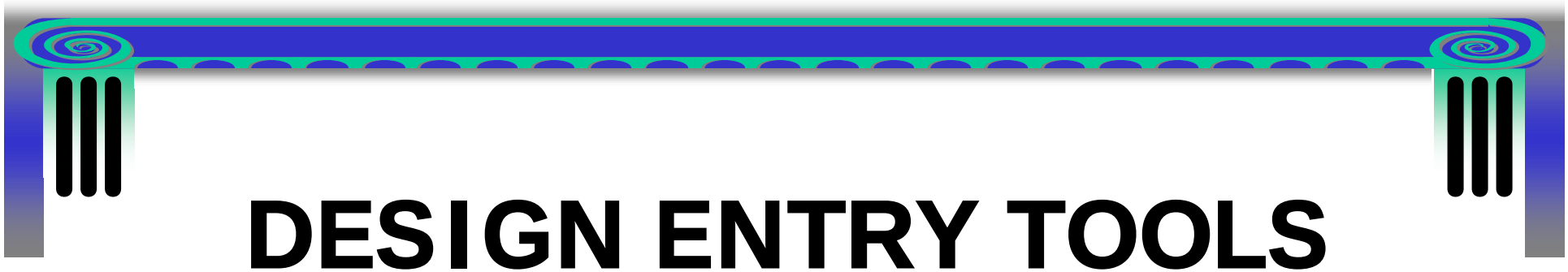
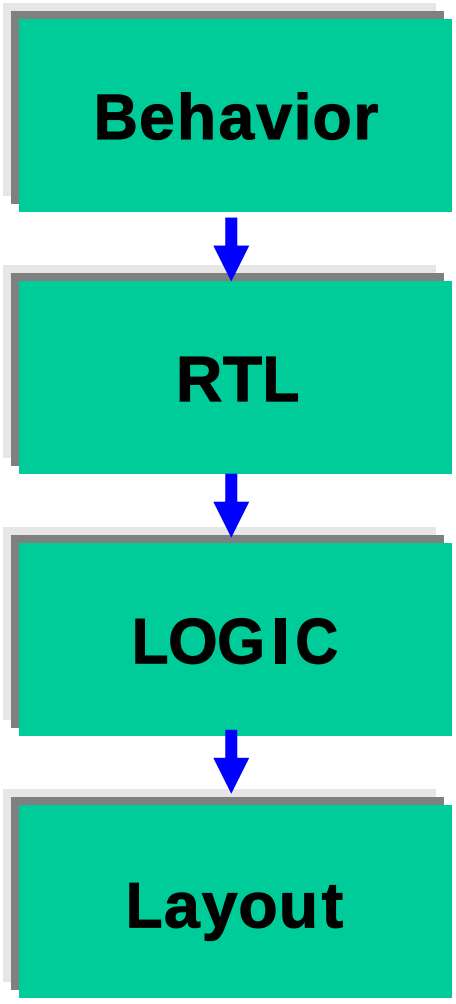


DESIGN ENTRY TOOLS AND TECHNIQUES

Nazar Abbas Saqib

Supervisor: Dr. Arturo Diaz Perez

ABSTRACTION LEVELS



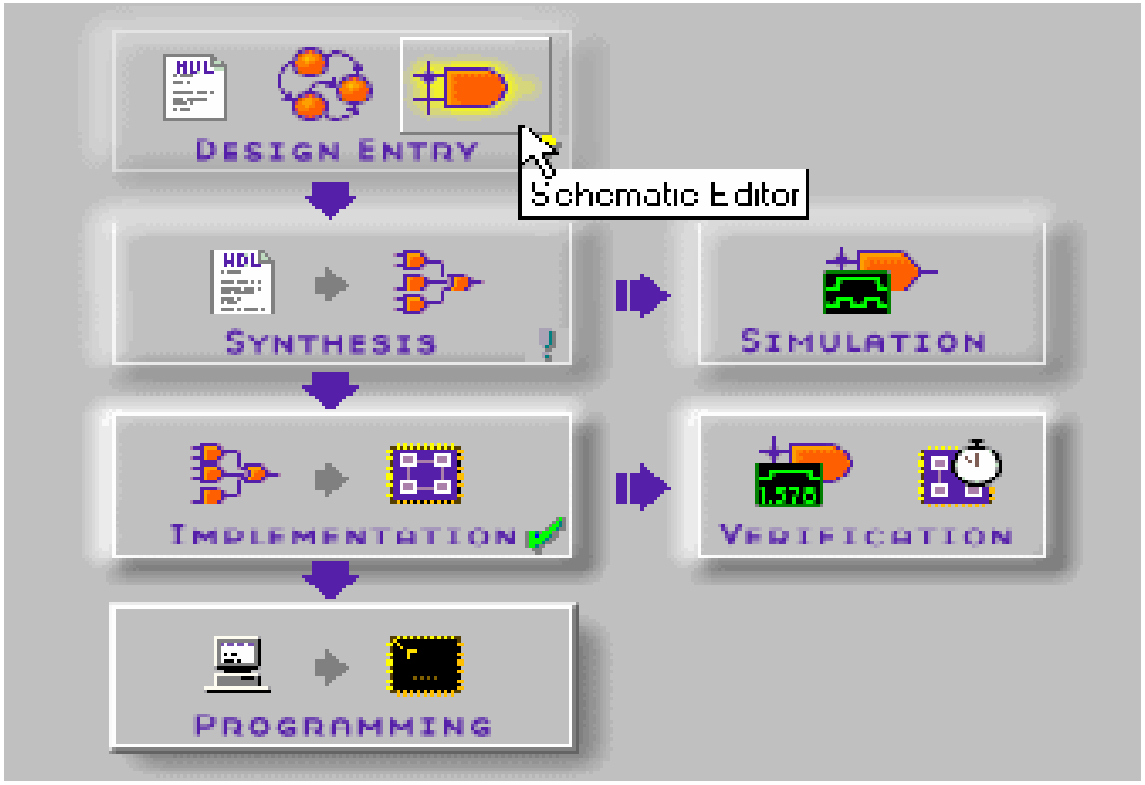
Algorithm level

Registers,Logic,Clock

Netlist, gate structure

Technology dependent

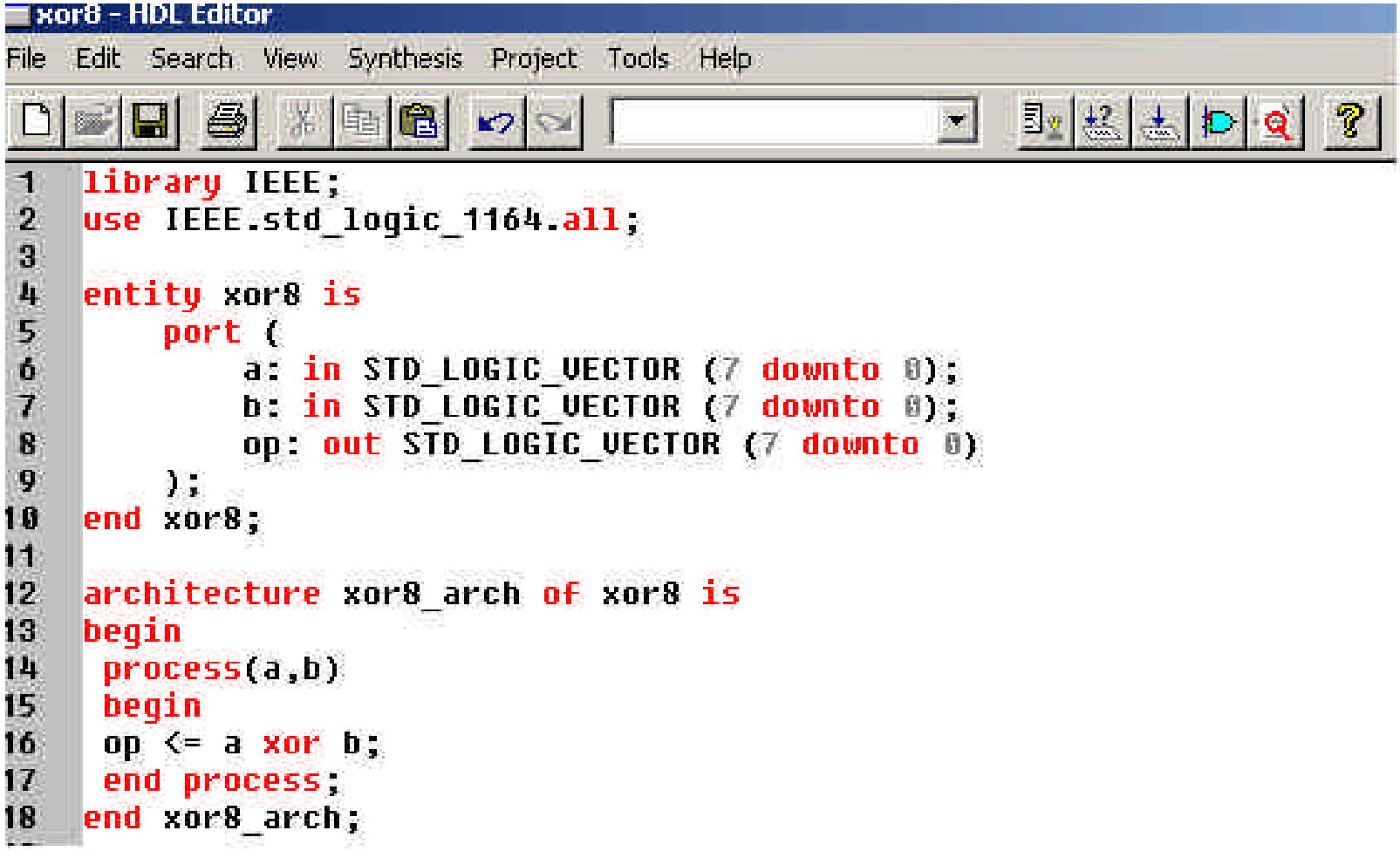
DESIGN TOOL
Xilinx Foundation 4.1i



DESIGN ENTRY

- VHDL (**V**ery high speed integerated circuits**H**ardware**D**escription**L**anguage)
- DEVICE LIBRARY COMPONENTS
- COREGENERATOR

DESIGN ENTRY (VHDL)



DESIGN ENTRY (VHDL)

