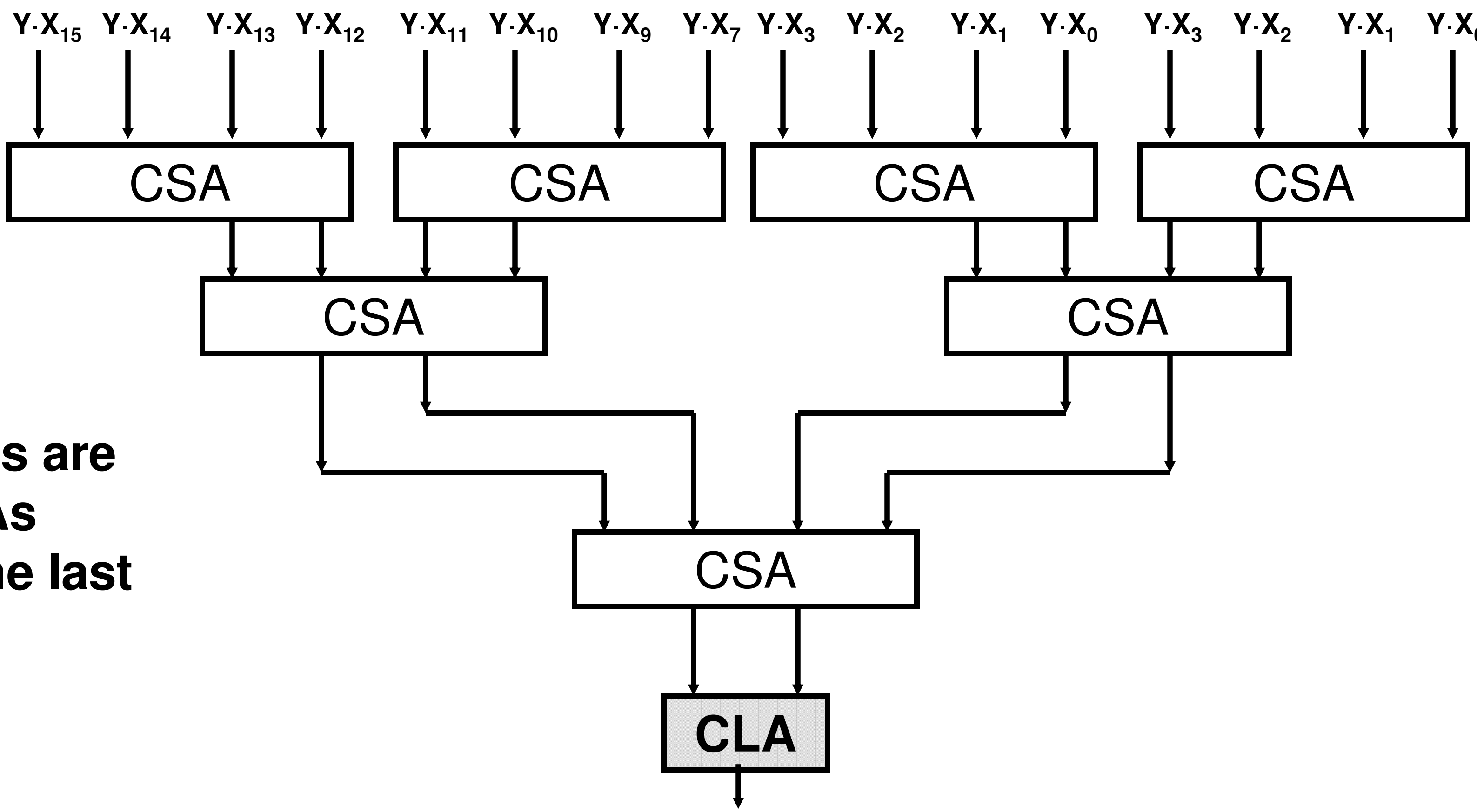
(See Guy Even' lectures)

1. O($\log_2 n$)
2. Use (n/2) 4-to-2 CSA structure for adding [$8Y_3+4Y_2+2Y_1+Y_0$], [$8Y_7+4Y_6+2Y_5+Y_4$] , etc.
3. Use (n/4) wider 4-to-2 CSAs for adding the (n/2) results
4. Continue in a binary tree like structure in the same manner
5. Delay is $T_{CSA} * \log_2 n$ since the tree depth is $\log_2 n$
6. Since $T_{CSA} = T_{FA}$ we get $O(\log_2 n)$
7. Cost = C_{CSA}*n*(1/2+1/4+…+1/2^{n-1}) =n*C_{CSA} (with average of 1.5n bits??) CHECK!!!

4-to-2 CSA


X_{k+1} Y_{k+1} Z_{k+1} W_{k+1}
X_k Y_k Z_k W_k
X_{k-1} Y_{k-1} Z_{k-1} W_{k-1}
FA
FA
FA
FA
FA
FA
C_{k+2} S_{k+1}
C_{k+1} S_k
C_k S_{k-1}

Binary tree with CSAs



All adders are 4to2 CSAs except the last one

Delay is $T_{CSA}*\log_2 n+T_{CLA}$

$= 2*T_{FA}*\log_2 n$