

Chapter 2

Design in the Energy-Delay Space

Scaling trends have driven CMOS technology into a power-limited regime, where energy has to be managed wisely to make performance truly available within a given power budget [13]. This chapter deals with the design of energy-efficient digital circuits, i.e. to the achievement of the desired speed under minimum energy consumption. Energy-delay models of logic circuits and the theoretical background relative to the analysis of circuits in the energy-delay space are discussed, in order to identify design criteria to achieve energy efficiency.

2.1 Energy Modeling

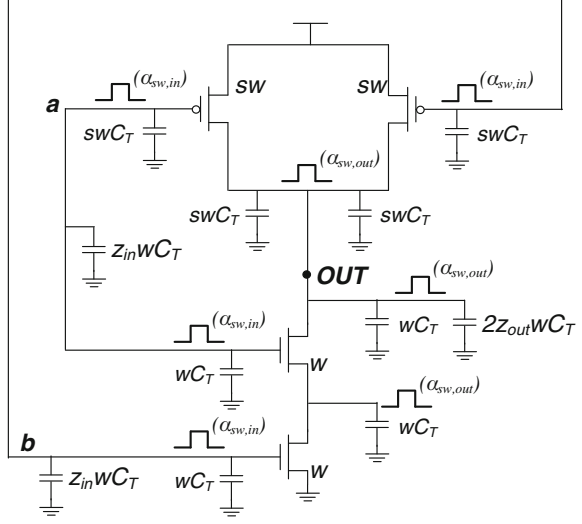
This chapter is focused on the joint optimization of speed and consumption in digital circuits. It is hence necessary to clarify the metrics that will be used to quantify the consumption at the transistor level of abstraction. In particular, two metrics are available: power and energy [24, 121].

Both metrics are actually interchangeable and choosing one or another is simply a matter of convention as long as transient (i.e., dynamic and short-circuit) and static (i.e., leakage) dissipative contributions are properly weighed [3]. In the following, energy is chosen as the metric for circuit consumption. This implies that transient contributions relative to a generic circuit operation have to be simply summed, whereas static leakage-related power has to be multiplied by the time between successive operations (e.g., the duration of a clock cycle in a pipelined system) and summed to the transient (dynamic) contribution to obtain the overall energy dissipation.

In the following, a model accounting for the above contributions is summarized [7, 8]. This model aims at the extraction of a factor χ featuring a logic gate and such that the overall gate energy E is simply expressed as linearly related to the input capacitance C_{IN} , i.e. to the gate size according to (2.1):

$$E = \chi \cdot C_{IN}. \quad (2.1)$$

Fig. 2.1 Capacitive contributions determining dynamic energy in a gate



Such a model intentionally excludes the energy dissipated in charging/discharging the load capacitance C_L , but includes the energy dissipated in charging/discharging C_{IN} and the capacitances at the internal nodes. Again, it is simply a matter of convention, when summing up all the energy contributions of the logic gates within the same module.

Let us consider a static CMOS gate such as the 2-inputs NAND in Fig. 2.1, where the various capacitive contributions determining the dynamic dissipation are depicted. One can distinguish among capacitances lying in the input nodes and switching according to the transition probability of the inputs $\alpha_{sw,in}$, and capacitances lying at the output node (or in the internal ones featuring stacked structures) and switching according to the transition probability of the output (internal) node $\alpha_{sw,out}$ [85].

Each of these capacitances is made up of nearly equal transistor contributions [144]: gate capacitances for the input nodes, and diffusion capacitances (drain-bulk, source-bulk) for the output and/or internal nodes. In particular, let us define

- C_T as the gate capacitance of a minimum-sized transistor, which can be defined as $C_{inv,min}/3$, being $C_{inv,min}$ the input capacitance of a symmetric minimum inverter (i.e., with $W_{PMOS} = 2W_{NMOS} = 2W_{min}$)
- s as a multiplicative factor that defines the widths of PMOS (again all equal and with minimum lengths) with respect to NMOS, thus leading to a certain skew in the speed of PUN and PDN [133].