DIVIDE BY 6 COUNTER

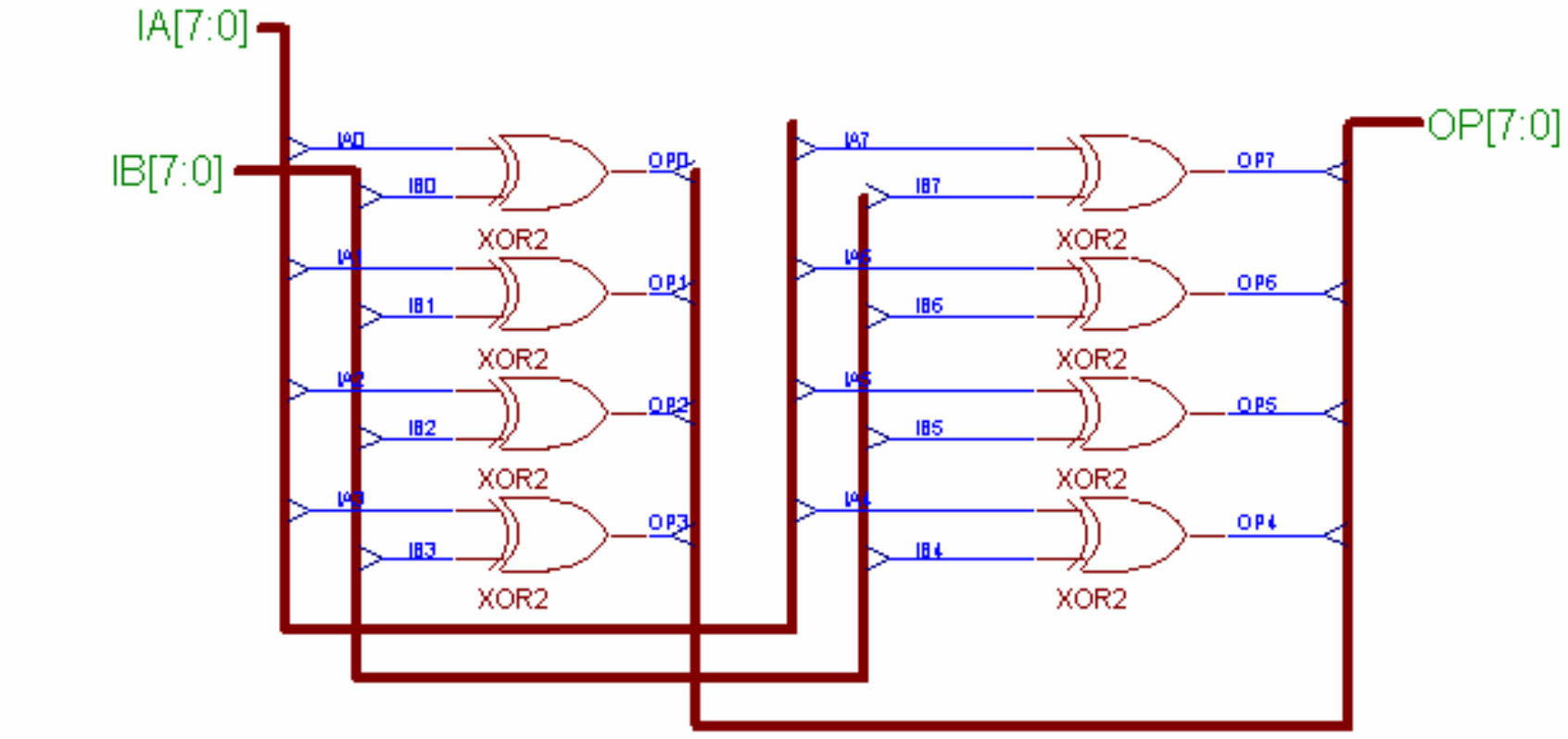
```
signal mod6_count_s : std_logic_vector( 2 downto 0);
begin
mod6_count_reg:
process (sys_reset_n_i,clk_i)
begin
if(sys_reset_n_i= '0') then
mod6_count_s <= "000";
elsif (clk_i'event and clk_i='1') then
if (mod6_count_s = "110") then
mod6_count_s <= (others => '0');       else
mod6_count_s <= mod6_count_s + 1 ;
end if;
end if;
end process mod6_count_reg;
mod6_count_o <= mod6_count_s ;
end mod6_count_rtl;
```

DESIGN ENTRY (VHDL)

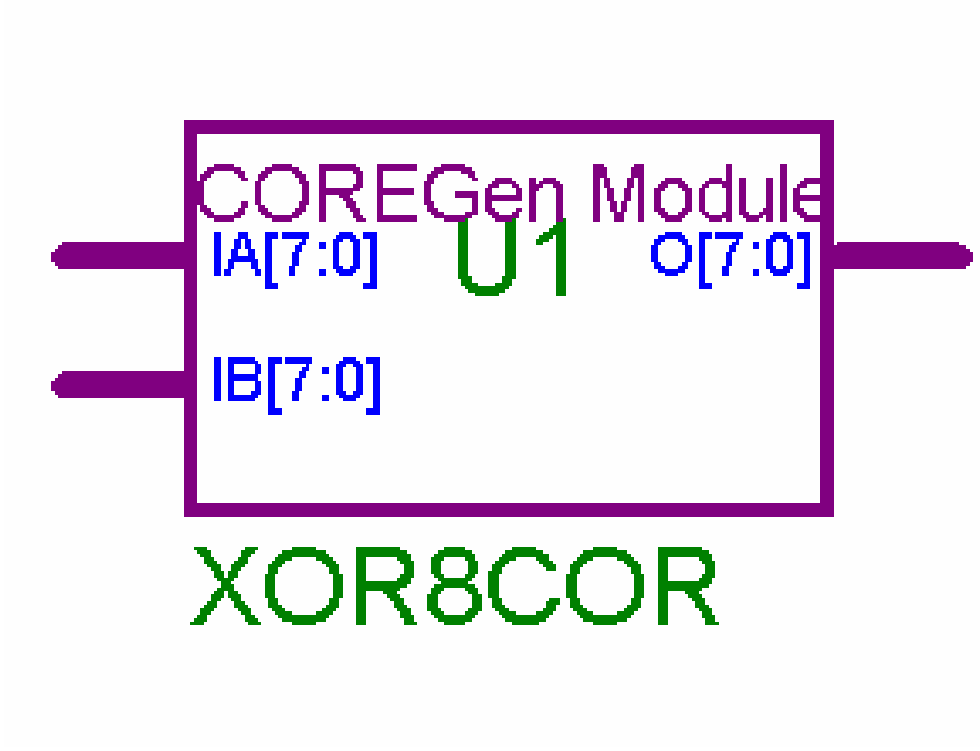
AN EFFICIENT METHODOLOGY

```
process(CE,CLK)
        variable count : std_logic_vector(6 downto 0);
        begin

            if CE = '0' then
                    count := "0000001" ;
            elsif CLK'event and CLK='1' then
                    count (6 downto 1) := count (5 downto 0);--shift --- register
                 count(0) := not count (6);--Last bit inverted ---- back into first bit
              S_M <= count(6);
             end if;

          end process;
```

DESIGN ENTRY
(Library components)



DESIGN ENTRY
(Coregenerator)



DESIGN ENTRY

Which one is better?

VHDL
Library Components
Coregenertor

!!!!!Write VHDL code like Coregenerator

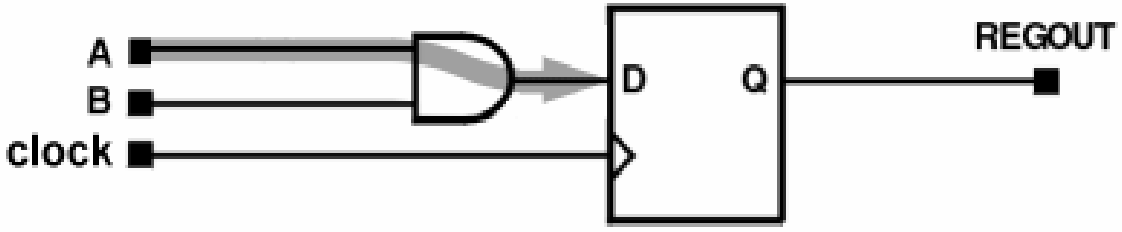
DESIGN IMPLEMENTATION

- FLOW ENGINE
- CONSTRAINT EDITOR
- FLOOR PLANNING

DESIGN CONSTRAINT

pad to setup time (P2S)

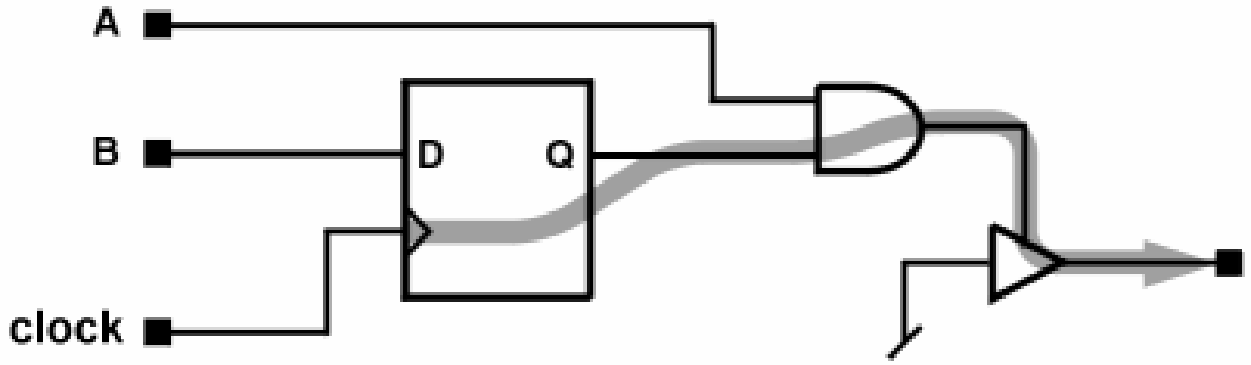
The maximum time required for the data to enter the chip, travel through logic and routing, and arrive at the input pin of the first synchronous element (flip-flop, latch, or RAM) where that pin has a setup requirement before a clocking signal. Setting a pad to setup time creates an OFFSET IN BEFORE constraint in the UCF file.



DESIGN CONSTRAINT

clock to pad time.

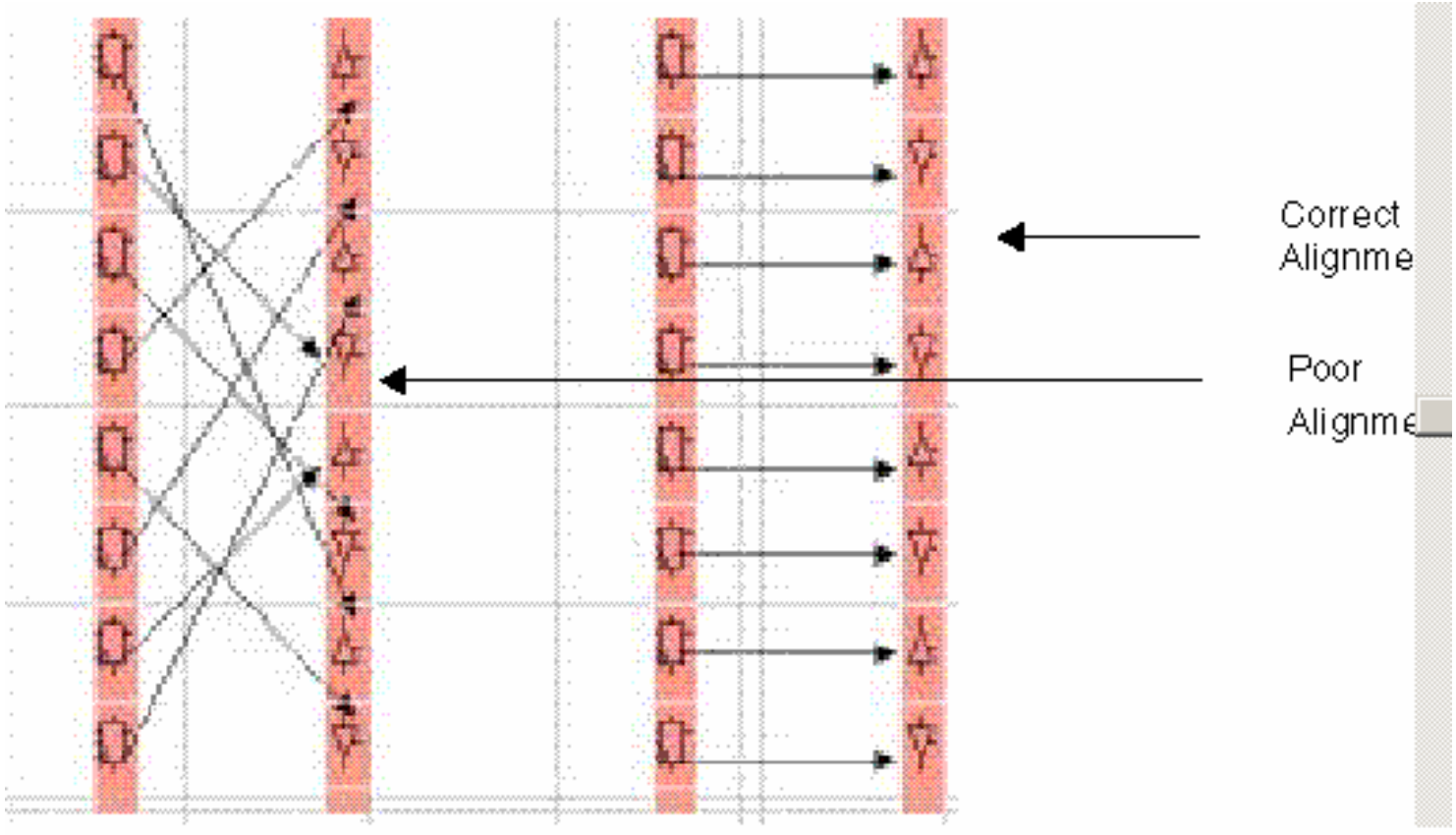
The maximum time required for data at the input of a flip-flop or latch to travel through logic and routing and arrive at the output of the chip before the next clock edge. It includes the clock-to-Q delay of the source flip-flop and the path delay from that flip-flop to the output pad. Setting a clock to pad time creates an OFFSET OUT AFTER constraint in the UCF file.



