Chapter 1

Pentium IV adder

1.1 Introduction

The aim of this project is to create a model of Pentium IV adder in order to estimate its area, delay and power comsumption. For this purpose, a Matlab script is implemented to derive and plot results. Simulations for 8, 12, 16, 20, 24, 28 and 32 bit cases are reported in this work.
In this project it is supposed that any elementary logic gate necessary are available as external components. Each logic gate is assumed to be already characterized in terms of area, delay, input capacitances and leakage currents.
The approach for the design is structural: each block in the pentium IV is characterized like a black-box with its area, delay and power consumption. In this way the internal structure of any block ca be changed without modifying anything else and then the Matlab script can be run with the previous settings. This could be very useful to make future improvements for more complex simulations and structural designs.

1.2 Library logic gates

The logic gates used in the Pentium IV adder are NAND2, NOR2, INV, XOR2 and the transmission gate (called for brevity "tgate"). In the table 1.1 there are the gates with their parameters used.

GATE TYPE	AREA	INPUT CAPACINTANCES	DELAY	AVERAGE LEAKAGE CURRENT
NAND2	A_{nand2}	$C_{in_{nand2}}$	$\tau_{int_{nand2}} + \alpha_{nand2} \cdot C_L$	$I_{leak_{nand2}}$
NOR2	A_{nor2}	$C_{in_{nor2}}$	$\tau_{int_{nor2}} + \alpha_{nor2} \cdot C_L$	$I_{leak_{nor2}}$
INV	A_{inv}	$C_{in_{inv}}$	$\tau_{int_{inv}} + \alpha_{inv} \cdot C_L$	$I_{leak_{inv}}$
XOR2	A_{xor2}	$C_{in_{xor2}}$	$\tau_{int_{xor2}} + \alpha_{xor2} \cdot C_L$	$I_{leak_{xor2}}$
TGATE	A_{tgate}	$C_{in1_{tgate}}, C_{in2_{tgate}}, C_{in3_{tgate}}$	$\tau_{int_{tgate}} + \alpha_{tgate} \cdot C_L$	$I_{leak_{tgate}}$

Table 1.1: library logic gates

A brief explanation of the parameters is given before the adder analysis. For instance, the NAND2 gate can be considered.
A_{nand2} is the area of the NAND2.
$I_{leak_{nand2}}$ is the average leakage current of the NAND2. The leakage current of a logic gate depends on the values of its inputs, therefore the average leakage current is a function of the input probabilities. For sake of semplicity, the average leakage current is computed by considering the probabilities of the inputs equal.
For the input capacitances and the delay of the NAND2 figure 1.1 can be considered.

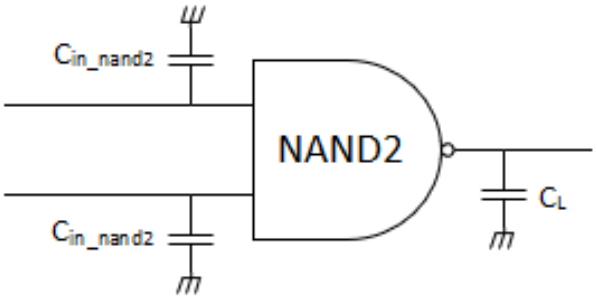

Figure 1.1: NAND2 gate

$C_{in_{nand2}}$ is the input capacitance in each input of the NAND2 gate.
The total delay of the NAND2 is $\tau_{int_{nand2}} + \alpha_{nand2} \cdot C_L$ where $\tau_{int_{nand2}}$ is the intrinsic delay of the NAND2 (the delay without load) while $\alpha_{nand2} \cdot C_L$ is the contribution that depends on the load C_L. The coefficient α_{nand2} depends on the sizes of the transistors inside the gate. The larger the transistors are, the smaller the delay is (α_{nand2} is smaller).
In the table, TGATE is the transmission gate. We will use the TGATE to make multiplexers. The TGATE is the only one in the table that has input

capacitances depending on which input is considered (differently respect to a normal CMOS gate).
It is assumed that all the parameters in table 1.1 are given externally. In other words, the design is a function of the parameters in the table.
Also the power supply will be a input parameter because it depends on the technology adopted.

1.3 Important assumptions in the design

To evaluate the dynamic power in the design the switching activity E_{sw} in all node of the circuit is assumed equal to one. This is a very pessimistic case that it will not happen in reality.
Evaluate the switching activity in each node is a very complex work. To do it the statistic of the inputs in our pentium IV should be known. Assuming $E_{sw} = 1$ the dynamic power is overestimated. For instance, the real value could be also one order of magnitude smaller. This means that the value of the dynamic power can be considered as an upper bound.
The working frequency that used for the dynamic power is the inverse of the critical path of the pentium IV (obviously the clock to output delay of a flip-flop, the setup time and the skew should be considered but this is also a simple estimation for power).
The leakage current in table 1.1 is in the case that the input probabilities for each gate are the same as already said. Obviously this is not true in reality, therefore the value obtained for the static power will be not exactly. However the order of magnitude is expected reasonably correct. This simplification is done because it is too complex to obtain the real statistic of the inputs of each gate: several simulation are to be carried in order to obtain the switching activity of each node.
To compute the delay of a path, all the delay of all gates in that path are summed. This means that, in table 1.1, the 50% delay for each gate is considered.

1.4 Pentium IV adder design

The 32-bit Pentium IV adder is constituted by two blocks as reported in figure 1.2. The carry-merge sparse tree computes the carry bit in position 4, 8, 12, 16, 20, 24, 28 and 32. The carry-select sum generator computes the sum bit.

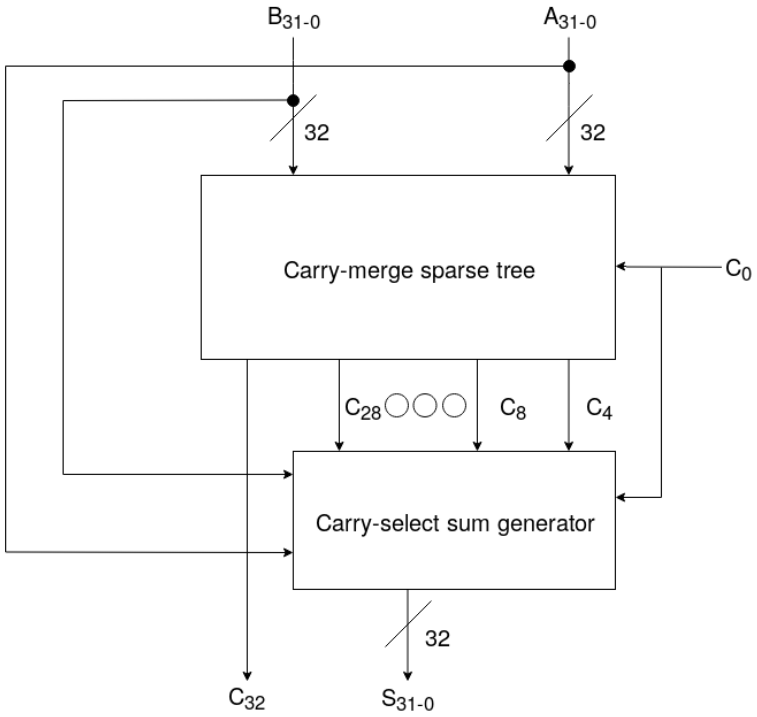

Figure 1.2: Pentium IV adder

In the following section we will see in detail these two blocks. A bottom-up approach is used to complete this design.

1.4.1 Carry-select sum generator design

In this subsection the carry-select sum generator is designed.

Full adder

In figure 1.3 it is shown the symbol of the full adder while in figure 1.4 its internal circuit.

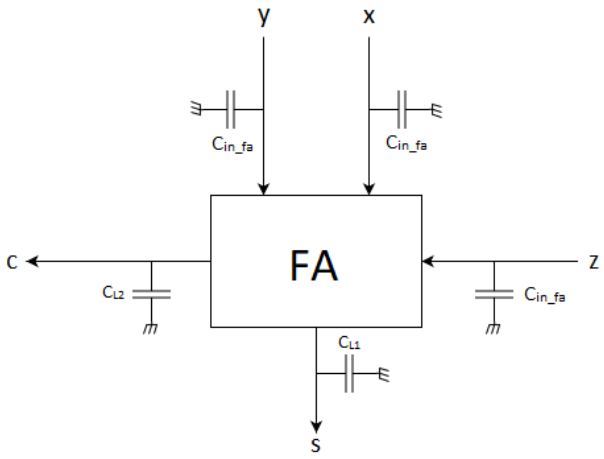

Figure 1.3: symbol of full adder

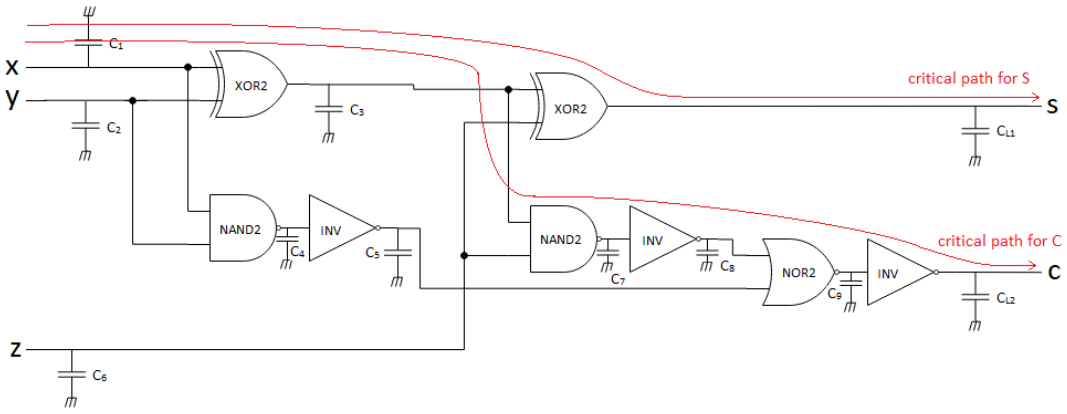

Figure 1.4: internal full adder circuit

The full adder area is given by:

$$A_{FA} = 2A_{xor2} + 2A_{nand2} + A_{nor2} + 3A_{inv} \tag{1.1}$$

The leakage current is:

$$I_{leak_{FA}} = 2I_{leak_{xor2}} + 2I_{leak_{nand2}} + I_{leak_{nor2}} + 3I_{leak_{inv}} \tag{1.2}$$

The static power is:

$$P_{stat_{FA}} = V_{DD} \cdot I_{leak_{FA}} \tag{1.3}$$

For the dynamic power, as we have already said, we assume that the switching activity E_{sw} of all the nodes in the circuit is equal to one. Therefore we can

write:

$$P_{dyn_{FA}} = V_{DD}^2 f_{CLK} E_{sw}(C_3 + C_4 + C_5 + C_7 + C_8 + C_9) \tag{1.4}$$

NOTE: for the dynamic power only the internal nodes are considered, which is that the input and output ones are not taken into account. The contribution of the input and output nodes will be considered when the FA-block will be used to create others blocks.
Looking at figure 1.4 we see that:

$$C_1 = C_2 = C_3 = C_6 = C_{in_{xor2}} + C_{in_{nand2}} \tag{1.5}$$

$$C_4 = C_7 = C_9 = C_{in_{inv}} \tag{1.6}$$

$$C_5 = C_8 = C_{in_{nor2}} \tag{1.7}$$

Thus we obtain:

$$P_{dyn_{FA}} = V_{DD}^2 f_{CLK} E_{sw}(C_{in_{xor2}} + C_{in_{nand2}} + 3C_{in_{inv}} + 2C_{in_{nor2}}) \tag{1.8}$$

The intrinsic delay (the delay without load) of the output S is:

$$\tau_{int_{FA_S}} = \tau_{int_{xor2}} + \alpha_{xor2} C_3 + \tau_{int_{xor2}} = 2\tau_{int_{xor2}} + \alpha_{xor2}(C_{in_{xor2}} + C_{in_{nand2}}) \tag{1.9}$$

The α parameter is:

$$\alpha_{FA_S} = \alpha_{xor2} \tag{1.10}$$

Thus the total delay of S (considering the load) is $\tau_{int_{FA_S}} + \alpha_{FA_S} C_{L1}$.