The average dynamic energy per clock cycle of a CMOS gate is given by

$$E_{DYN} = [(1+s)\alpha_{sw,in} + (1+s)\alpha_{sw,out}] \cdot w \cdot C_T \cdot m \cdot V_{DD}^2 \quad (2.2)$$

where m is the number of gate inputs (equal to 2 for the NAND in Fig. 2.1), and it is assumed that each transistor contributes with a single gate and a single parasitic capacitance.¹

In order to include the parasitic capacitances due to local interconnects (within the logic gate) at the input and the output of the gate, let us introduce parameters z_{in} and z_{out} that weigh² parasitic capacitive contributions through the gate size w . Hence, the overall local wire capacitance $C_{par,j}$ at a generic node j can be expressed as [7, 8]

$$C_{par,j} = z_{out,i-1}w_{i-1}C_T + z_{in,i}w_iC_T \quad (2.3)$$

being j the node at the output and the input of the $(i-1)$ th and the i th stage, respectively, and the average dynamic energy (2.2) becomes

$$E_{DYN} = [(1 + s + z_{in})\alpha_{sw,in} + (1 + s + z_{out})\alpha_{sw,out}]w \cdot C_T \cdot m \cdot V_{DD}^2 \quad (2.4)$$

A similar analysis concerning the static dissipation of a CMOS gate can be carried out. In particular, let us define

- $\rho_{sub,n}$ ($\rho_{sub,p}$) as the sub-threshold leakage current of a minimum-sized transistor; in other words, these parameters express the leakage dependency on the technology and temperature, regardless of the size
- $\rho_{gate,n}$ and $\rho_{gate,p}$ are defined similarly, but they refer to gate leakage
- $T_{sub,n}$ ($T_{sub,p}$) as the PDN (PUN) sub-threshold leakage current at room temperature normalized to a single minimum-sized transistor; in other words, these parameters model the effect of the PDN (PUN) topology on its average leakage, averaging out the various currents for each input, and including the effect of transistor stacking
- $T_{gate,n}$ and $T_{gate,p}$ are defined similarly, but they refer to gate leakage
- $\beta_{sub,n}$ ($\beta_{sub,p}$) as the probability that the leakage of the gate is determined by the PDN (PUN), which is set by the statistics of input and output nodes of the gate (i.e., $\beta_{sub,n} + \beta_{sub,p} = 1$).

Accordingly, the average energy due to sub-threshold and gate leakage in a clock cycle with period T_{CK} is given by

$$E_{STAT} = w \left(\beta_{sub,n} \frac{\rho_{sub,n}}{T_{sub,n}} + s\beta_{sub,p} \frac{\rho_{sub,p}}{T_{sub,p}} + \frac{\rho_{gate,n}}{T_{gate,n}} + s \frac{\rho_{gate,p}}{T_{gate,p}} \right) V_{DD} \cdot T_{CK} \cdot \theta. \quad (2.5)$$

In (2.5), the parameter θ accounts for the percentage of time spent in active and standby mode by the module that the considered gate belongs to. In other words, θ is

¹ The approximation of considering a single inter-nodal capacitance for each stacked transistor is reasonably accurate, through the methods discussed in Chap. 1

² Although the dependence of such parasitics on w is in general nonlinear, linear fitting can be extracted without seriously compromising the accuracy of lumped local wires capacitances, as discussed in the following.

a correction factor that defines an effective clock period $T_{CK}\theta$, as a result of the fact that only leakage dissipation is present in standby mode, while dynamic one is not.

The above expressions (2.4) and (2.5) can be immediately generalized to model some effects more accurately. For instance, (2.4) and (2.5) can be easily generalized to deal with gates with non-minimum channel lengths, with non static gates (e.g., dynamic, ratioed logic), to accurately weigh the impact of inter-nodal capacitances on dynamic energy and of stacking effect on leakage, to consider the cases where some NMOS (PMOS) transistor within the PDN (PUN) has a width proportional but not equal to w , and so on. Hence, such models are very general but still reasonably simple for analysis purposes. Furthermore, as already discussed for the Logical Effort model in the previous chapter, many of the parameters in (2.4)–(2.5) can be accurately characterized through simulations.

Once E_{DYN} and E_{STAT} are evaluated, the overall energy dissipation of the gate is

$$E = E_{DYN} + E_{STAT}. \quad (2.6)$$

According to the previous definitions, C_{IN} including the wire parasitic capacitance can be expressed as

$$C_{IN} = (1 + s + z_{in}) \cdot w \cdot C_T. \quad (2.7)$$

This generalized model of the input capacitance can be plugged into Logical Effort calculations to include all the above discussed effects, which are not accounted for in the classical LE methodology.³ Accordingly, the parameter χ in (2.1) can be expressed as

$$\begin{aligned} \chi = & \left(\alpha_{sw,in} + \alpha_{sw,out} \frac{1 + s + z_{out}}{1 + s + z_{in}} \right) \cdot m \cdot V_{DD}^2 \\ & + \frac{\left(\beta_{sub,n} \frac{\rho_{sub,n}}{T_{sub,n}} + s \beta_{sub,p} \frac{\rho_{sub,p}}{T_{sub,p}} + \frac{\rho_{gate,n}}{T_{gate,n}} + s \frac{\rho_{gate,p}}{T_{gate,p}} \right)}{(1 + s + z_{in}) C_T} \cdot V_{DD} \cdot T_{CK} \cdot \theta. \end{aligned} \quad (2.8)$$