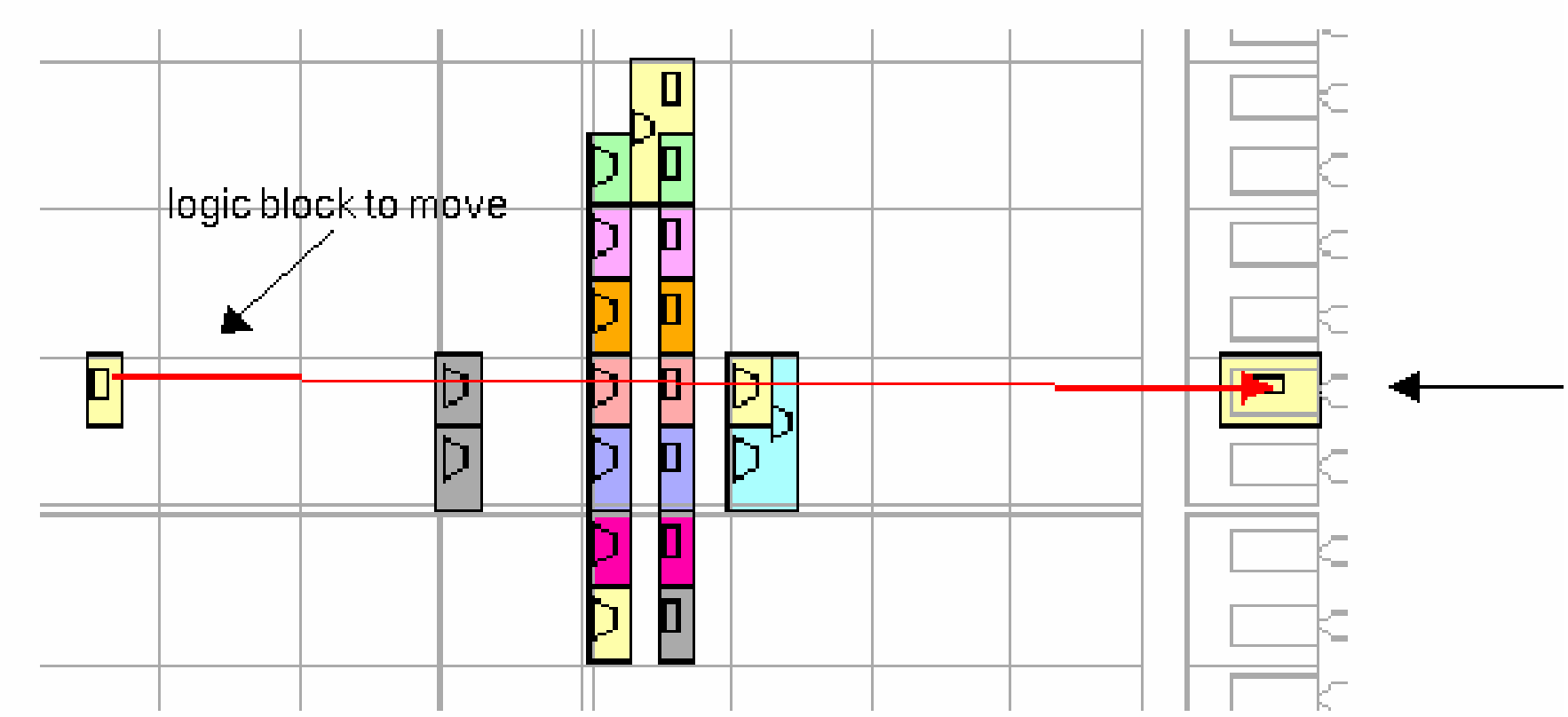
DESIGN CONSTRAINT

- USESKEWLINES
- FIX VOLTAGE
- FIX TEMPERATURE
- SET IO TO PULLDOWN OR PULLUP
- OUTPUT FAST/SLOW
- ETC.

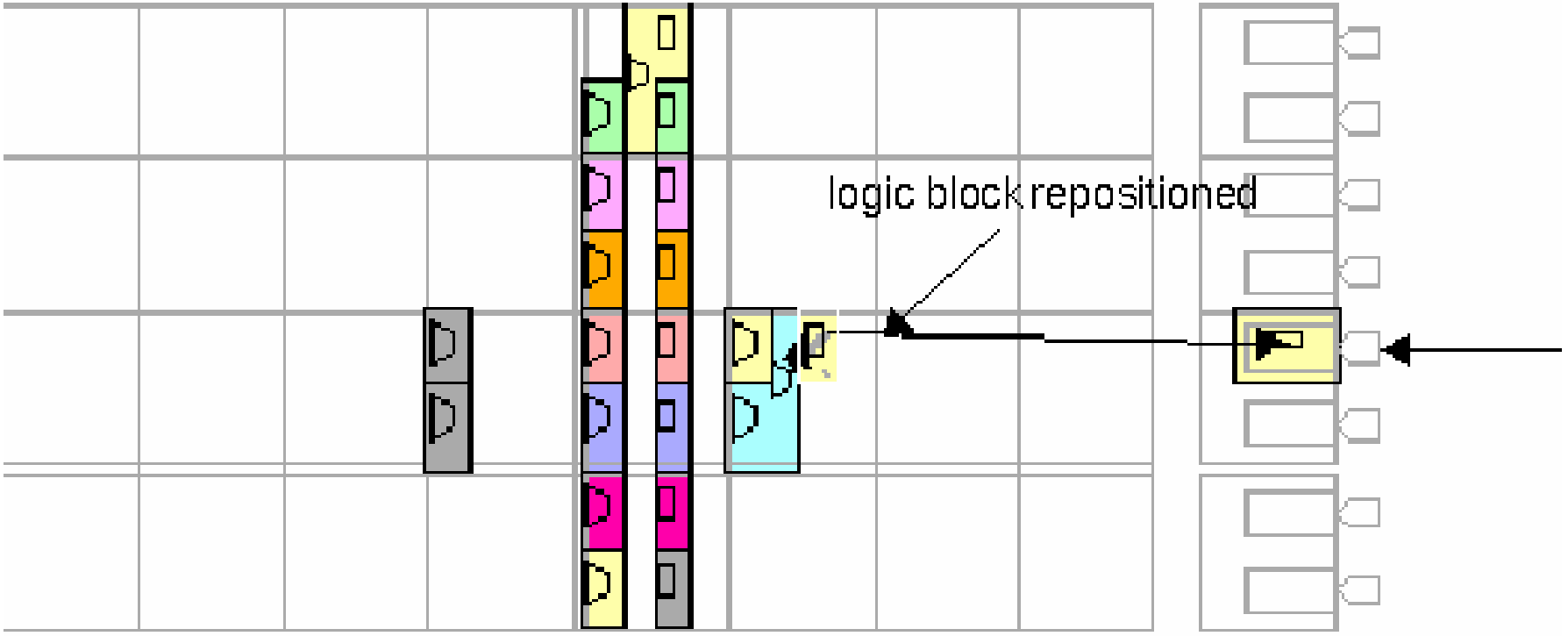
FLOOR PLANNING
FALSE & TRUE ROUTING



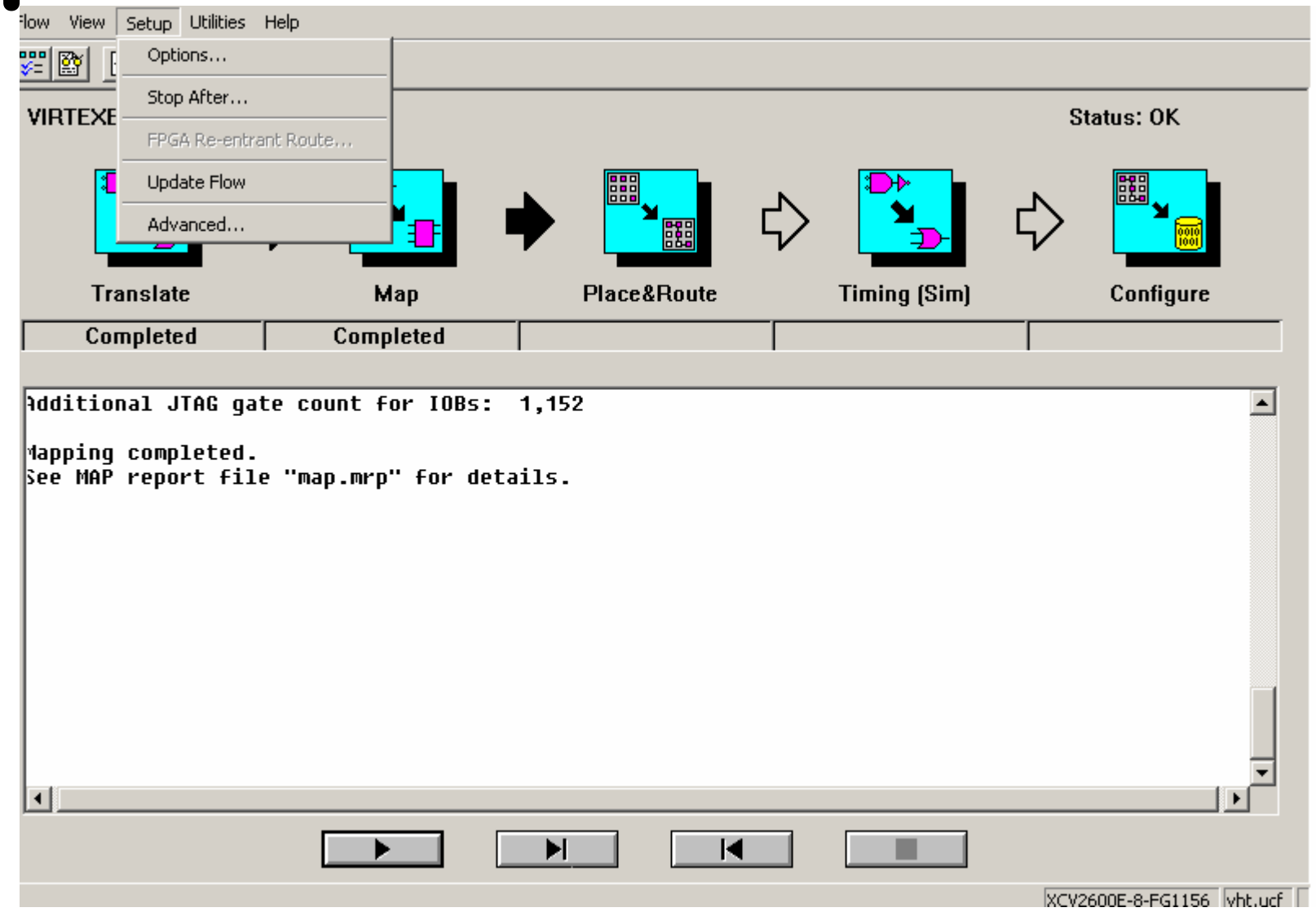
FLOOR PLANNING
False placement

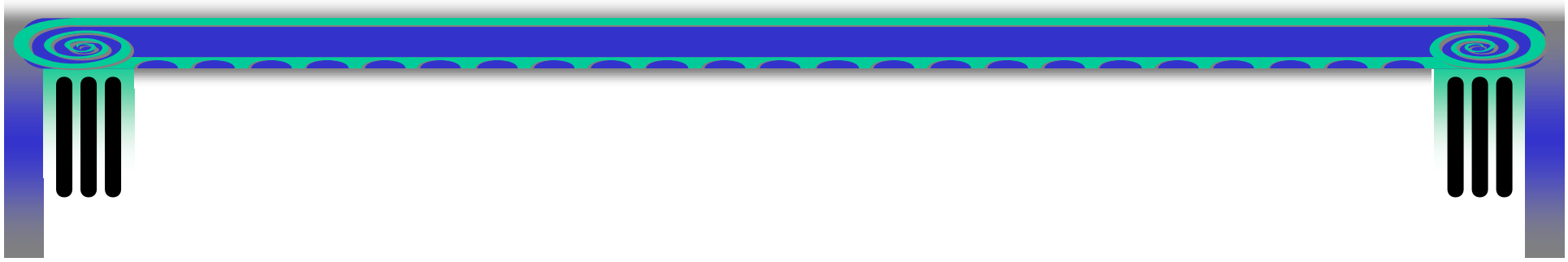


FLOOR PLANNING
Correct placement



FLOW ENGINE


Flow View Setup Utilities Help
Options...
Stop After...
FPGA Re-entrant Route...
Update Flow
Advanced...
VIRTEXE
Status: OK
Translate
Map
Place&Route
Timing (Sim)
Configure
Completed
Completed
Additional JTAG gate count for IOBs: 1,152
Mapping completed.
See MAP report file "map.mrp" for details.
XCV2600E-8-FG1156
vht.ucf