Analogously for the output C it is obtained:

$$\tau_{int_{FA_C}} = \tau_{int_{xor2}} + \alpha_{xor2} C_3 + \tau_{int_{nand2}} + \alpha_{nand2} C_7 + \tau_{int_{inv}} + \alpha_{inv} C_8 + \tau_{int_{nor2}} + \alpha_{nor2} C_9 + \tau_{int_{inv}} \tag{1.11}$$

$$\alpha_{FA_C} = \alpha_{inv} \tag{1.12}$$

The delay of C considering the load will be $\tau_{int_{FA_C}} + \alpha_{FA_C} C_{L2}$.

The input capacitances of the full adder are:

$$C_{in_x} = C_{in_y} = C_{in_z} = C_{in_{FA}} = C_{in_{xor2}} + C_{in_{nand2}} \tag{1.13}$$

4-bit ripple carry adder

In figure 1.5 it is shown the symbol of the 4-bit ripple carry adder while in figure 1.6 its internal circuit.

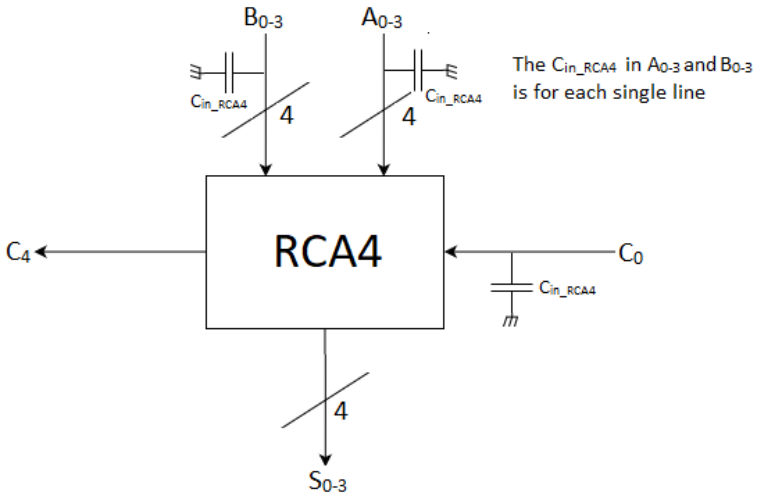

Figure 1.5: symbol of 4-bit ripple carry adder

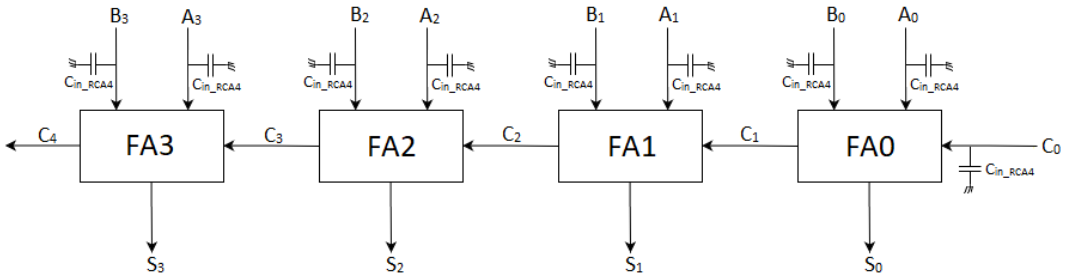

Figure 1.6: internal circuit of 4-bit ripple carry adder

The area of 4-bit ripple carry adder is given by:

$$A_{RCA4} = 4A_{FA} \tag{1.14}$$

The leakage current is:

$$I_{leak_{RCA4}} = 4I_{leak_{FA}} \tag{1.15}$$

The static power is:

$$P_{stat_{RCA4}} = V_{DD} \cdot I_{leak_{RCA4}} \tag{1.16}$$

As the same as for the full adder, to compute the dynamic power, only the internal nodes are considered:

$$P_{dyn_{RCA4}} = V_{DD}^2 f_{CLK} E_{sw}(3C_{in_{FA}}) + 4P_{dyn_{FA}} \tag{1.17}$$

NOTE: the term $4P_{dyn_{FA}}$ is for the internal nodes of the FAs.

To compute the delay, it can be observed that the delay to generate the 4 sum bits (S_0, S_1, S_2 and S_3) is the main concern. This is why the output carry (C_4) will be generated by the tree. Thus the delay of RCA4 to be considered is the delay to generate S_3. The intrinsic delay (that without load) is:

$$\tau_{int_{RCA4}} = 3\tau_{int_{FA_C}} + \alpha_{FA_C}(3C_{in_{FA}}) + \tau_{int_{FA_S}} \tag{1.18}$$

For the alpha coefficient it is derived that:

$$\alpha_{RCA4} = \alpha_{FA_S} \tag{1.19}$$

The input capacitances for the carry input (C_0) and for each line of A_{0-3} and B_{0-3} is the same and equal to:

$$C_{in_{RCA4}} = C_{in_{FA}} \tag{1.20}$$

Transmission gate

The transmission gate will be used to make multiplexers. Using transmission gate to implement MUXs, instead of normal CMOS gates, is a good solution in terms of area, delay and power consuption.
In figure 1.7 it is shown the symbol of the transmission gate while in figure 1.8 its internal circuit.

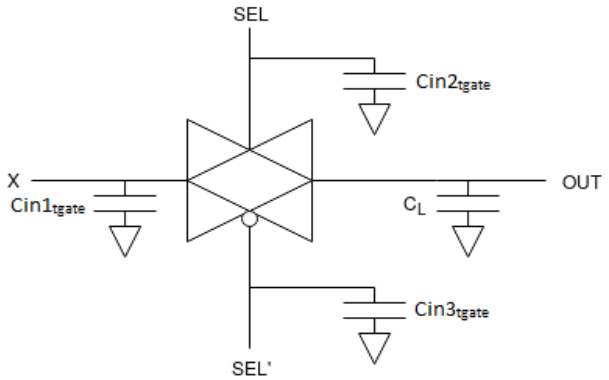

Figure 1.7: symbol of the transmission gate

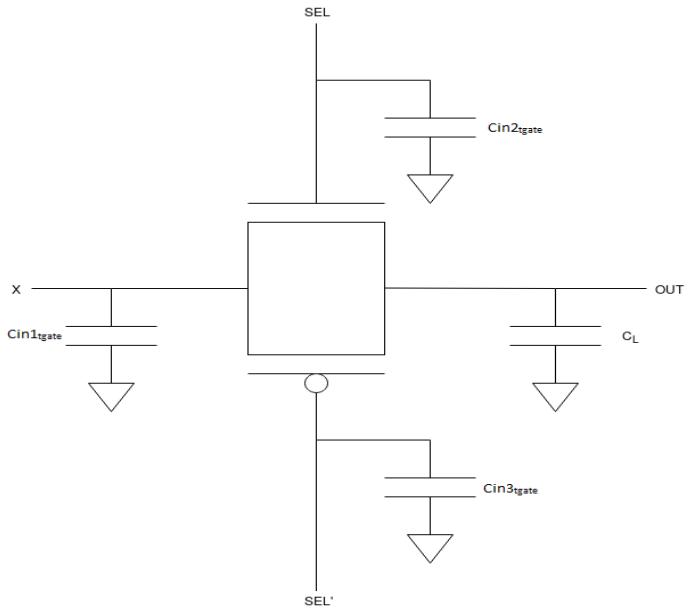

Figure 1.8: internal circuit of the transmission gate

From figure 1.8 it can be seen that the input capacitances of the transmission gate $C_{in1_{tgate}}$, $C_{in2_{tgate}}$ and $C_{in3_{tgate}}$ can be different. As said previously their values are given externally.
Also the area A_{tgate} and the leakage current $I_{leak_{tgate}}$ are given.
The delay of the transmission gate is the time required for the signal at the input X to reach the output node (OUT) when the two transistors become ON. This delay is equal to $\tau_{int_{tgate}} + \alpha_{tgate} \cdot C_L$ where $\tau_{int_{tgate}}$ and α_{tgate} are given.

Multiplexer 2:1

From the transmission gate described before, a 2:1 way multiplexer (mux) can be designed and characterized. Figure 1.9 reports the designed mux.

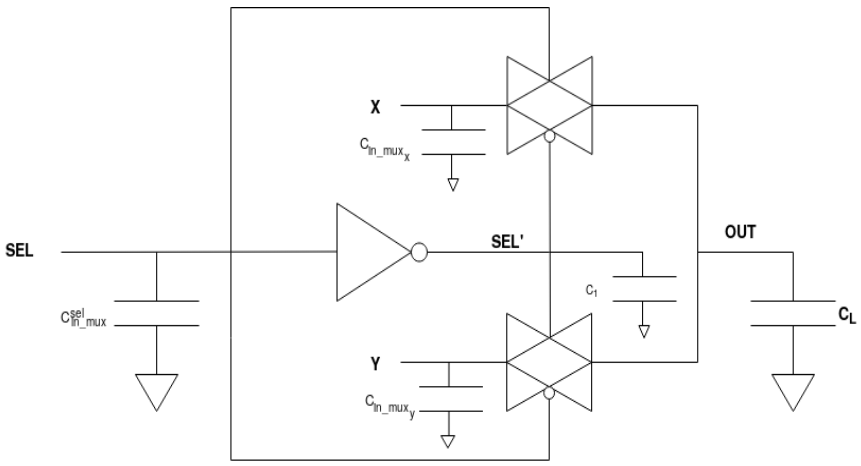

Figure 1.9: 2:1 way multiplexer implementation with transmission gates

The black-box model used for the multiplexer is the one in figure 1.10.