It is worth noting that the above model neglects short-circuit dissipation. Differently from the dynamic and leakage energy, the short-circuit contribution cannot be approximated as a linear function of the gate size. Indeed, it increases with gate size for three simultaneous reasons:

- the linear dependence of the PDN and PUN on-current on w
- the approximately proportional dependence on the input rise/fall time, i.e. on the output rise/fall time of the preceding gate [122]
- the approximately inverse dependence on the output rise/fall time of the gate itself [122].

³ The Logical Effort model does not consider the capacitive contributions due to local interconnects and those proportional to the input transistors size (here weighed through the z_{in} term).

The last two terms can be assumed as nearly linearly dependent on w , by neglecting the parasitic delays in the computation of input rise/fall times.

Overall, the short-circuit dissipation can be written as

$$E_{SC} = \frac{d_{in}}{d_{out}} \rho_{sc} [(T_{sc,n} + sT_{sc,p}) \alpha_{sw,out}] w \quad (2.9)$$

where d_{in} and d_{out} are input and output rise/fall times according to Logical Effort model, while parameters $T_{sc,n}$ and $T_{sc,p}$ average out the various possible output transition cases according to PDN and PUN topologies. Finally ρ_{sc} is a further parameter accounting for the impact of technology and V_{DD} .

The short-circuit energy tends to be rather small in nanometer technologies, due to the progressive reduction (although relatively slow, recently) in V_{DD}/V_{TH} [122]. The only and very infrequent case where the short-circuit energy is non negligible occurs when the input rise/fall times are much larger than the gate delay.

2.2 Energy-Delay Space Analysis and Hardware-Intensity

2.2.1 The Energy-Efficient Curve

For a digital circuit under a fixed supply voltage V_{DD} and whose last stage is loaded with a capacitance C_L , the “energy-efficient curve” (EEC) is made up of the design points (i.e., sizing of all transistors within the gate) exhibiting the minimum delay for a fixed energy dissipation or, equivalently, the minimum energy consumption for a given delay [4, 119]. By definition, other design points above the EEC lead to a needlessly higher energy under the same speed performances and need to be discarded, as shown in Fig. 2.2.

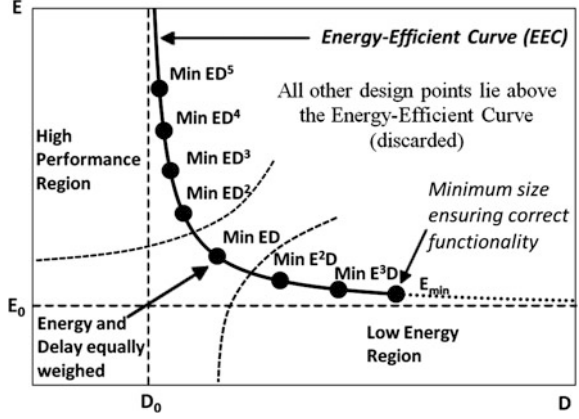
As previously stated, we consider the input capacitance of (the first stage of) the circuit C_{IN} as a further design variable to be optimized, and including (excluding) the energy dissipated in charging/discharging C_{IN} (C_L). This assumption is different from that adopted in [92, 111, 157, 158]. While it was a simple matter of convention when referring to the modeling of the energy of a circuit, we will show that it becomes a necessary assumption when the target is the full exploration of the E-D potential of a flip-flop topology.

In [119] it was predicted that the EEC of any circuit has a hyperbolic trend

$$(E - E_0)(D - D_0) = E_0 D_0 \quad (2.10)$$

being E_0 and D_0 the minimum energy and minimum delay asymptotes, as shown in Fig. 2.2. Actually, substantial deviations from (2.10) are found when analyzing

Fig. 2.2 Energy-efficient curve and designs optimizing the metrics $E^i D^j$



real circuits and hence a correction factor γ (typically $0 < \gamma < 1$) can be introduced to fit real data [157, 158]

$$(E - E_0)(D - D_0) = \gamma E_0 D_0 \quad (2.11)$$

Despite of the above mentioned differences with respect to [157, 158], the general character of (2.11) is retained. In particular, looking at the generic EEC depicted in Fig. 2.2, there is a minimum energy value E_{min} , which is achievable with the minimum transistors sizes allowing correct operation, hence the points between E_0 and E_{min} have not a physical correspondence (see Fig. 2.2).