APPLICATION
BINARY KARATSUBA MULTIPLIER
GF(2^{193})

BINARY KARATSUBA GF(2^{193}) IMPLEMENTATION

$$C(x) = A(x)B(x) = \left(\sum_{i=0}^{m-1} a_i x^i \right) \left(\sum_{i=0}^{m-1} b_i x^i \right)$$

$$C'(x) = C(x) \bmod P(x) \quad ?$$

$$A = \sum_{i=0}^{m-1} a_i x^i = \sum_{i=\frac{m}{2}}^{m-1} a_i x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i$$

$$= x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i = x^{\frac{m}{2}} A^H + A^L$$

$$B = \sum_{i=0}^{m-1} b_i x^i = \sum_{i=\frac{m}{2}}^{m-1} b_i x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i$$

$$= x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} b_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i = x^{\frac{m}{2}} B^H + B^L$$

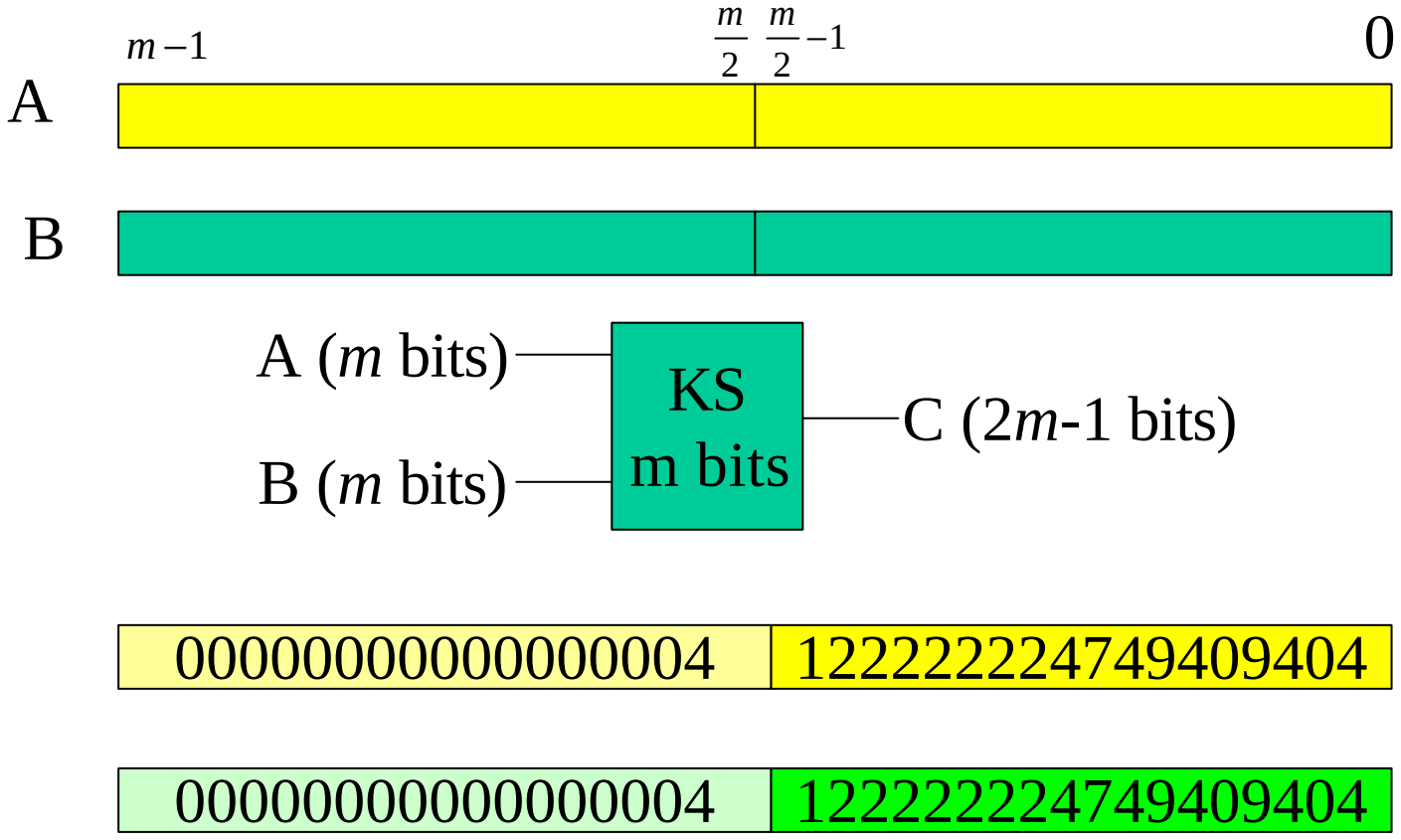
Classical approach

$$C = x^{m} A^{H} B^{H} + \left(A^{H} B^{L} + A^{L} B^{H} \right) x^{\frac{m}{2}} + A^{L} B^{L}$$

Karatsuba approach

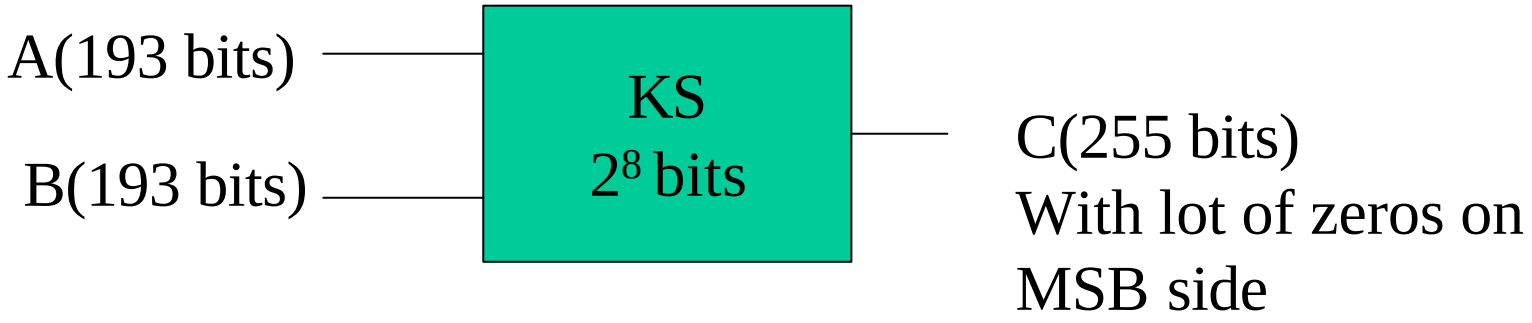
$$C = x^{m} A^{H} B^{H} + \left(A^{H} B^{H} + A^{L} B^{L} + \left(A^{H} + A^{L} \right) \left(B^{H} + B^{L} \right) \right) x^{\frac{m}{2}} + A^{L} B^{L}$$

BINARY KARATSUBA GF(2^{193}) IMPLEMENTATION



BINARY KARATSUBA GF(2^{193}) IMPLEMENTATION

For example
m=193=2^7 + 65 bits
USE KARATSUBA MULTIPLIER OF 2^8



To save lot of arithmatic operations, we can use binary approach?

BINARY KARATSUBA GF(2^{193}) IMPLEMENTATION

A_H	A_L
65	128

A_{HH}	A_{HL}
1	64

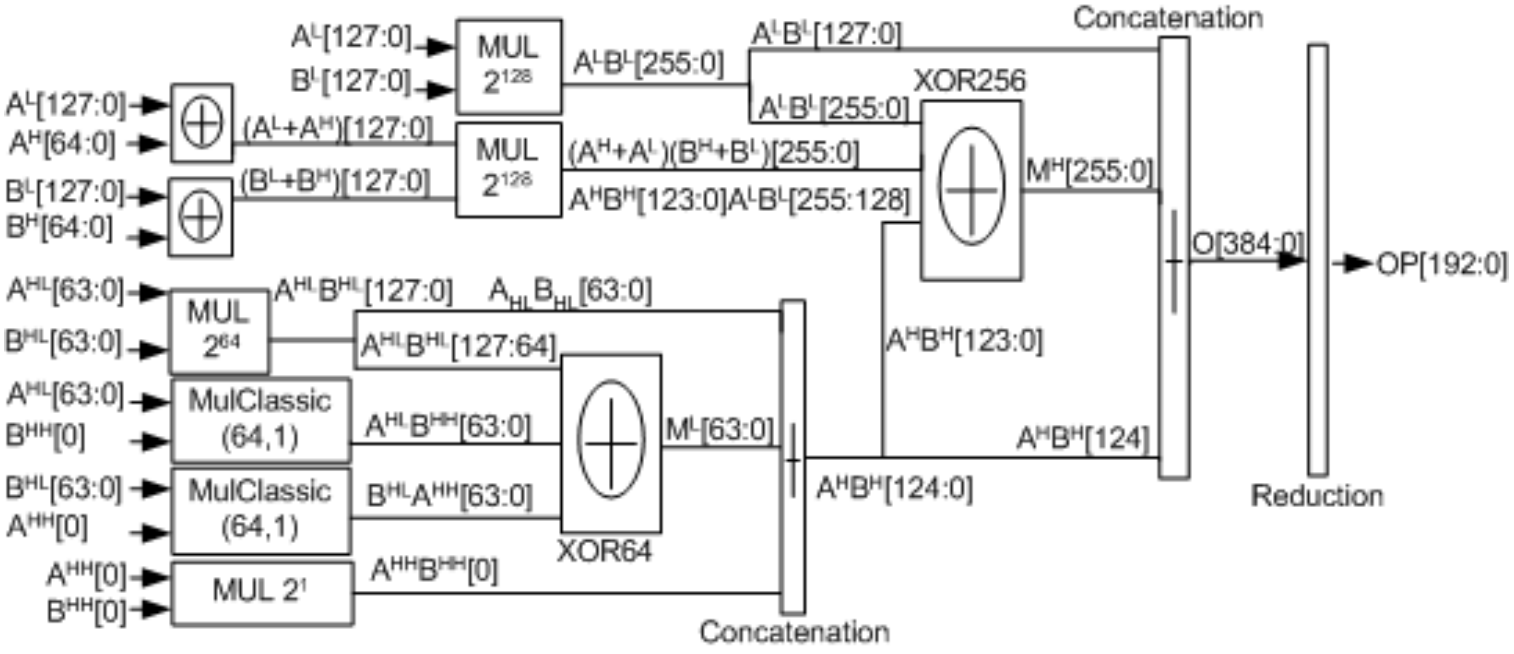
B_H	B_L
65	128

B_{HH}	B_{HL}
1	64

A_LB_L=128 bits $(A_L+A_H)(B_H+B_L)$=128 bits

A_HB_H=65x65=???