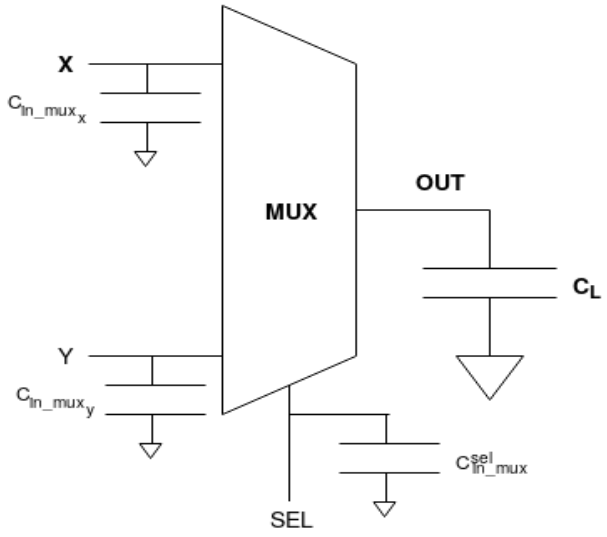

Figure 1.10: multiplexer 2:1 symbol

The multiplexer area is given by:

$$A_{mux} = A_{inv} + 2A_{tgate} \tag{1.21}$$

The leakage current is given by:

$$I_{leak_{mux}} = I_{leak_{inv}} + 2I_{leak_{tgate}} \tag{1.22}$$

The static power is

$$P_{stat_{mux}} = V_{DD} I_{leak_{mux}} \tag{1.23}$$

Looking at the figure 1.9, the dynamic power can be derived:

$$P_{dyn_{mux}} = V_{DD}^2 f_{CLK} E_{sw} C_1 \tag{1.24}$$

Since $C_1 = C_{in2_{tgate}} + C_{in3_{tgate}}$ we obtain:

$$P_{dyn_{mux}} = V_{DD}^2 f_{CLK} E_{sw} (C_{in2_{tgate}} + C_{in3_{tgate}}) \tag{1.25}$$

The intrinsic delay (the delay without load) for the mux is:

$$\tau_{int_{mux}} = \tau_{int_{inv}} + \alpha_{inv} C_1 + \tau_{int_{tgate}} \tag{1.26}$$

Replacing C_1 we obtain:

$$\tau_{int_{mux}} = \tau_{int_{inv}} + \alpha_{inv} (C_{in2_{tgate}} + C_{in3_{tgate}}) + \tau_{int_{tgate}} \tag{1.27}$$

The α coefficient is:

$$\alpha_{mux} = \alpha_{tgate} \tag{1.28}$$

Thus the overall delay is $\tau_{int_{mux}} + \alpha_{mux} C_L$. The input capacitance seen from the inputs X or Y is given by:

$$C_{in_{mux}}^{input} = C_{in_{mux_x}} = C_{in_{mux_y}} = C_{in1_{tgate}} \tag{1.29}$$

while the input capacitance seen from the select input is:

$$C_{in_{mux}}^{sel} = C_{in_{inv}} + C_{in2_{tgate}} + C_{in3_{tgate}} \tag{1.30}$$

Multiplexer 2:1 for M-length signal vectors

In the Pentium IV adder design is useful a compact rappresentation of a selector which takes two vectors with a length equal to M and takes one of them to the output. Figure 1.11 shows the implementation of such mux (called for brevity M-mux), while figure 1.12 reports the black-box symbol.

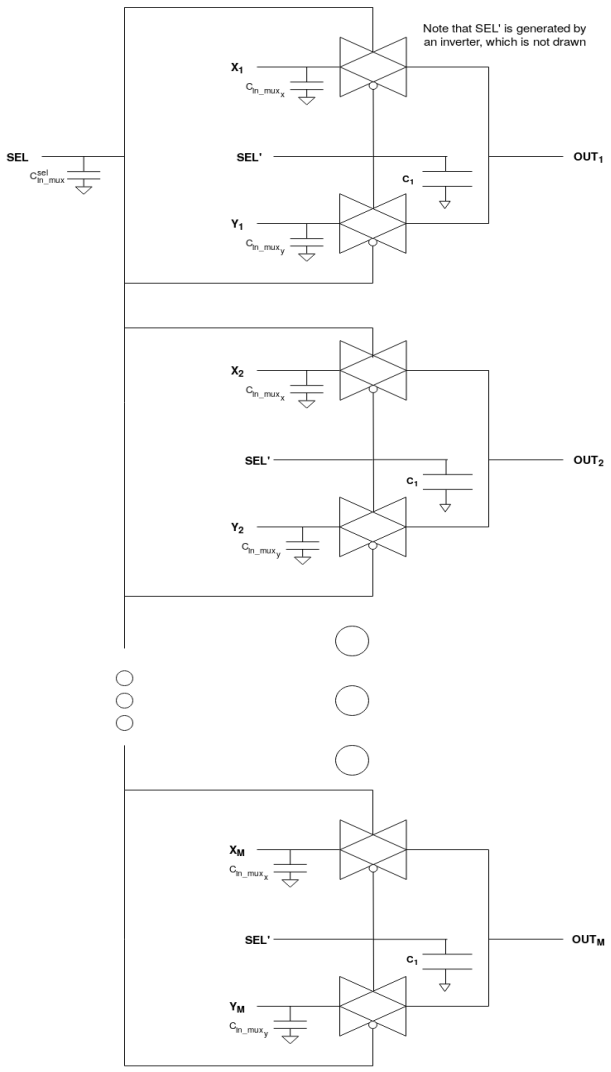

Figure 1.11: mux 2:1 with M-bit inputs

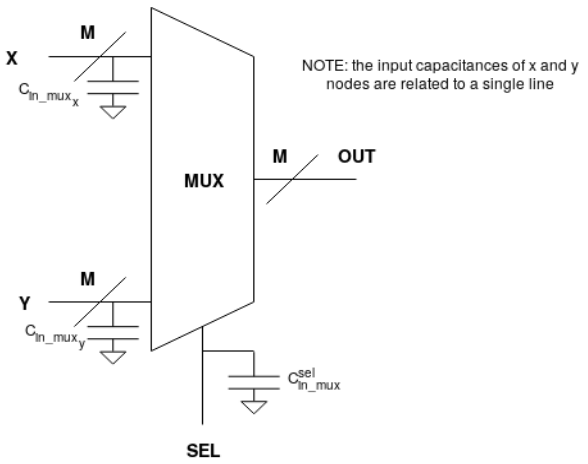

Figure 1.12: symbol of mux 2:1 with M-bit inputs

Since in figure 1.11 the SEL' signal is generated by an inverter which is not drawn, the area of the M-mux is:

$$A_{M-mux} = A_{inv} + 2MA_{tgate} \tag{1.31}$$

The leakage current is given by:

$$I_{leak_{M-mux}} = I_{leak_{inv}} + 2MI_{leak_{tgate}} \tag{1.32}$$

The static power is given by:

$$P_{stat_{M-mux}} = V_{DD} I_{leak_{M-mux}} \tag{1.33}$$

The dynamic power is given by:

$$P_{dyn_{M-mux}} = V_{DD}^2 f_{CLK} E_{sw} \cdot M(C_{in2_{tgate}} + C_{in3_{tgate}}) \tag{1.34}$$

The intrinsic delay is:

$$\tau_{int_{M-mux}} = \tau_{int_{inv}} + \alpha_{inv} M(C_{in2_{tgate}} + C_{in3_{tgate}}) + \tau_{int_{tgate}} \tag{1.35}$$

The α coefficient for the M-mux is:

$$\alpha_{M-mux} = \alpha_{tgate} \tag{1.36}$$

The input capacitances for each line of the x and y inputs are:

$$C_{in_{M-mux}}^{input} = C_{in_{mux_x}} = C_{in_{mux_y}} = C_{in1_{tgate}} \tag{1.37}$$

while the input capacitance seen from the select input is:

$$C_{in_{M-mux}}^{sel} = C_{in_{inv}} + M(C_{in2_{tgate}} + C_{in3_{tgate}}) \tag{1.38}$$

Carry-select block

The schematic of the carry-select block and its corresponding black box are reported in figure 1.13.

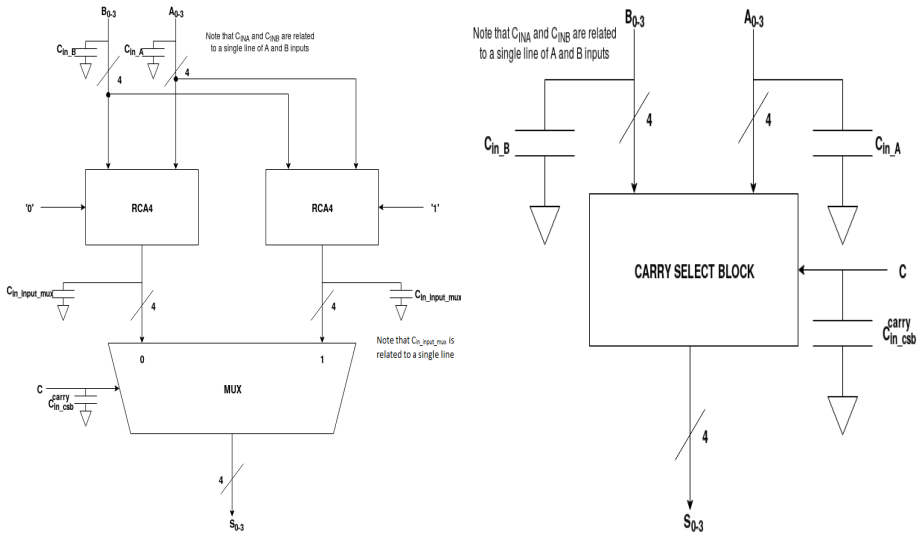

Figure 1.13: carry-select block schematic view and block symbol

The area of the carry-select block is given by:

$$A_{csb} = 2A_{RCA4} + A_{4-mux} \tag{1.39}$$

The leakage current is:

$$I_{leak_{csb}} = 2I_{leak_{RCA4}} + I_{leak_{4-mux}} \tag{1.40}$$

The static power is given by:

$$P_{stat_{csb}} = V_{DD} I_{leak_{csb}} \tag{1.41}$$

The dynamic power is given by:

$$P_{dyn_{csb}} = V_{DD}^2 f_{CLK} E_{sw} (8C_{in_{4-mux}}^{input}) + 2P_{dyn_{RCA4}} + P_{dyn_{4-mux}} \tag{1.42}$$

Assuming that all the inputs A_{0-3}, B_{0-3} and C are stable, the delay is that of the path from A_{0-3} and B_{0-3} to S_{0-3}. Therefore the intrinsic delay (the delay without load) is:

$$\tau_{int_{csb}} = \tau_{int_{RCA4}} + \alpha_{RCA4} C_{in_{4-mux}}^{input} + \tau_{int_{4-mux}} \tag{1.43}$$

The α coefficient of the carry-select block is:

$$\alpha_{csb} = \alpha_{4-mux} \tag{1.44}$$

The input capacitance of a single input line (A_{0-3} or B_{0-3}) is:

$$C_{in_{csb}}^{input} = C_{in_B} = C_{in_A} = 2C_{in_{RCA4}} \tag{1.45}$$

The carry input capacitance of the carry-select block is

$$C_{in_{csb}}^{carry} = C_{in_{4-mux}}^{sel} \tag{1.46}$$

Carry-select sum generator

Carry-select sum generator (cssg) schematic is reported in figure 1.14 and its black box symbol in figure 1.15. In these figures the number of bits is set to 32.