BINARY KARATSUBA GF(2^{193}) IMPLEMENTATION



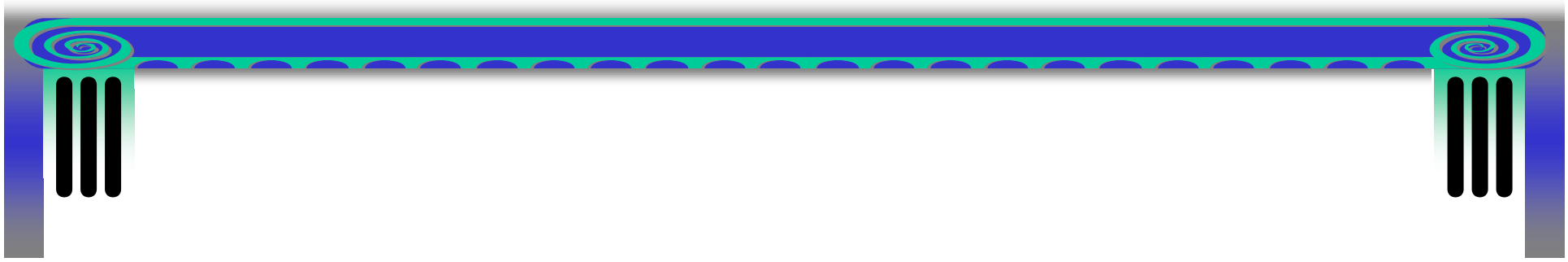
A_H		A_L
65		128
A_{HH}	A_{HL}	
1	64	

B_H		B_L
65		128
B_{HH}	B_{HL}	
1	64	

Conclusions

To have an efficient implementation,

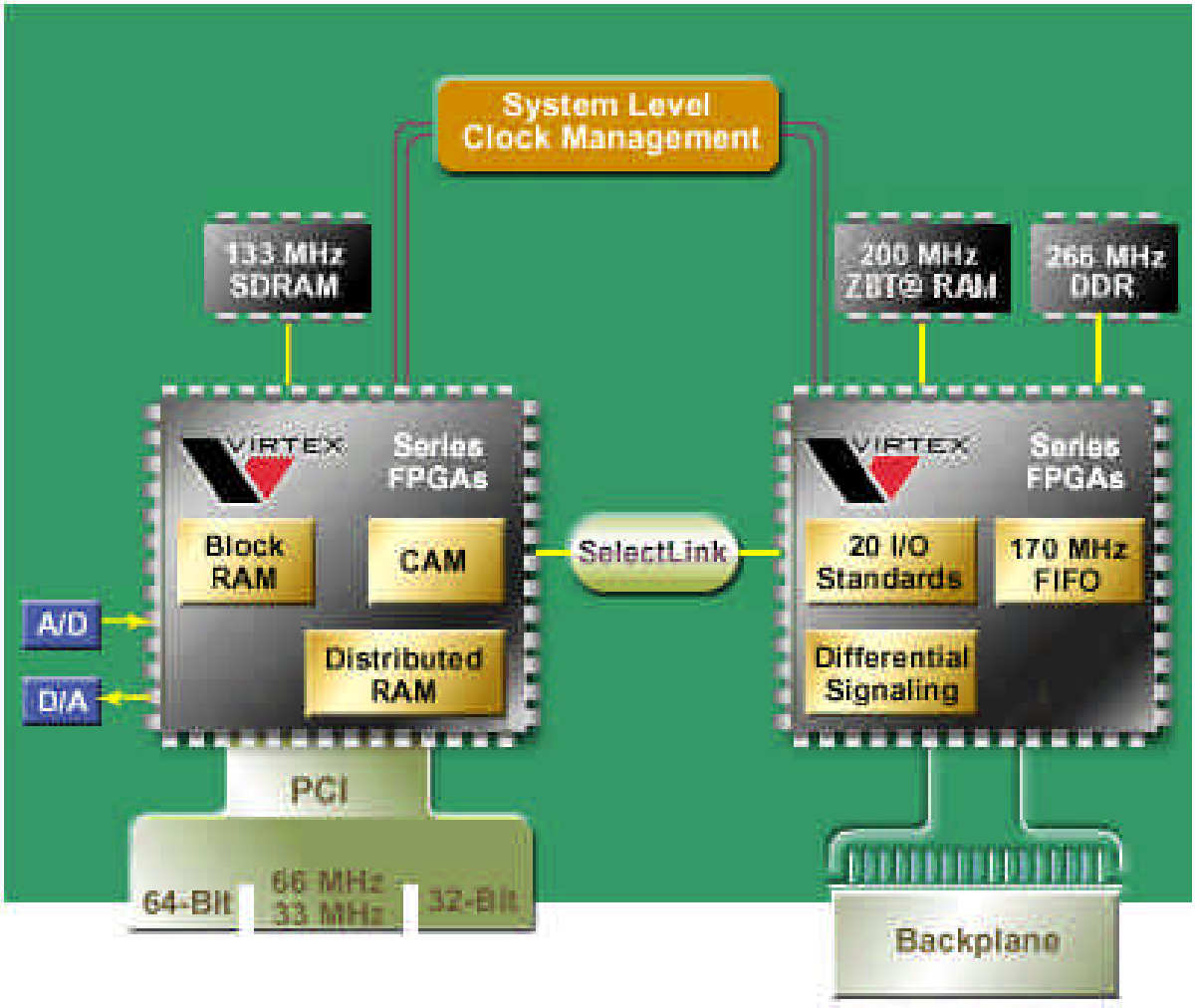
- **Good algorithmic techniques**
- **Good designing techniques**
- **Good implementing techniques**
- **Skill to use Desinging tool**



DESIGN ENTRY TOOLS AND TECHNIQUES

(Advanced Design Techniques)

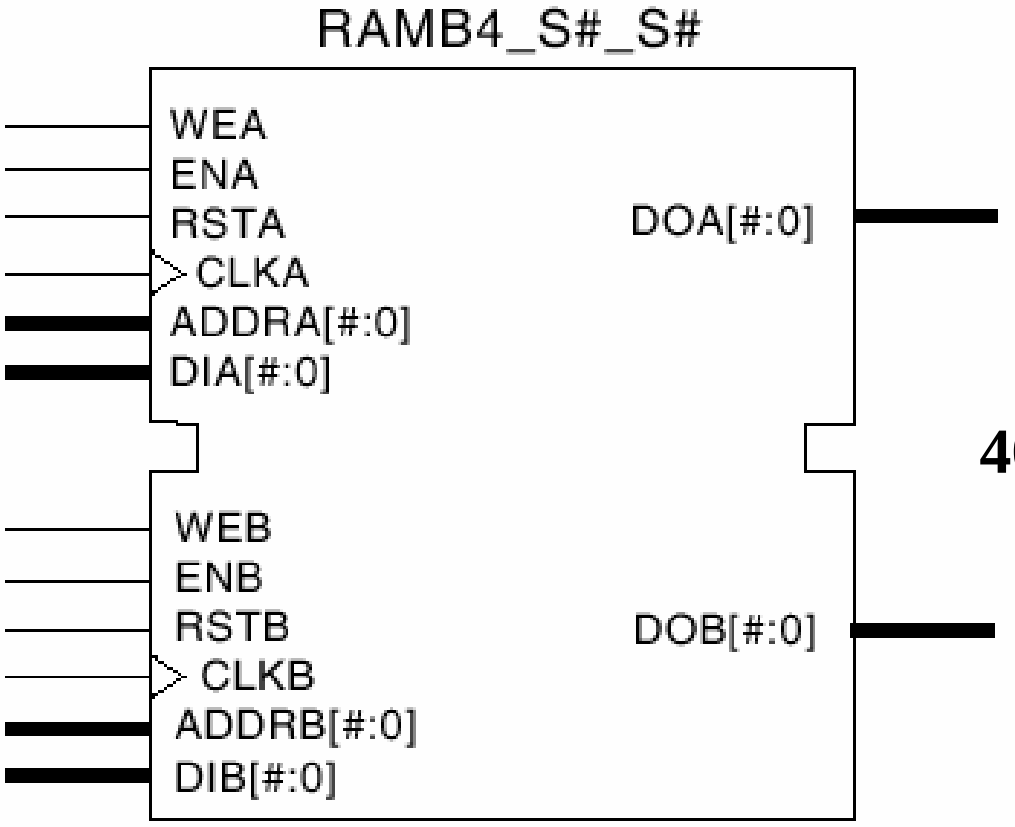
Virtex Series FPGAs



Virtex-II Pro

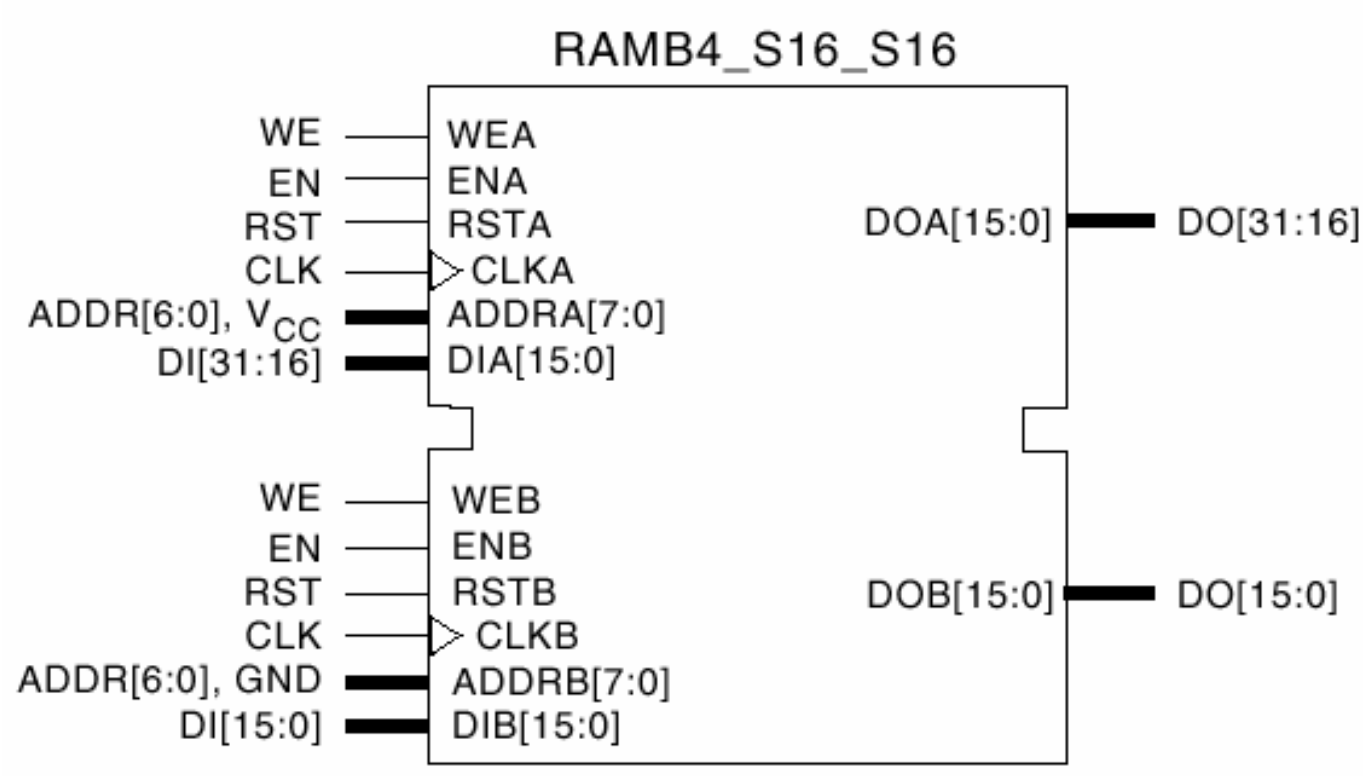
Feature/Product	XC 2VP2	XC 2VP4	XC 2VP7	XC 2VP20	XC 2VP30	XC 2VP40	XC 2VP50	XC 2VP70	XC 2VP100	XC 2VP125
EasyPath cost reduction	-	-	-	-	XCE 2VP30	XCE 2VP40	XCE 2VP50	XCE 2VP70	XCE 2VP100	XCE 2VP125
Logic Cells	3,168	6,768	11,088	20,880	30,816	43,632	53,136	74,448	99,216	125,136
Slices	1,408	3,008	4,928	9,280	13,696	19,392	23,616	33,088	44,096	55,616
BRAM (Kbits)	216	504	792	1,584	2,448	3,456	4,176	5,904	7,992	10,008
18x18 Multipliers	12	28	44	88	136	192	232	328	444	556
Digital Clock Management Blocks	4	4	4	8	8	8	8	8	12	12
Config (Mbits)	1.31	3.01	4.49	8.21	11.36	15.56	19.02	25.6	33.65	42.78
PowerPC Processors	0	1	1	2	2	2	2	2	2	4
Max Available Multi-Gigabit Transceivers*	4	4	8	8	8	12*	16*	20	20*	24*
Max Available User I/O*	204	348	396	564	644	804	852	996	1164	1200

VIRTEX & VIRTEXE BLOCKRAMS (BRAMs)



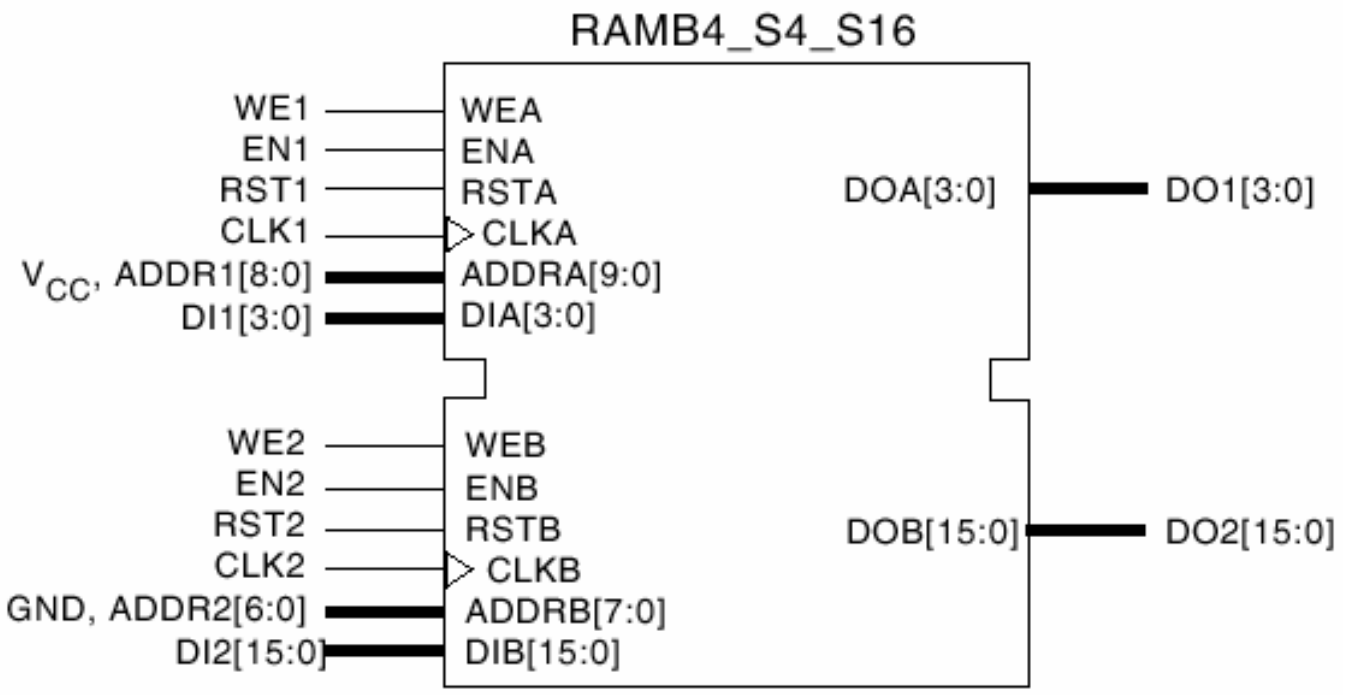
4096 memory cells

SINGLE-PORT BLOCKRAM(BRAM)

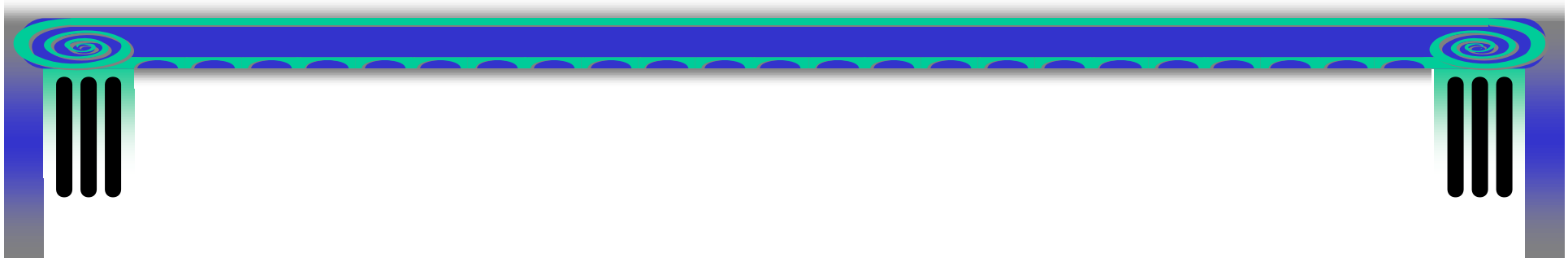


128 x 32 = 4096

DUAL-PORT BLOCKRAM(BRAM)



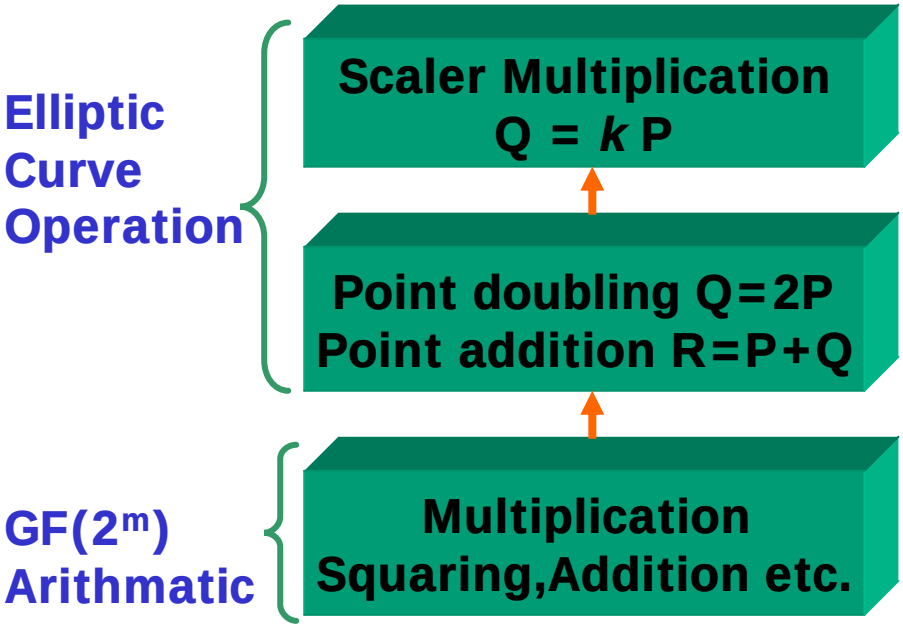
512 x 4 = 2048
128 x 16 = 2048
Total = 4096



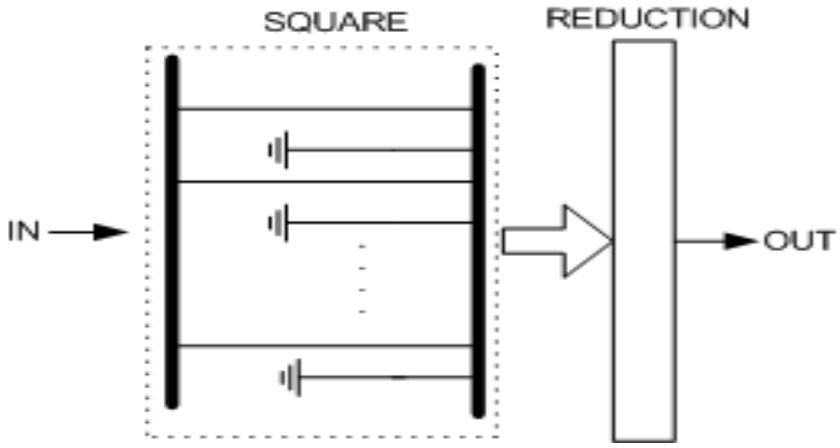
AN APPLICATION TO BRAMS

ELLIPTIC CURVE IMPLEMENTATION

ELLIPTIC CURVE SCALAR MULTIPLICATION



GF(2^{191}) Arithmetic-Square

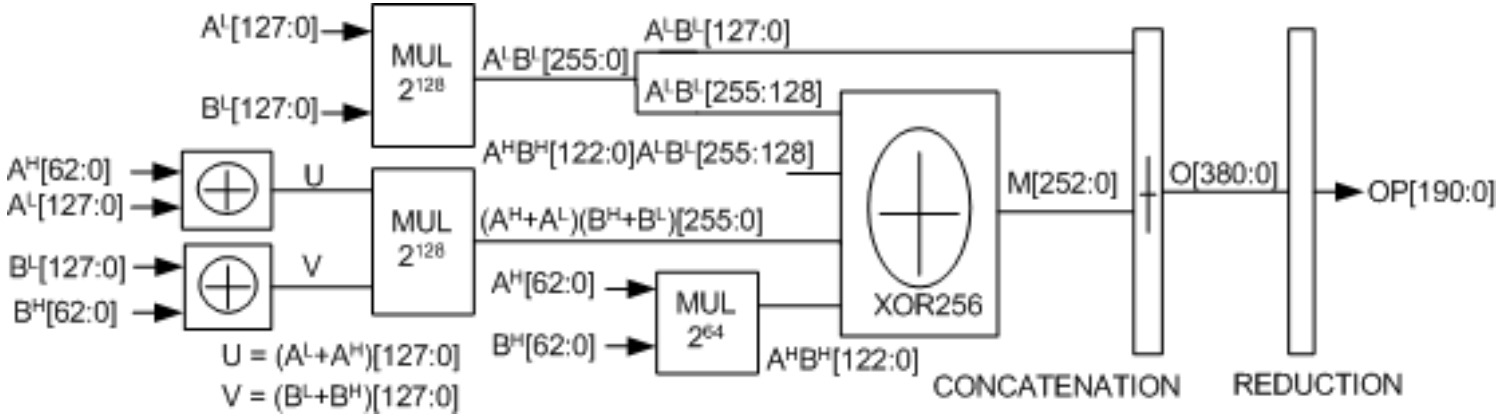


$$A = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

$$A^2 = a_6 x^6 + a_4 x^4 + a_2 x^2 + a_0$$

A = 1111
A^2 = 1010101

Karatsuba Multiplier GF(2^{191})



$$A = \sum_{i=0}^{m-1} a_i x^i = \sum_{i=\frac{m}{2}}^{m-1} a_i x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i = x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i = x^{\frac{m}{2}} A^H + A^L$$

$$B = \sum_{i=0}^{m-1} b_i x^i = \sum_{i=\frac{m}{2}}^{m-1} b_i x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i = x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} b_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i = x^{\frac{m}{2}} B^H + B^L$$

Then Polynomial multiplication of A and B is:

$$C = x^m A^H B^H + \left(A^H B^L + A^L B^H\right) x^{\frac{m}{2}} + A^L B^L$$

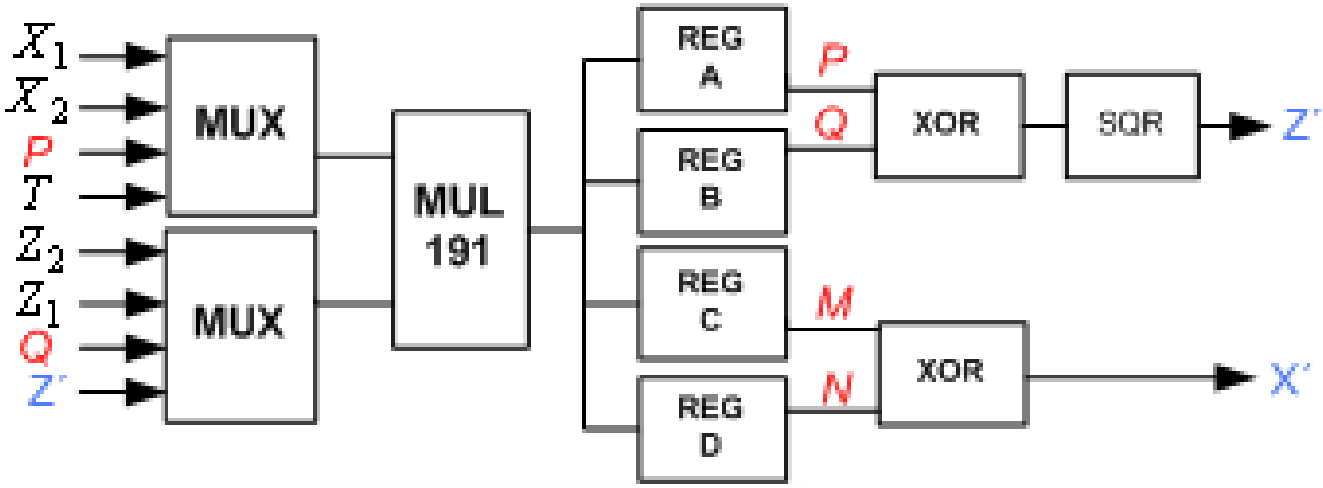
The karatsuba algorithm has an idea that the above product can be written as:

$$C = x^m A^H B^H + \left(A^H B^H + A^L B^L + \left(A^H + A^L\right)\left(B^H + B^L\right)\right) x^{\frac{m}{2}} + A^L B^L$$

$$= x^m C^H + C^L$$

Point Addition in GF(2^{191})