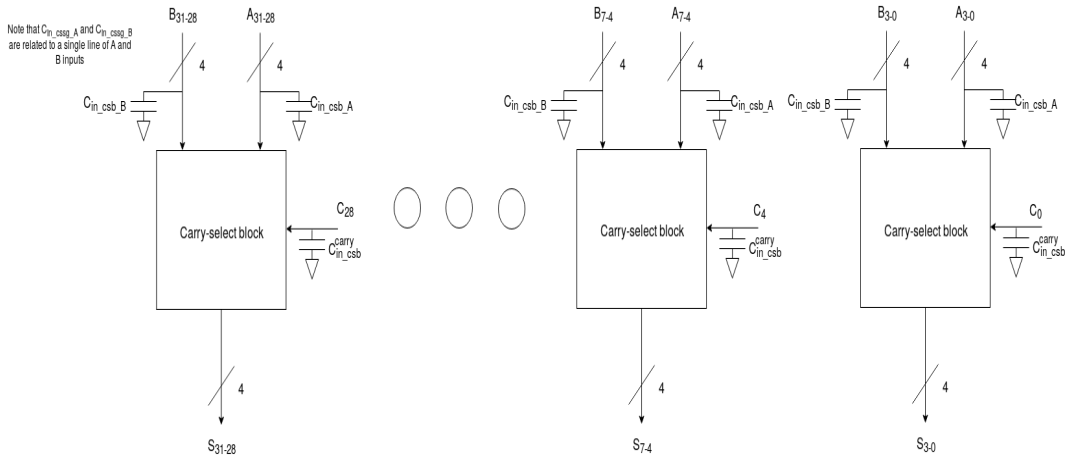

Figure 1.14: Carry-select sum generator schematic

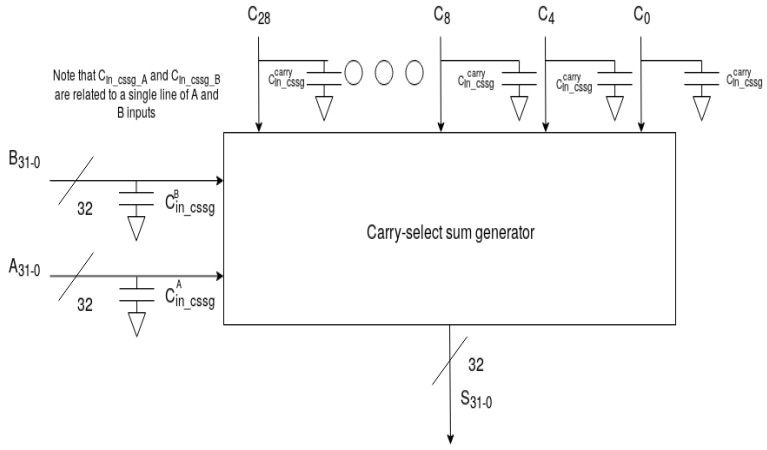

Figure 1.15: Carry-select sum generator black-box

Introducing N, which is the number of bits of the adder, the area can be derived:

$$A_{cssg} = \frac{N}{4} A_{csb} \tag{1.47}$$

Note that the values of N considered are 8, 12, 16, 20, 24, 28 and 32.
The leakage current is:

$$I_{leak_{cssg}} = \frac{N}{4} I_{leak_{csb}} \tag{1.48}$$

The static power is:

$$P_{stat_{cssg}} = V_{DD} I_{leak_{cssg}} \tag{1.49}$$

The dynamic power is:

$$P_{dyn_{cssg}} = \frac{N}{4} P_{dyn_{csb}} \tag{1.50}$$

The intrinsic delay (the delay without load) is:

$$\tau_{int_{cssg}} = \tau_{int_{csb}} \tag{1.51}$$

The α parameter is:

$$\alpha_{cssg} = \alpha_{csb} \tag{1.52}$$

The input capacitances for each line of the A and B inputs are:

$$C_{in_{cssg}}^{input} = C_{in_{cssg}}^{A} = C_{in_{cssg}}^{B} = C_{in_{csb}}^{input} \tag{1.53}$$

while the input capacitance seen from each carry input is:

$$C_{in_{cssg}}^{carry} = C_{in_{csb}}^{carry} \tag{1.54}$$

1.4.2 Carry-merge sparse tree

In this subsection the three blocks that constitute the carry-merge sparse tree are presented. They are the pre-computation block (pcb), the propagate-generate block (pg) and the G-block (gb). Using these blocks the carry-merge sparse tree is designed.

Pre-computation block

In figure 1.16 it is shown the symbol and the schematic view of the pre-computation block.

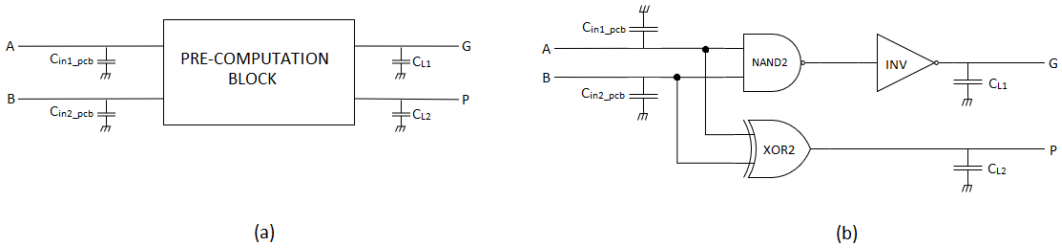

Figure 1.16: Pre-computation block. (a) Symbol. (b) Schematic view.

The area of the pre-computation block is given by:

$$A_{pcb} = A_{nand2} + A_{inv} + A_{xor2} \tag{1.55}$$

The leakage current is:

$$I_{leak_{pcb}} = I_{leak_{nand2}} + I_{leak_{inv}} + I_{leak_{xor2}} \tag{1.56}$$

The static power is:

$$P_{stat_{pcb}} = V_{DD} \cdot I_{leak_{pcb}} \tag{1.57}$$

The dynamic power is:

$$P_{dyn_{pcb}} = V_{DD}^2 f_{CLK} E_{sw} C_{in_{inv}} \tag{1.58}$$

The intrinsic delay (the delay without load) of the output G is:

$$\tau_{int_{pcb_G}} = \tau_{int_{nand2}} + \alpha_{nand2} C_{in_{inv}} + \tau_{int_{inv}} \tag{1.59}$$

Then the α parameter is:

$$\alpha_{pcb_G} = \alpha_{inv} \tag{1.60}$$

Thus the total delay of G (considering the load) is $\tau_{int_{pcb_G}} + \alpha_{pcb_G} C_{L1}$.

Analogously for the output P we have:

$$\tau_{int_{pcb_P}} = \tau_{int_{xor2}} \tag{1.61}$$

$$\alpha_{pcb_P} = \alpha_{xor2} \tag{1.62}$$

The delay of P considering the load will be $\tau_{int_{pcb_P}} + \alpha_{pcb_P} C_{L2}$.

The input capacitances of the pre-computation block are:

$$C_{in_{pcb}} = C_{in1_{pcb}} = C_{in2_{pcb}} = C_{in_{nand2}} + C_{in_{xor2}} \tag{1.63}$$

Propagate-generate block

Figure 1.17 shows the propagate-generate block (pg-block) schematic and figure 1.18 its black-box model.

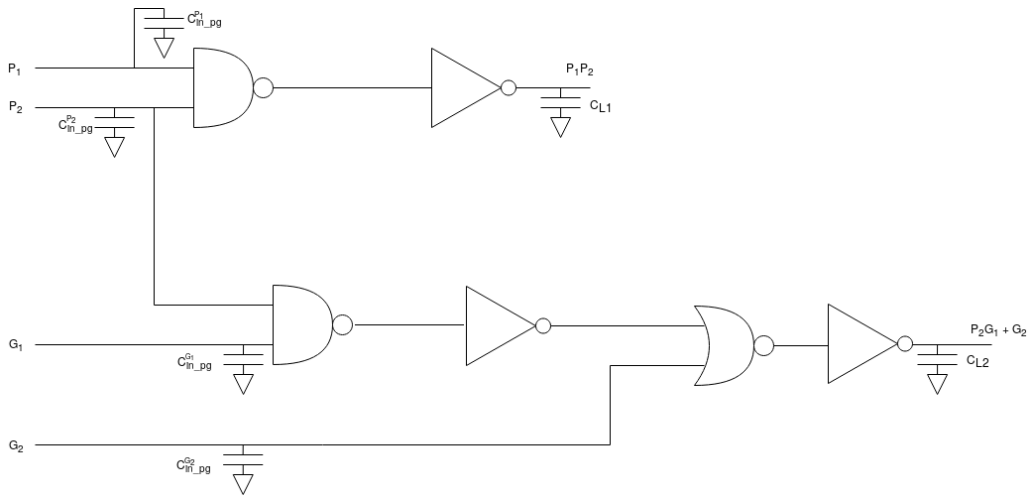

Figure 1.17: Schematic of propagate and generate carry block

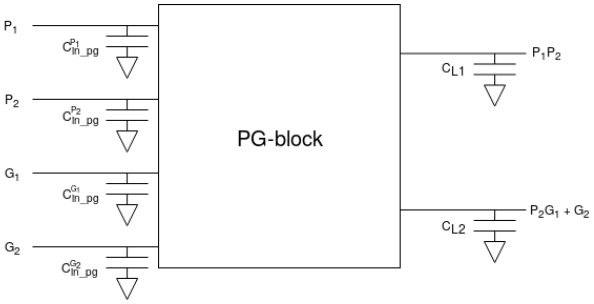

Figure 1.18: Symbol of propagate and generate carry block

From the schematic figure, the area can be evaluated as:

$$A_{pg} = 2A_{nand2} + A_{nor2} + 3A_{inv} \tag{1.64}$$

The leakage current is:

$$I_{leak_{pg}} = 2I_{leak_{nand2}} + I_{leak_{nor2}} + 3I_{leak_{inv}} \tag{1.65}$$

The static power is:

$$P_{stat_{pg}} = V_{DD} I_{leak_{pg}} \tag{1.66}$$

The dynamic power is:

$$P_{dyn_{pg}} = V_{DD}^2 f_{CLK} E_{sw} (3C_{in_{inv}} + C_{in_{nor2}}) \tag{1.67}$$

The intrinsic delay (delay without load) for the P_1P_2 output is:

$$\tau_{int_{pg_p}} = \tau_{int_{nand2}} + \alpha_{nand2} C_{in_{inv}} + \tau_{int_{inv}} \tag{1.68}$$

The α parameter is:

$$\alpha_{pg_p} = \alpha_{inv} \tag{1.69}$$

Thus the overall delay for the P_1P_2 output is $\tau_{int_{pg_p}} + \alpha_{pg_p} C_{L1}$.