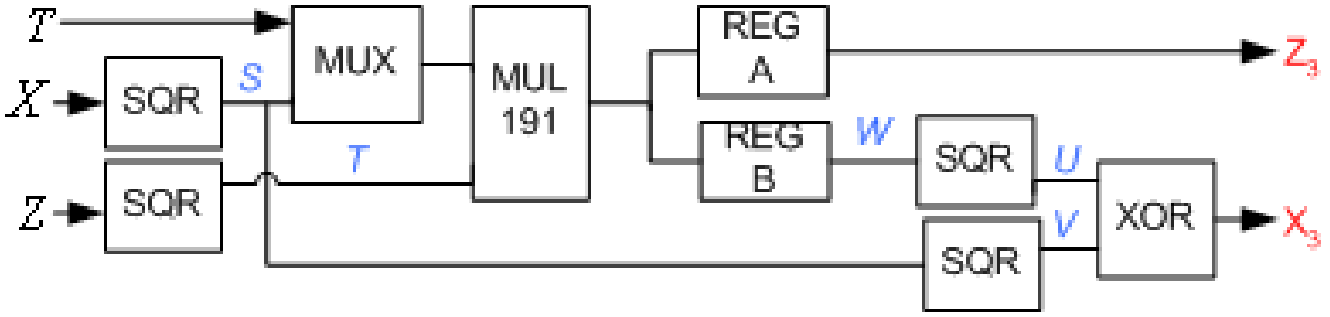
$$P(X',Z') = P(X_1,Z_1) + P(X_2,Z_2)$$



$$\begin{aligned}
T_2 &= x \\
P &= X_1 \times Z_2 \\
Q &= Z_1 \times X_2 \\
R &= P + Q \\
Z' &= R^2 \\
M &= P \times Q \\
N &= T_2 \times Z' \\
X' &= M + N
\end{aligned}$$

Point Doubling in GF(2^{191})

$P(X_3,Z_3) = P(X,Z) + P(X,Z)$



$$
\begin{aligned}
T_1 &= c \\
S &= X_2^2 \\
T &= Z_2^2 \\
Z_3 &= S \times T \\
W &= T \times T_1 \\
U &= W^2 \\
V &= S^2 \\
X_3 &= U + V
\end{aligned}
$$

Table 1. $GF(2^m)$ Elliptic Curve Point Multiplication Computational Costs

Strategy		Required No. of Field Multipliers	EC Operation Cost		Total Number of Field Multiplications
2nd Layer	3rd Layer		Doubling	Addition	
S	S	1	$2M$	$4M$	6mM
S	P	2	$2M$	$4M$	4mM
P	S	2	M	$2M$	3mM
P	P	4	M	$2M$	2mM

Point Addition **Point Doubling**

$$
\begin{aligned}
T_2 &= x \\
P &= X_1 \times Z_2 \\
Q &= Z_1 \times X_2 \\
R &= P + Q \\
Z' &= R^2 \\
M &= P \times Q \\
N &= T_2 \times Z' \\
X' &= M + N
\end{aligned}
$$

$$
\begin{aligned}
T_1 &= c \\
S &= X_2^2 \\
T &= Z_2^2 \\
Z_3 &= S \times T \\
W &= T \times T_1 \\
U &= W^2 \\
V &= S^2 \\
X_3 &= U + V
\end{aligned}
$$

Input: $k = (k_{n-1}, k_{n-2}....., k_1, k_0)_2$ with $k_{n-1} = 1$, $P(x, y) \in E(F_{2^m})$
Output: $Q = kP$

1. Set $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$
2. For i from $n-2$ downto **0** do
3. if $(k_i = 1)$ then
4. Madd(X_1, Z_1, X_2, Z_2), Mdouble(X_2, Z_2)
5. else
6. Madd(X_2, Z_2, X_1, Z_1), Mdouble(X_1, Z_1)
7. Return$(Q = M_{xy}(X_1, Z_1, X_2, Z_2))$

Montgomery point multiplication

ELLIPTIC CURVE SCALAR MULTIPLICATION

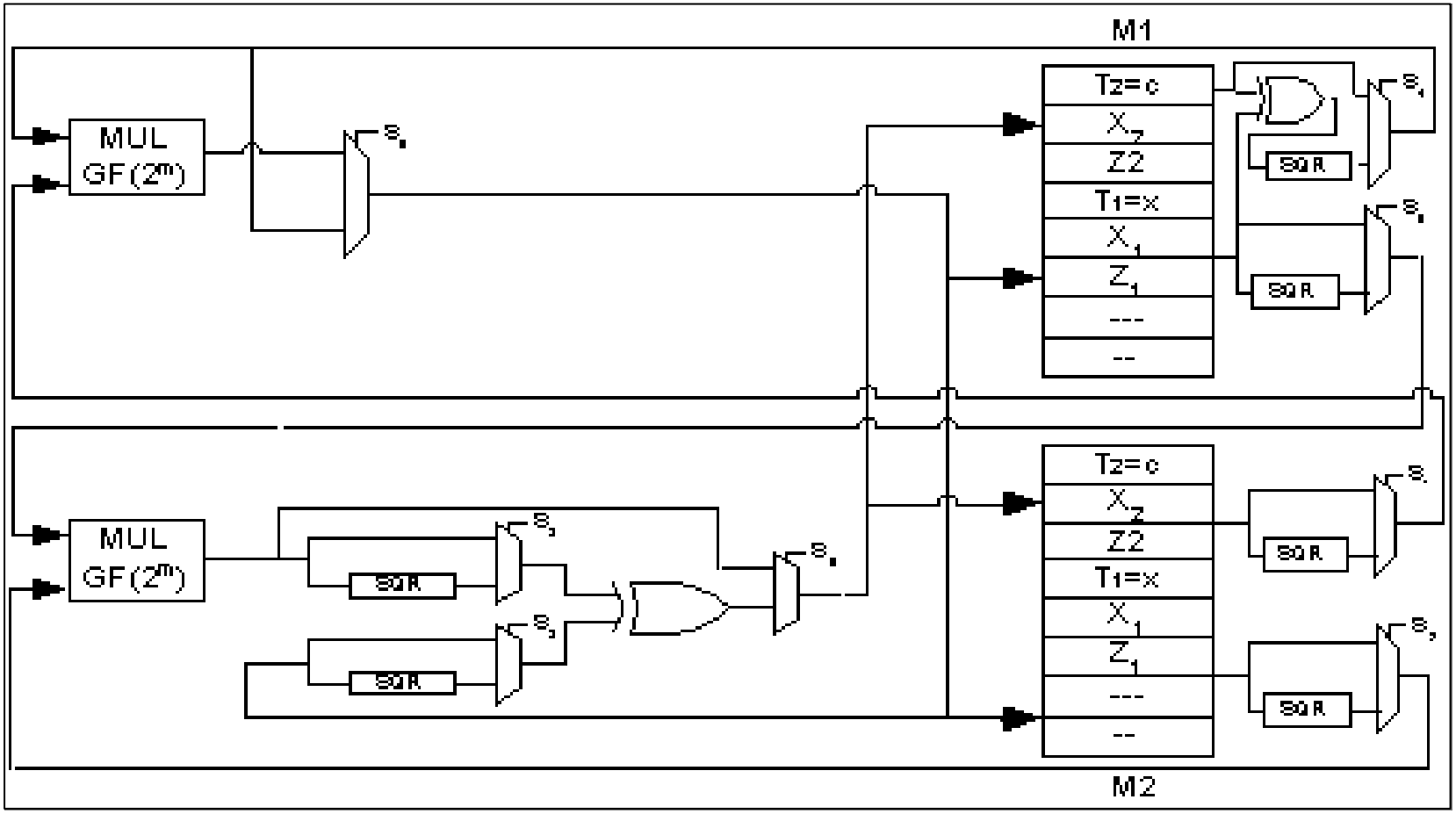


Table 2. kP **computation, if test-bit is '1'**

Cycle	Read				Write	
	M1		M2		M1/M2	
	$PT1$	$PT2$	$PT1$	$PT2$	$PT1$	$PT2$
1	X_1	Z_2	Z_1	X_2	P	Q
2	X_2	Z_2	Z_2	T_1	Z_2=Z_3	X_2=X_3
3	P	Q	Q	T_2	X_1=X'	Z_1=Z'

Table 3. kP **computation, if test-bit is '0'**

Cycle	Read				Write	
	M1		M2		M1/M2	
	$PT1$	$PT2$	$PT1$	$PT2$	$PT1$	$PT2$
1	X_2	Z_1	Z_2	X_1	P	Q
2	X_1	Z_1	Z_1	T_1	Z_1=Z_3	X_1=X_3
3	P	Q	Q	T_2	X_2=X'	Z_2=Z'

ELLIPTIC CURVE SCALAR MULTIPLICATION

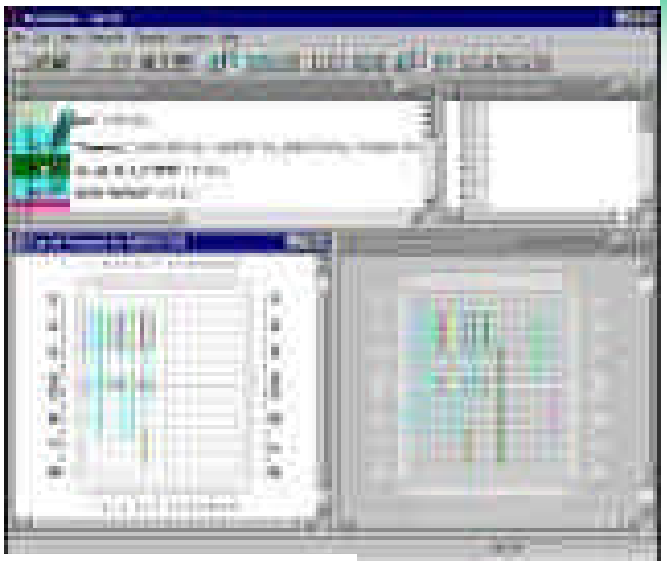
Results

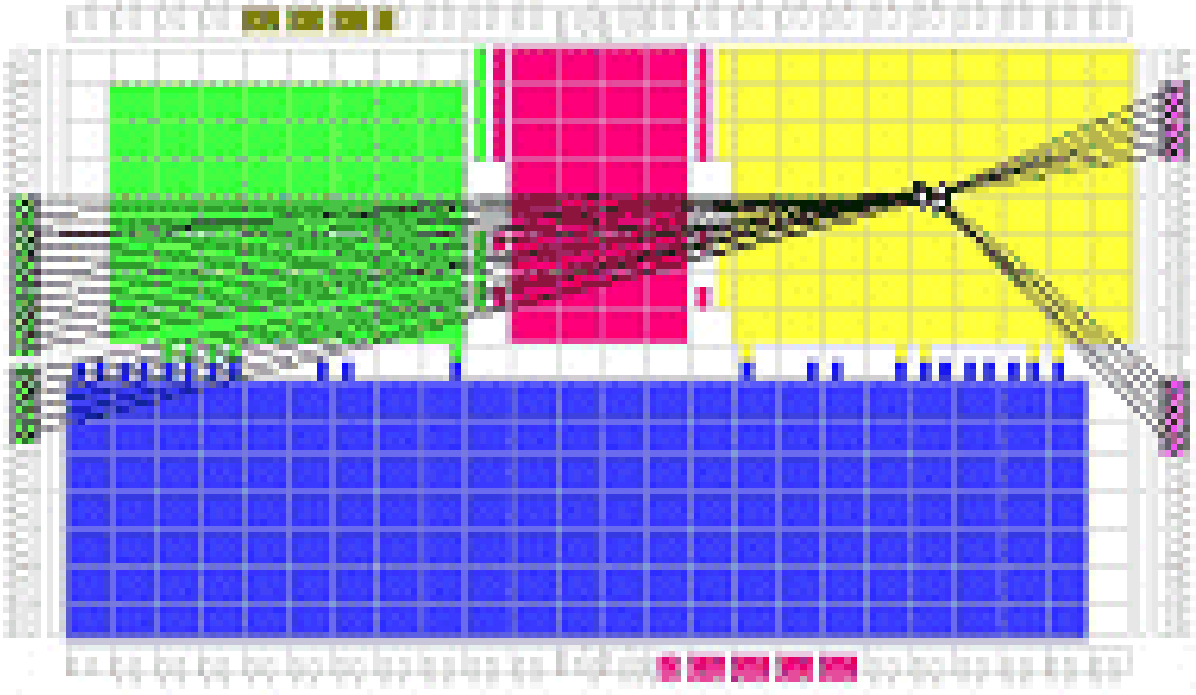
Target Device = XCV3200E
CLB Slices = 18314
BRAMs = 24
Timings = ? (waited = <100us