The intrinsic delay for the $P_2G_1 + G_2$ output is:

$$\tau_{int_{pg_g}} = \tau_{int_{nand2}} + \alpha_{nand2} C_{in_{inv}} + \tau_{int_{inv}} + \alpha_{inv} C_{in_{nor2}} + \tau_{int_{nor2}} + \alpha_{nor2} C_{in_{inv}} + \tau_{int_{inv}} \tag{1.70}$$

The α parameter is:

$$\alpha_{pg_g} = \alpha_{inv} \tag{1.71}$$

The overall delay of this output is $\tau_{int_{pg_g}} + \alpha_{pg_g} C_{L2}$.

In order from P_1 to G_2, the input capacitances are:

$$C_{in_{pg}}^{P_1} = C_{in_{nand2}} \tag{1.72}$$

$$C_{in_{pg}}^{P_2} = 2C_{in_{nand2}} \tag{1.73}$$

$$C_{in_{pg}}^{G_1} = C_{in_{nand2}} \tag{1.74}$$

$$C_{in_{pg}}^{G_2} = C_{in_{nor2}} \tag{1.75}$$

Then the total input capacitance for this block is defined as:

$$C_{in_{pg}}^{tot} = C_{in_{pg}}^{P_1} + C_{in_{pg}}^{P_2} + C_{in_{pg}}^{G_1} + C_{in_{pg}}^{G_2} \tag{1.76}$$

G-block

This block is the same of the previous one but there is only the second output (G-output). In figure 1.19 it is shown the symbol and the schematic view of the G-block.

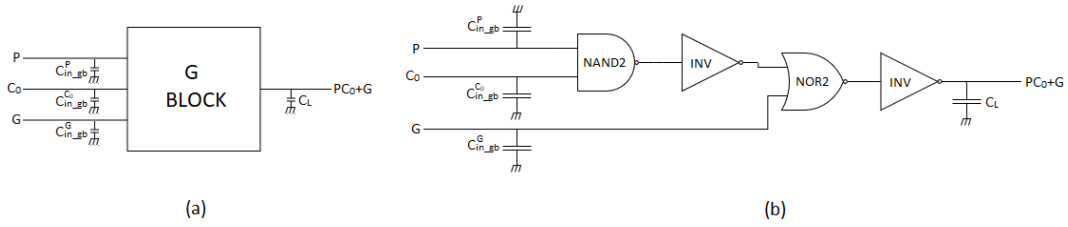

Figure 1.19: G-block. (a) Symbol. (b) Schematic view.

The area of the G-block is given by:

$$A_{gb} = A_{nand2} + A_{nor2} + 2A_{inv} \tag{1.77}$$

The leakage current is:

$$I_{leak_{gb}} = I_{leak_{nand2}} + I_{leak_{nor2}} + 2I_{leak_{inv}} \tag{1.78}$$

The static power is:

$$P_{stat_{gb}} = V_{DD} \cdot I_{leak_{gb}} \tag{1.79}$$

The dynamic power is:

$$P_{dyn_{gb}} = V_{DD}^2 f_{CLK} E_{sw} (2C_{in_{inv}} + C_{in_{nor2}}) \tag{1.80}$$

The intrinsic delay (the delay without load) is:

$$\tau_{int_{gb}} = \tau_{int_{nand2}} + \alpha_{nand2} C_{in_{inv}} + \tau_{int_{inv}} + \alpha_{inv} C_{in_{nor2}} + \tau_{int_{nor2}} + \alpha_{nor2} C_{in_{inv}} + \tau_{int_{inv}} \tag{1.81}$$

Then the α parameter is:

$$\alpha_{gb} = \alpha_{inv} \tag{1.82}$$

Thus the total delay (considering the load) is $\tau_{int_{gb}} + \alpha_{gb} C_L$.

The input capacitances are:

$$C_{in_{gb}}^{P} = C_{in_{nand2}} \tag{1.83}$$

$$C_{in_{gb}}^{C_0} = C_{in_{nand2}} \tag{1.84}$$

$$C_{in_{gb}}^{G} = C_{in_{nor2}} \tag{1.85}$$

Carry-merge sparse tree

In figure 1.20 and in figure 1.21 are respectively shown the black-box and the schematic view of the carry-merge sparse tree for a number of bit $N = 32$.

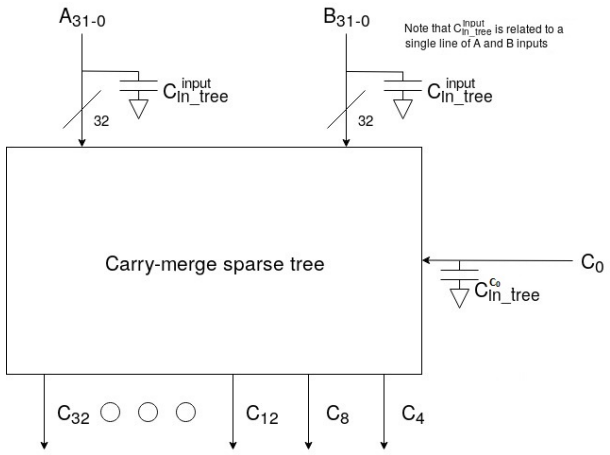

Figure 1.20: symbol of carry-merge sparse tree with 32 bit

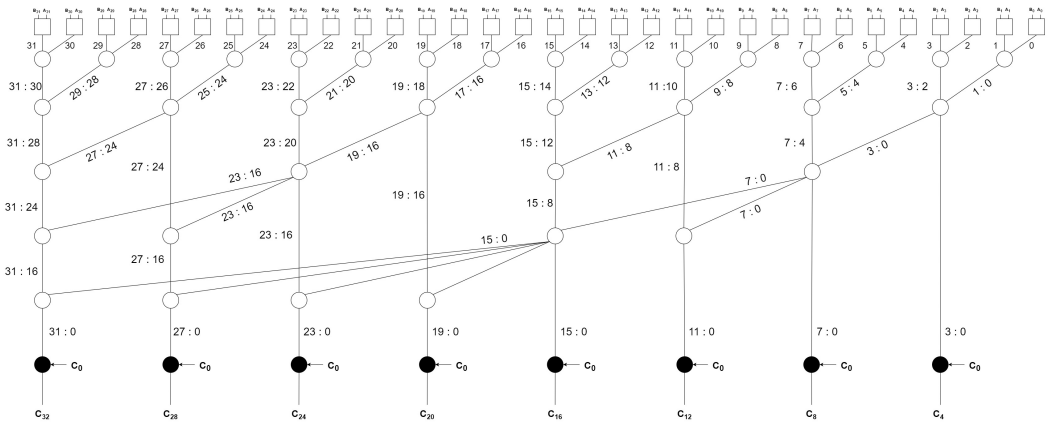

Figure 1.21: schematic view of the carry-merge sparse tree with 32 bit

In the schematic view the white squares are pre-computation blocks (pcb), the white circles are propagate-generate blocks (pg) and the black circles are G-blocks (gb). Note that the pcb and pg-blocks have two output but in the schematic view there is only one line for both of them. This is only to have a less complicated picture of the schematic view.
The outputs of this tree are the carry from C_4 to C_{32} with a step of 4.
Area, power consumption and delay of the tree are evaluated as a function of the number of bit N, in the case where N can be equal to 8, 12, 16, 20, 24, 28 or 32.
The area of the tree is equal to:

$$A_{tree} = N_{pcb}A_{pcb} + N_{pg}A_{pg} + N_{gb}A_{gb} \tag{1.86}$$

Where:

$$N_{pcb} = N \tag{1.87}$$

and

$$N_{gb} = \frac{N}{4} \tag{1.88}$$

N_{pg} cannot be written as a function of N in a simple way. Therefore the

number of pg-blocks for each case looking at the figure 1.21 can be counted.

$$N_{pg} = \begin{cases} 7 & N = 8 \\ 11 & N = 12 \\ 16 & N = 16 \\ 20 & N = 20 \\ 25 & N = 24 \\ 30 & N = 28 \\ 36 & N = 32 \end{cases} \tag{1.89}$$

The leakage current is given by:

$$I_{leak_{tree}} = N_{pcb} I_{leak_{pcb}} + N_{pg} I_{leak_{pg}} + N_{gb} I_{leak_{gb}} \tag{1.90}$$

The static power is:

$$P_{stat_{tree}} = V_{DD} \cdot I_{leak_{tree}} \tag{1.91}$$

The dynamic power is:

$$P_{dyn_{tree}} = V_{DD}^2 f_{CLK} E_{sw} \Big[N_{pg} C_{in_{pg}}^{tot} + N_{gb} \big(C_{in_{gb}}^{P} + C_{in_{gb}}^{G} \big) \Big] + N_{pcb} P_{dyn_{pcb}} + N_{pg} P_{pyn_{pg}} + N_{gb} P_{dyn_{gb}} \tag{1.92}$$

Without doing simplifications (for instance considering an upper bound for the delay instead of the real delay), there is not a simple way to evaluate the delay of the tree as a function of N. This is why when N changes, the capacitances in the nodes of the tree can change.
To evaluate the delay of the tree no simplifications are used. For this reason, each case of N is considered separately.

First of all, we note that the pcb-blocks have always the same load for each value of N. Thus we can compute the total delay of a pcb-block as:

$$\tau_{pcb_{tree}} = max\Big(\tau_{pcb_{tree}}^{G}, \tau_{pcb_{tree}}^{P} \Big) \tag{1.93}$$

where $\tau_{pcb_{tree}}^{G}$ is the delay related to the G output while $\tau_{pcb_{tree}}^{P}$ is related to the P one. Since the load of each the pcb-block is always one pg-block, it can be written:

$$\tau_{pcb_{tree}}^{G} = \tau_{int_{pcb_G}} + \alpha_{pcb_G} \cdot max\Big(C_{in_{pg}}^{G_1}, C_{in_{pg}}^{G_2} \Big) \tag{1.94}$$

$$\tau_{pcb_{tree}}^{P} = \tau_{int_{pcb_P}} + \alpha_{pcb_P} \cdot max\Big(C_{in_{pg}}^{P_1}, C_{in_{pg}}^{P_2} \Big) \tag{1.95}$$

In order to compute the delay of the tree for each value of N, it is useful computing the delay of the pg-block for different loads.
$\tau^G_{pg_{xy}}$ is the delay related to the output G of a pg-block when the load is a number x of pg-blocks and a number y of G-blocks. Similarly $\tau^P_{pg_{xy}}$ is the delay related to the output P of a pg-block when the load is a number x of pg-blocks and a number y of G-blocks.