ADVANCED DESIGN TECHNIQUES

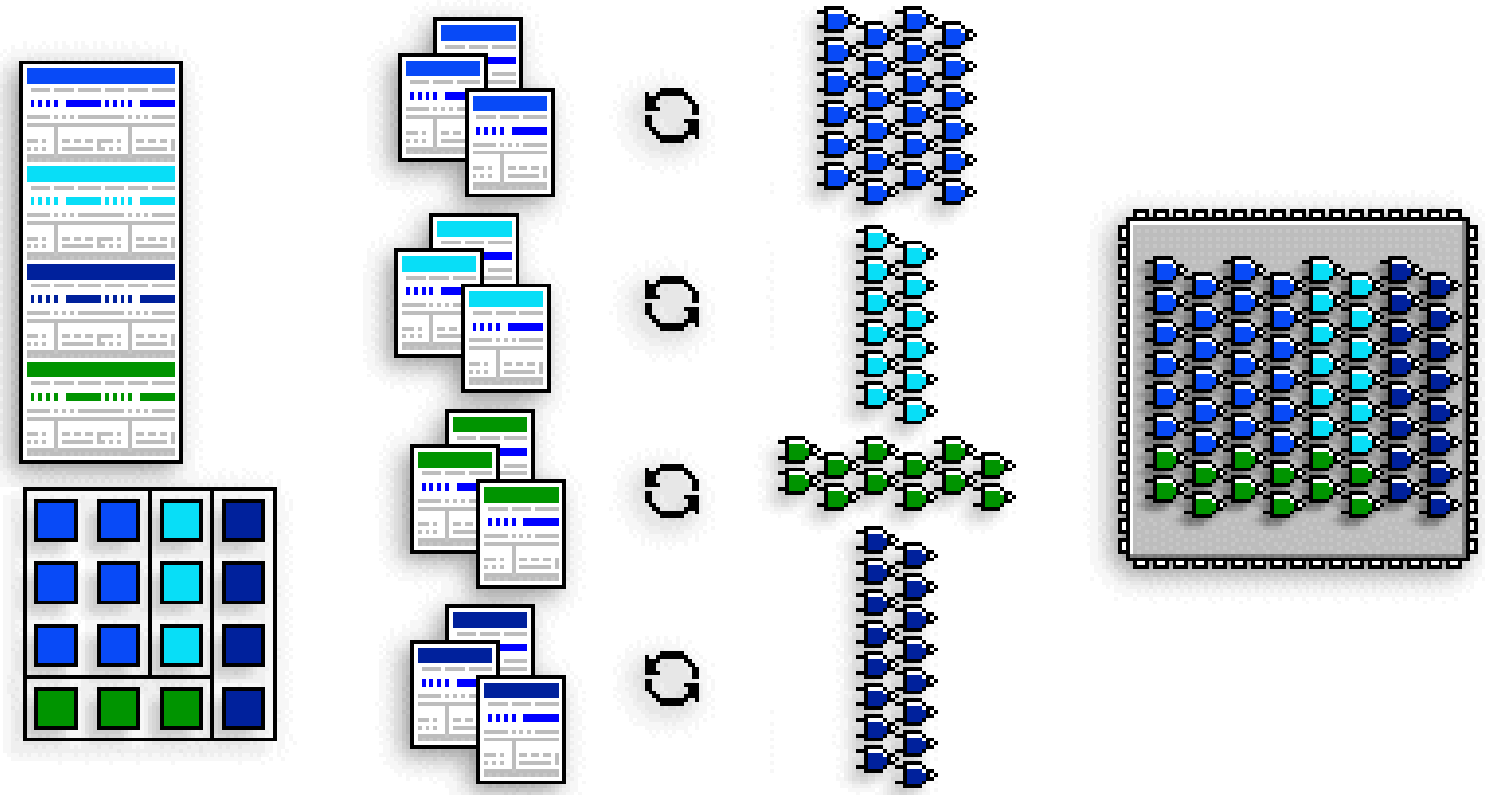
- HIGH-LEVEL FLOORPLANNING
- MODULAR DESIGN
- INCREMENTAL DESIGN
- HIGH LEVEL LANGUAGES
- PARTIAL RECONFIGURABILITY

HIGH-LEVEL FLOOR-PLANNING





THE MODULAR DESIGN FLOW



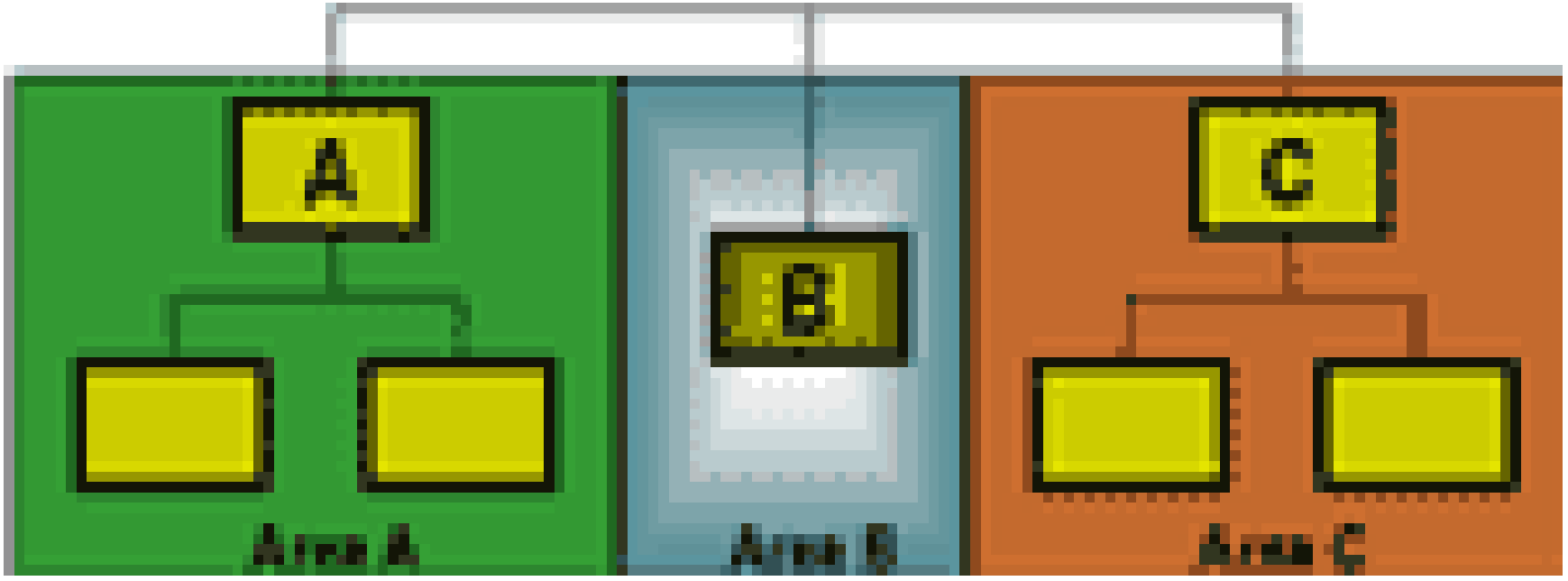
Initial Budgeting

Design of Each Module

Module Implementation

Global Place and Route

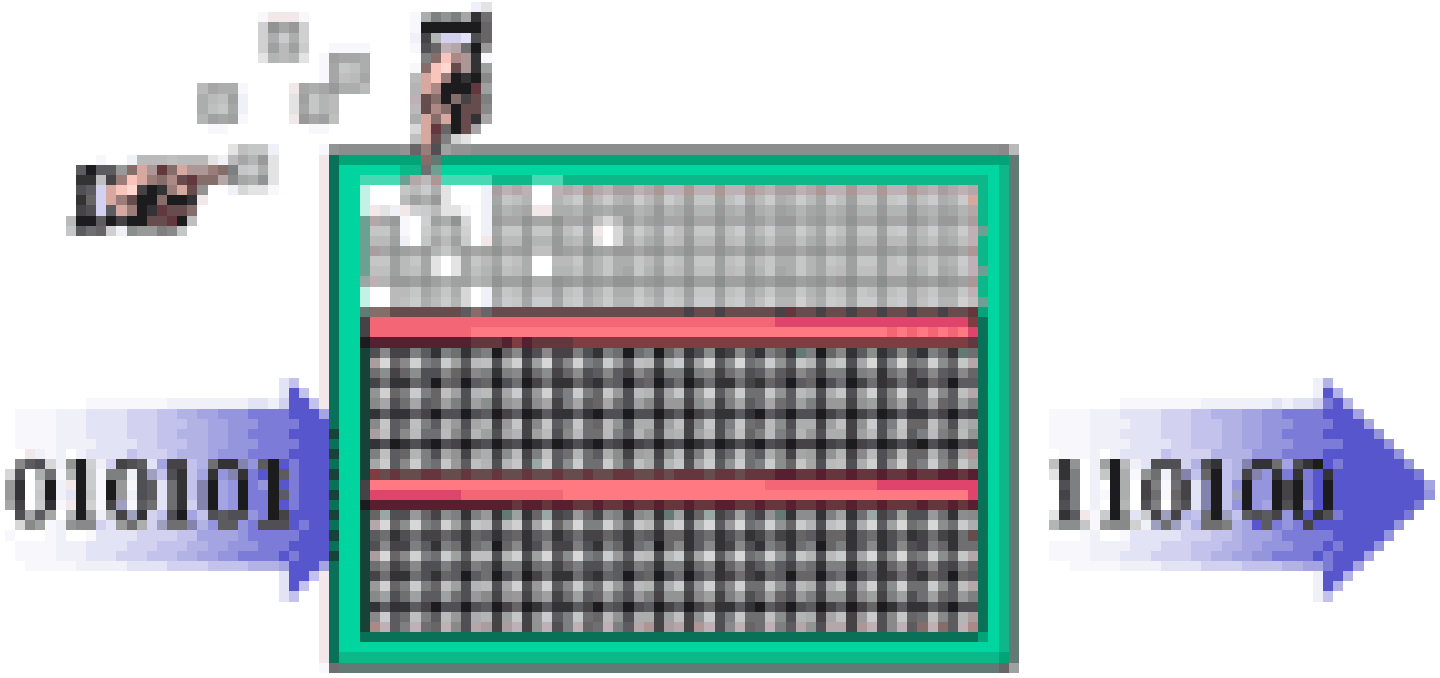
INCREMENTAL DESIGN



HIGH-LEVEL LANGUAGES

FORGE: java code to verilog RTL
SYNOPSYS: systemC
Celoxica: Handel-C and SystemC

PARTIAL RECONFIGURABILITY



CONCLUSIONS

To have an efficient implementation,

- **Good algorithmic techniques**
- **Good designing techniques**
- **Good implementing techniques**
- **Skill to use Desinging tool**