Thus the delay of a pg-block with a load of 1 pg-block and 0 G-block is:

$$\tau_{pg_{10}} = max\left(\tau^G_{pg_{10}}, \tau^P_{pg_{10}}\right) \tag{1.96}$$

where:

$$\tau^G_{pg_{10}} = \tau_{int_{pg_g}} + \alpha_{pg_g} \cdot max\left(C^{G_1}_{in_{pg}}, C^{G_2}_{in_{pg}}\right) \tag{1.97}$$

$$\tau^P_{pg_{10}} = \tau_{int_{pg_p}} + \alpha_{pg_p} \cdot max\left(C^{P_1}_{in_{pg}}, C^{P_2}_{in_{pg}}\right) \tag{1.98}$$

The delay of a pg-block with a load of 1 pg-block and 1 G-block is:

$$\tau_{pg_{11}} = max\left(\tau^G_{pg_{11}}, \tau^P_{pg_{11}}\right) \tag{1.99}$$

where:

$$\tau^G_{pg_{11}} = \tau_{int_{pg_g}} + \alpha_{pg_g} \cdot \left[max\left(C^{G_1}_{in_{pg}}, C^{G_2}_{in_{pg}}\right) + C^G_{in_{gb}}\right] \tag{1.100}$$

$$\tau^P_{pg_{11}} = \tau_{int_{pg_p}} + \alpha_{pg_p} \cdot \left[max\left(C^{P_1}_{in_{pg}}, C^{P_2}_{in_{pg}}\right) + C^P_{in_{gb}}\right] \tag{1.101}$$

The delay of a pg-block with a load of 2 pg-block and 1 G-block is:

$$\tau_{pg_{21}} = max\left(\tau^G_{pg_{21}}, \tau^P_{pg_{21}}\right) \tag{1.102}$$

where:

$$\tau^G_{pg_{21}} = \tau_{int_{pg_g}} + \alpha_{pg_g} \cdot \left[2max\left(C^{G_1}_{in_{pg}}, C^{G_2}_{in_{pg}}\right) + C^G_{in_{gb}}\right] \tag{1.103}$$

$$\tau^P_{pg_{21}} = \tau_{int_{pg_p}} + \alpha_{pg_p} \cdot \left[2max\left(C^{P_1}_{in_{pg}}, C^{P_2}_{in_{pg}}\right) + C^P_{in_{gb}}\right] \tag{1.104}$$

The delay of a pg-block with a load of 4 pg-block and 1 G-block is:

$$\tau_{pg_{41}} = max\left(\tau^G_{pg_{41}}, \tau^P_{pg_{41}}\right) \tag{1.105}$$

where:

$$\tau^{G}_{pg_{41}} = \tau_{int_{pg_g}} + \alpha_{pg_g} \cdot \left[4max\left(C^{G_1}_{in_{pg}}, C^{G_2}_{in_{pg}}\right) + C^{G}_{in_{gb}}\right] \tag{1.106}$$

$$\tau^{P}_{pg_{41}} = \tau_{int_{pg_p}} + \alpha_{pg_p} \cdot \left[4max\left(C^{P_1}_{in_{pg}}, C^{P_2}_{in_{pg}}\right) + C^{P}_{in_{gb}}\right] \tag{1.107}$$

The delay of a pg-block with a load of 0 pg-block and 1 G-block is:

$$\tau_{pg_{01}} = max\left(\tau^{G}_{pg_{01}}, \tau^{P}_{pg_{01}}\right) \tag{1.108}$$

where:

$$\tau^{G}_{pg_{01}} = \tau_{int_{pg_g}} + \alpha_{pg_g} \cdot C^{G}_{in_{gb}} \tag{1.109}$$

$$\tau^{P}_{pg_{01}} = \tau_{int_{pg_p}} + \alpha_{pg_p} \cdot C^{P}_{in_{gb}} \tag{1.110}$$

Now the intrinsic delay (the delay without load) of the tree, for each value of N, can be evaluated. We will denote the intrinsic delay as $\tau^{N}_{int_{tree}}$ where N is the number of bit. Looking at the figure 1.21 we can write:

$$\tau^{8}_{int_{tree}} = \tau_{pcb_{tree}} + \tau_{pg_{10}} + \tau_{pg_{11}} + \tau_{pg_{01}} + \tau_{int_{gb}} \tag{1.111}$$

$$\tau^{12}_{int_{tree}} = \tau_{pcb_{tree}} + \tau_{pg_{10}} + \tau_{pg_{11}} + \tau_{pg_{21}} + \tau_{pg_{01}} + \tau_{int_{gb}} \tag{1.112}$$

$$\tau^{16}_{int_{tree}} = \tau^{12}_{int_{tree}} \tag{1.113}$$

$$\tau^{20}_{int_{tree}} = \tau_{pcb_{tree}} + \tau_{pg_{10}} + \tau_{pg_{11}} + \tau_{pg_{21}} + \tau_{pg_{41}} + \tau_{int_{gb}} \tag{1.114}$$

$$\tau^{24}_{int_{tree}} = \tau^{28}_{int_{tree}} = \tau^{32}_{int_{tree}} = \tau^{20}_{int_{tree}} \tag{1.115}$$

The α coefficient of the tree does not depend on N and it is equal to:

$$\alpha_{tree} = \alpha_{gb} \tag{1.116}$$

The input capacitance for each line of the A_{31-0} and B_{31-0} inputs is:

$$C^{input}_{in_{tree}} = C_{in_{pcb}} \tag{1.117}$$

while the input capacitance seen from the C_0:

$$C^{C_0}_{in_{tree}} = N_{gb} C^{C_0}_{in_{gb}} \tag{1.118}$$

1.4.3 Pentium IV adder

The overall architecture can be summarized into the schematic in figure 1.22. The black-box symbol is reported in figure 1.23.

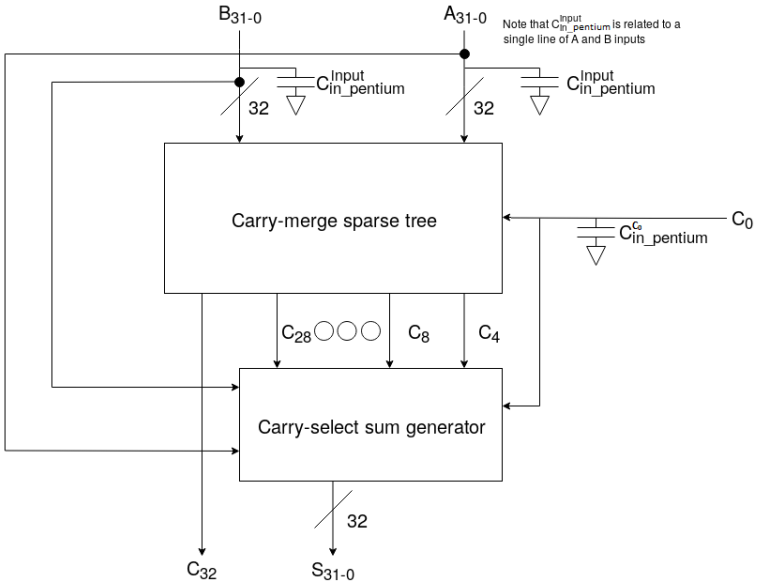

Figure 1.22: Schematic of the 32-bit Pentium IV adder

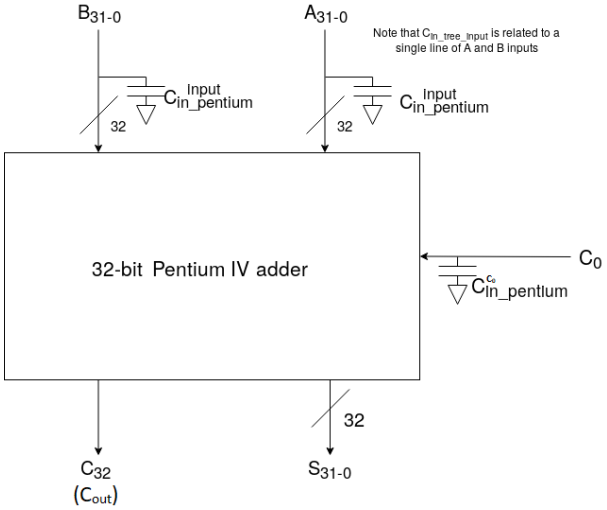

Figure 1.23: Pentium IV black-box view

The area can be evaluated as:

$$A_{pentium} = A_{tree} + A_{cssg} \tag{1.119}$$

The leakage current is:

$$I_{leak_{pentium}} = I_{leak_{tree}} + I_{leak_{cssg}} \tag{1.120}$$

The static power is:

$$P_{stat_{pentium}} = V_{DD} I_{leak_{pentium}} \tag{1.121}$$

The dynamic power is:

$$P_{dyn_{pentium}} = V_{DD}^2 f_{CLK} E_{sw} \left(\frac{N}{4} - 1 \right) C_{in_{cssg}}^{carry} + P_{dyn_{tree}} + P_{dyn_{cssg}} \tag{1.122}$$

The input signals A_{31-0} and B_{31-0} enter both in the tree and in the carry-select sum generator. Thus the tree and all RCA4s inside the carry-select sum generator work in parallel. This means that if the delay of the tree ($\tau_{int_{tree}}^{N} + \alpha_{tree} C_{in_{cssg}}^{carry}$) is greater than the delay of the RCA4 ($\tau_{int_{RCA4}} + \alpha_{RCA4} C_{in_{4-mux}}^{input}$), the delay of the Pentium without load will be $\tau_{int_{tree}}^{N} + \alpha_{tree} C_{in_{cssg}}^{carry} + \tau_{int_{4-mux}}$ where $C_{in_{cssg}}^{carry} = C_{in_{csb}}^{carry}$. On the contrary, if the delay of the tree is smaller than the delay of the RCA4 (usually this case does not happen if the number of bit is big), the delay of the the Pentium without load will be $\tau_{int_{cssg}} = \tau_{int_{csb}} = \tau_{int_{RCA4}} + \alpha_{RCA4} C_{in_{4-mux}}^{input} + \tau_{int_{4-mux}}$.

Therefore the delay of the Pentium without is:

$$\tau_{int_{pentium}} = max\left(\tau_{int_{tree}}^{N} + \alpha_{tree} C_{in_{cssg}}^{carry} + \tau_{int_{4-mux}}, \tau_{int_{cssg}} \right) \tag{1.123}$$

The α parameter is:

$$\alpha_{pentium} = \alpha_{cssg} \tag{1.124}$$

The input capacitance for each line of the A_{31-0} and B_{31-0} inputs is:

$$C_{in_{pentium}}^{input} = C_{in_{tree}}^{input} + C_{in_{cssg}}^{input} \tag{1.125}$$

while the input capacitance seen from the C_0:

$$C_{in_{pentium}}^{C_0} = C_{in_{tree}}^{C_0} + C_{in_{cssg}}^{carry} \tag{1.126}$$

Now the working frequency can be estimated. As already said in section 1.3, the inverse of the delay of the Pentium IV without load is used as working frequency.

$$f_{CLK} = \frac{1}{\tau_{int_{pentium}}} \tag{1.127}$$

1.5 Matlab implementation

All the formulas in this report are implemented in code with Matlab in order to evaluate area, power consumption and delay of the Pentium IV. Obviously the values of the input parameters in table 1.1 are set to see the results, but the characterization these logic gates is assumed already done by other works. For this reason reasonable values for the input parameters, considering the internal structure of the logic CMOS gates, are used.
In the following is reported the Matlab code.

```
clear all
close all

Vdd = 1; %V
Esw = 1;
N = [8,12,16,20,24,28,32];

% INPUT PARAMETERS

% NAND2
A_nand2 = 0.5; % (um)^2
Cin_nand2 = 4; %fF
tau_int_nand2 = 10.3; %ps
alpha_nand2 = 3; %KOhm
I_leak_nand2 = 200; %nA

% NOR2
A_nor2 = 0.625; % (um)^2
Cin_nor2 = 5; %fF
tau_int_nor2 = 10.7; %ps
alpha_nor2 = 3; %KOhm
I_leak_nor2 = 200; %nA

% INVERTER
A_inv = 0.1875; % (um)^2
Cin_inv = 3; %fF
tau_int_inv = 9; %ps
alpha_inv = 3; %KOhm
I_leak_inv = 100; %nA

% XOR2
A_xor2 = 1.875; % (um)^2
Cin_xor2 = 6; %fF
tau_int_xor2 = 12.6; %ps
alpha_xor2 = 3; %KOhm
I_leak_xor2 = 400; %nA

```

```
% TRANSMISSION GATE
A_tgate = 0.1875; % (um)^2
Cin1_tgate = 0.5; %fF
Cin2_tgate = 1; %fF
Cin3_tgate = 2; %fF
tau_int_tgate = 5.8; %ps
alpha_tgate = 3; %KOhm
I_leak_tgate = 50; %nA

% END INPUT PARAMETERS

% FULL ADDER
A_fa = 2*A_xor2 + 2*A_nand2 + A_nor2 + 3*A_inv;
Cin_fa = Cin_xor2 + Cin_nand2;
I_leak_fa = 2*I_leak_xor2 + 2*I_leak_nand2 + I_leak_nor2 + ...
    3*I_leak_inv;
P_stat_fa = Vdd*I_leak_fa;
E_dyn_fa = (Vdd^2)*Esw*(Cin_xor2 + Cin_nand2 + 3*Cin_inv + ...
    2*Cin_nor2);
tau_int_fa_sum = 2*tau_int_xor2 + ...
    alpha_xor2*(Cin_xor2+Cin_nand2);
alpha_fa_sum = alpha_xor2;
tau_int_fa_carry = tau_int_xor2 + ...
    alpha_xor2*(Cin_xor2+Cin_nand2) + tau_int_nand2 + ...
    alpha_nand2*Cin_inv + tau_int_inv + alpha_inv*Cin_nor2 ...
    + tau_int_nor2 + alpha_nor2*Cin_inv + tau_int_inv;
alpha_fa_carry = alpha_inv;

% MULTIPLEXER PARALLELISM 4
M = 4; %parallelism of mux inputs
Cin_inputs_mux_par4 = Cin1_tgate; %for each line of the inputs
Cin_sel_mux_par4 = Cin_inv + M*(Cin2_tgate+Cin3_tgate);
A_mux_par4 = A_inv + 2*M*A_tgate;
P_stat_mux_par4 = Vdd*(I_leak_inv + 2*M*I_leak_tgate);
E_dyn_mux_par4 = (Vdd^2)*Esw*M*(Cin2_tgate+Cin3_tgate);
tau_int_mux_par4 = tau_int_inv + ...
    alpha_inv*M*(Cin2_tgate+Cin3_tgate) + tau_int_tgate;
alpha_mux_par4 = alpha_tgate;

% MULTIPLEXER PARALLELISM 33
M = 33; %parallelism of mux inputs
Cin_inputs_mux_par33 = Cin1_tgate; %for each line of the ...
    inputs
Cin_sel_mux_par33 = Cin_inv + M*(Cin2_tgate+Cin3_tgate);
A_mux_par33 = A_inv + 2*M*A_tgate;
P_stat_mux_par33 = Vdd*(I_leak_inv + 2*M*I_leak_tgate);
E_dyn_mux_par33 = (Vdd^2)*Esw*M*(Cin2_tgate+Cin3_tgate);
tau_int_mux_par33 = tau_int_inv + ...
    alpha_inv*M*(Cin2_tgate+Cin3_tgate) + tau_int_tgate;
```

```
78  alpha_mux_par33 = alpha_tgate;
79
80  % RIPPLE CARRY ADDER 4 BITS
81  Cin_rca4 = Cin_fa; % for the carry input and for each line ...
        of the two data inputs
82  A_rca4 = 4*A_fa;
83  P_stat_rca4 = 4*P_stat_fa;
84  E_dyn_rca4 = (Vdd^2)*Esw*(3*Cin_fa) + 4*E_dyn_fa;
85  tau_int_rca4 = 3*tau_int_fa_carry + ...
        3*alpha_fa_carry*Cin_fa + tau_int_fa_sum;
86  alpha_rca4 = alpha_fa_sum;
87
88  % CARRY-SELECT BLOCK
89  Cin_inputs_csb = 2*Cin_rca4; % for each line of the inputs
90  Cin_carry_csb = Cin_sel_mux_par4;
91  A_csb = 2*A_rca4 + A_mux_par4;
92  P_stat_csb = 2*P_stat_rca4 + P_stat_mux_par4;
93  E_dyn_csb = (Vdd^2)*Esw*8*Cin_inputs_mux_par4 + ...
        2*E_dyn_rca4 + E_dyn_mux_par4;
94  tau_int_csb = tau_int_rca4 + ...
        alpha_rca4*Cin_inputs_mux_par4 + tau_int_mux_par4;
95  alpha_csb = alpha_mux_par4;
96
97  % CARRY-SELECT SUM GENERATOR
98  Cin_data_inputs_cssg = Cin_inputs_csb; % for each line of ...
        the inputs
99  Cin_carry_cssg = Cin_carry_csb; % for each carry line
100 A_cssg = (N/4)*A_csb;
101 P_stat_cssg = (N/4)*P_stat_csb;
102 E_dyn_cssg = (N/4)*E_dyn_csb;
103 tau_int_cssg = tau_int_csb;
104 alpha_cssg = alpha_csb;
105
106 % PRE COMPUTATION BLOCK
107 Cin_pcb = Cin_nand2 + Cin_xor2;
108 A_pcb = A_nand2 + A_inv + A_xor2;
109 I_leak_pcb = I_leak_nand2 + I_leak_inv + I_leak_xor2;
110 P_stat_pcb = Vdd*I_leak_pcb;
111 E_dyn_pcb = (Vdd^2)*Esw*Cin_inv;
112 tau_int_pcb_g = tau_int_nand2 + alpha_nand2*Cin_inv + ...
        tau_int_inv; % related to g output
113 alpha_pcb_g = alpha_inv; % related to g output
114 tau_int_pcb_p = tau_int_xor2; % related to p output
115 alpha_pcb_p = alpha_xor2; % related to p output
116
117 % PROPAGATED AND GENERATED CARRY BLOCK
118 Cin_pg_p1 = Cin_nand2;
119 Cin_pg_p2 = 2*Cin_nand2;
120 Cin_pg_g1 = Cin_nand2;
```

```
121 Cin_pg_g2 = Cin_nor2;
122 Cin_pg_tot = Cin_pg_p1 + Cin_pg_p2 + Cin_pg_g1 + Cin_pg_g2;
123 A_pg = 2*A_nand2 + A_nor2 + 3*A_inv;
124 I_leak_pg = 2*I_leak_nand2 + I_leak_nor2 + 3*I_leak_inv;
125 P_stat_pg = Vdd*I_leak_pg;
126 E_dyn_pg = (Vdd^2)*Esw*(3*Cin_inv+Cin_nor2);
127 tau_int_pg_g = tau_int_nand2 + alpha_nand2*Cin_inv + ...
       tau_int_inv + alpha_inv*Cin_nor2 + tau_int_nor2 + ...
       alpha_nor2*Cin_inv + tau_int_inv; % related to g output
128 alpha_pg_g = alpha_inv; % related to g output
129 tau_int_pg_p = tau_int_nand2 + alpha_nand2*Cin_inv + ...
       tau_int_inv; % related to p output
130 alpha_pg_p = alpha_inv; % related to p output
131
132 % ONLY G BLOCK
133 Cin_gb_p = Cin_nand2;
134 Cin_gb_c0 = Cin_nand2;
135 Cin_gb_g = Cin_nor2;
136 A_gb = A_nand2 + A_nor2 + 2*A_inv;
137 I_leak_gb = I_leak_nand2 + I_leak_nor2 + 2*I_leak_inv;
138 P_stat_gb = Vdd*I_leak_gb;
139 E_dyn_gb = (Vdd^2)*Esw*(2*Cin_inv+Cin_nor2);
140 tau_int_gb = tau_int_nand2 + alpha_nand2*Cin_inv + ...
       tau_int_inv + alpha_inv*Cin_nor2 + tau_int_nor2 + ...
       alpha_nor2*Cin_inv + tau_int_inv;
141 alpha_gb = alpha_inv;
142
143 % TREE
144 N_of_pcb_block = N;
145 N_of_pg_block = [7,11,16,20,25,30,36];
146 N_of_gb_block = N/4;
147
148 Cin_data_inputs_tree = Cin_pcb; % for each line of the ...
       data inputs A and B
149 Cin_C0_tree = N_of_gb_block*Cin_gb_c0;
150
151 A_tree = N_of_pcb_block*A_pcb + N_of_pg_block*A_pg + ...
       N_of_gb_block*A_gb;
152 P_stat_tree = N_of_pcb_block*P_stat_pcb + ...
       N_of_pg_block*P_stat_pg + N_of_gb_block*P_stat_gb;
153 E_dyn_tree = (Vdd^2)*Esw*(N_of_pg_block*Cin_pg_tot + ...
       N_of_gb_block*(Cin_gb_p+Cin_gb_g)) + ...
       N_of_pcb_block*E_dyn_pcb + N_of_pg_block*E_dyn_pg + ...
       N_of_gb_block*E_dyn_gb;
154
155 % delay of a pcb block with one pg block as load
156 tau_pcb_tree_g = tau_int_pcb_g + ...
       alpha_pcb_g*max(Cin_pg_g1,Cin_pg_g2);
157 tau_pcb_tree_p = tau_int_pcb_p + ...
```

```
      alpha_pcb_p*max(Cin_pg_p1,Cin_pg_p2);
158 tau_pcb_tree = max(tau_pcb_tree_g,tau_pcb_tree_p);
159
160 % delay of a pg block with 1 pg block and 0 gb block as load
161 tau_pg_tree_1_0_g = tau_int_pg_g + ...
      alpha_pg_g*(1*max(Cin_pg_g1,Cin_pg_g2));
162 tau_pg_tree_1_0_p = tau_int_pg_p + ...
      alpha_pg_p*(1*max(Cin_pg_p1,Cin_pg_p2));
163 tau_pg_tree_1_0 = max(tau_pg_tree_1_0_g,tau_pg_tree_1_0_p);
164
165 % delay of a pg block with 1 pg block and 1 gb block as load
166 tau_pg_tree_1_1_g = tau_int_pg_g + ...
      alpha_pg_g*(1*max(Cin_pg_g1,Cin_pg_g2)+Cin_gb_g);
167 tau_pg_tree_1_1_p = tau_int_pg_p + ...
      alpha_pg_p*(1*max(Cin_pg_p1,Cin_pg_p2)+Cin_gb_p);
168 tau_pg_tree_1_1 = max(tau_pg_tree_1_1_g,tau_pg_tree_1_1_p);
169
170 % delay of a pg block with 2 pg block and 1 gb block as load
171 tau_pg_tree_2_1_g = tau_int_pg_g + ...
      alpha_pg_g*(2*max(Cin_pg_g1,Cin_pg_g2)+Cin_gb_g);
172 tau_pg_tree_2_1_p = tau_int_pg_p + ...
      alpha_pg_p*(2*max(Cin_pg_p1,Cin_pg_p2)+Cin_gb_p);
173 tau_pg_tree_2_1 = max(tau_pg_tree_2_1_g,tau_pg_tree_2_1_p);
174
175 % delay of a pg block with 4 pg block and 1 gb block as load
176 tau_pg_tree_4_1_g = tau_int_pg_g + ...
      alpha_pg_g*(4*max(Cin_pg_g1,Cin_pg_g2)+Cin_gb_g);
177 tau_pg_tree_4_1_p = tau_int_pg_p + ...
      alpha_pg_p*(4*max(Cin_pg_p1,Cin_pg_p2)+Cin_gb_p);
178 tau_pg_tree_4_1 = max(tau_pg_tree_4_1_g,tau_pg_tree_4_1_p);
179
180 % delay of a pg block with 0 pg block and 1 gb block as load
181 tau_pg_tree_0_1_g = tau_int_pg_g + alpha_pg_g*Cin_gb_g;
182 tau_pg_tree_0_1_p = tau_int_pg_p + alpha_pg_p*Cin_gb_p;
183 tau_pg_tree_0_1 = max(tau_pg_tree_0_1_g,tau_pg_tree_0_1_p);
184
185 % intrinsic delay of the tree
186 tau_int_tree(1) = tau_pcb_tree + tau_pg_tree_1_0 + ...
      tau_pg_tree_1_1 + tau_pg_tree_0_1 + tau_int_gb; % N=8;
187 tau_int_tree(2) = tau_pcb_tree + tau_pg_tree_1_0 + ...
      tau_pg_tree_1_1 + tau_pg_tree_2_1 + tau_pg_tree_0_1 + ...
      tau_int_gb; % N=12;
188 tau_int_tree(3) = tau_int_tree(2); % N=16
189 tau_int_tree(4) = tau_pcb_tree + tau_pg_tree_1_0 + ...
      tau_pg_tree_1_1 + tau_pg_tree_2_1 + tau_pg_tree_4_1 + ...
      tau_int_gb; % N=20;
190 tau_int_tree(5) = tau_int_tree(4); % N=24
191 tau_int_tree(6) = tau_int_tree(4); % N=28
192 tau_int_tree(7) = tau_int_tree(4); % N=32
```

```
193
194 % the alpha tree in independent of N
195 alpha_tree = alpha_gb;
196
197 % PENTIUM IV
198 Cin_inputs_pentium = Cin_data_inputs_tree + ...
       Cin_data_inputs_cssg; % for each line of the inputs
199 Cin_C0_pentium = Cin_C0_tree + Cin_carry_cssg;
200 A_pentium = A_tree + A_cssg;
201 P_stat_pentium = P_stat_tree + P_stat_cssg; % nW
202 E_dyn_pentium = (Vdd^2)*Esw*((N/4)-1)*Cin_carry_cssg + ...
       E_dyn_tree + E_dyn_cssg;
203 tau_int_pentium = max(tau_int_tree + ...
       alpha_tree*Cin_carry_cssg + tau_int_mux_par4, ...
       tau_int_cssg); % ps
204 alpha_pentium = alpha_cssg;
205
206 % MAXIMUM OPERATING FREQUENCY
207 fmax = 1./tau_int_pentium; % THz
208
209 % DYNAMIC POWER PENTIUM IV
210 P_dyn_pentium = E_dyn_pentium.*fmax; % mW
211
212 % PLOTS
213
214 figure(1)
215 hold on
216 grid on
217 plot(N,A_pentium,'.','MarkerSize' ,25);
218 title('Area Pentium IV as a function of N');
219 xlabel('N');
220 ylabel('A  (\mum^2)');
221
222 figure(2)
223 hold on
224 grid on
225 plot(N,P_stat_pentium./1000000,'.','MarkerSize' ,25);
226 title('Static power Pentium IV as a function of N');
227 xlabel('N');
228 ylabel('P_{stat}  (mW)');
229
230 figure(3)
231 hold on
232 grid on
233 plot(N,P_dyn_pentium,'.','MarkerSize' ,25);
234 title('Dynamic power Pentium IV as a function of N');
235 xlabel('N');
236 ylabel('P_{dyn}  (mW)');
237
```

```
figure(4)
hold on
grid on
plot(N,tau_int_pentium,'.','MarkerSize' ,25);
title('Delay without load Pentium IV as a function of N');
xlabel('N');
ylabel('Delay  (ps)');
```

1.6 Results

In this section are reported in graphs the results obtained for area, power and delay of the Pentium IV using 8, 12, 16, 20, 24, 28 and 32 bits parallelism.
As expected the area increases by increasing the number of bits (figure 1.24). The static power increases with the same trend of the area (figure 1.25). This is why each block in the Pentium IV has a leakage current, thus if the area increases also the static power increases.
If the area increases (N increases), it is expected also that the dynamic power increases as it can be seen in figure 1.26.
Figure 1.27 shows the delay without load of the Pentium IV. The figure could be strange but it is not. It can be explained remembering that for $N = 12$ and $N = 16$ the tree has the same delay (eq.1.113). Also for $N = 20$, $N = 24$, $N = 28$ and $N = 32$ the tree has the same delay (eq.1.115).

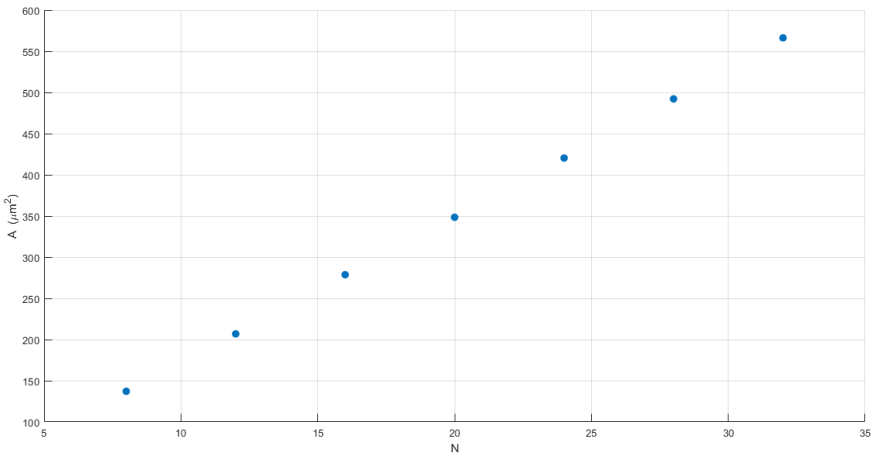

Figure 1.24: Area Pentium IV as a function of N

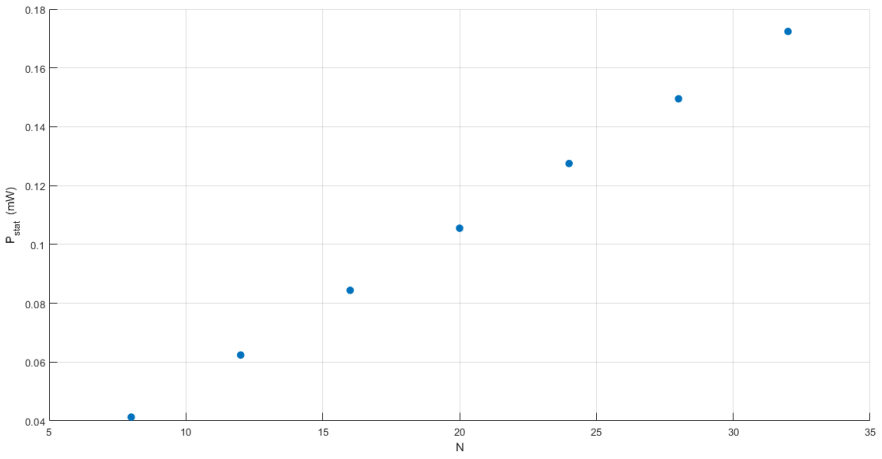

Figure 1.25: Static power Pentium IV as a function of N

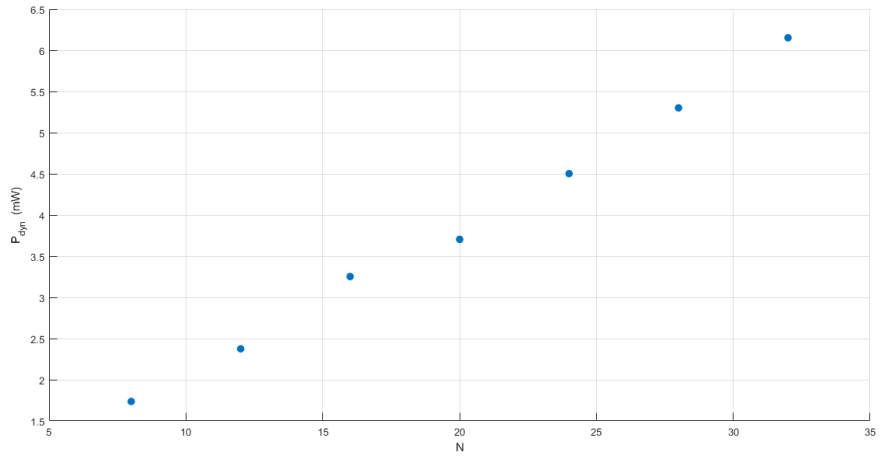

Figure 1.26: Dynamic power Pentium IV as a function of N

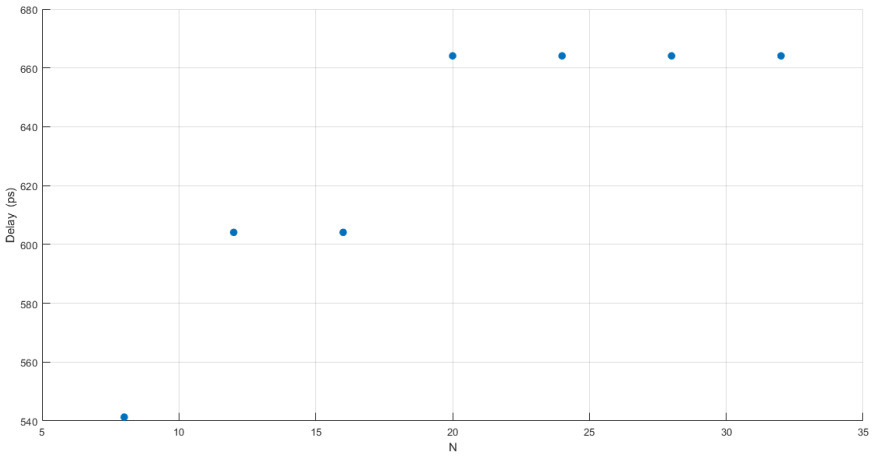

Figure 1.27: Delay without load Pentium IV as a function of N