Regarding the delay, the value D_0 can be approached only asymptotically through transistor sizing, and measures the maximum speed potential of a specific topology. More specifically, one can indefinitely trade energy for delay by increasing C_{IN} . On the contrary, if C_{IN} is fixed [92, 111, 157, 158], a minimum delay for a given load is actually reachable and corresponds to the Logical Effort sizing. The asymptotic value D_0 under an unbounded C_{IN} can be estimated as well through Logical Effort, and it corresponds to the parasitic delay P since it is simply the delay under large sizing.

As concerns parameter γ in (2.11) and the actual analytical expression of the EEC under our assumption, analytical calculations can be carried out only for a single logic gate [7, 8]. Indeed, according to Logical Effort model, one has

$$\frac{D - D_0}{D_0} = \frac{gh}{p} = \frac{g}{p} \frac{C_L}{C_{IN}} \quad (2.12)$$

As concerns the energy, from (2.1) we get

$$\frac{E - E_0}{E_0} = \frac{\chi C_{IN} - \chi C_{IN,min}}{\chi C_{IN,min}} = \frac{C_{IN} - C_{IN,min}}{C_{IN,min}} \quad (2.13)$$

being $C_{IN,min}$ the minimum input capacitance of the gate (i.e., when its transistors are all minimum sized).

By referring to (2.11) and using (2.12)–(2.13), γ is given by

$$\gamma = \frac{gC_L}{pC_{IN,min}} - \frac{D - D_0}{D_0} = \frac{g\tau\chi C_L}{D_0 E_0} - \frac{D - D_0}{D_0} \quad (2.14)$$

where τ is the delay of an inverter loaded by an inverter of the same size, i.e. the Logical Effort normalization delay parameter (see Chap. 1).

The above formula indicates that, under the above and more general assumptions, the value of γ in (2.11) actually depends on the variable D , i.e. the EEC is not a pure hyperbole. However, γ can be approximated in a sufficiently accurate way by its first term, $gC_L/pC_{IN,min}$ as long as the delay is not much higher than $D_0 = p\tau$. In this case, γ becomes constant and independent of D , and the EEC is a pure hyperbole.

Nevertheless, when dealing with circuits made up by more than one gate, no analytical expression can be determined for γ . In such case, γ is typically assumed to be a constant parameter in (2.11) for simplicity.

2.2.2 Energy-Delay Metrics and Hardware Intensity

In the last two decades digital circuit designers have become familiar with the use of composite energy-delay metrics to effectively translate the more and more stringent constraints on the speed, while not disregarding the energy dissipation.

The first (and at first glance the most appropriate) composite metric to be introduced is the simple ED product, which equally weighs the two quantities. Another popular metric is the ED^2 product where speed has priority over energy. The latter metric is claimed to have useful properties such as a nearly zero sensitivity on the supply voltage [80]. However, although designs optimizing (i.e., minimizing) the above metrics are maximally efficient for a given delay (or energy), it is clear that a generalization is required when exploring and designing a circuit over the entire spectrum of the delay (energy) values it can achieve.

Hence, the general class of metrics $E^i D^j$, or equivalently ED^η (being η equal to j/i) as originally presented in [119], are introduced. By varying the exponents $i \geq 0$ and $j \geq 0$ ($\eta \geq 0$), any tradeoff between energy and delay can be explored. The extreme cases are obtained when $j/i = 0$ ($\eta = 0$) and when $j/i = \infty$ ($\eta = \infty$), which, once optimized, represent the designs having the minimum possible energy and delay, respectively.

Interestingly, every design solution minimizing a metric $E^i D^j$ (ED^η) lies in the EEC [119], and the latter curve is hence made up of all points that minimize $E^i D^j$ (ED^η) for some i and j (η), as shown in Fig. 2.2. The demonstration of this assertion is quite simple and intuitive. Indeed, considering a circuit under a fixed

load and supply voltage, both its delay and energy are functions of its sizing W (W is an array containing the sizes of transistors in all circuit gates). A design minimizing an $E^i D^j$ metric for some (i, j) has a delay D^* which is obtained with a certain size W^* (i.e., $D^* = D(W^*)$). Since the size W^* minimizes a product $E^i D^j$, in which the energy is taken into account with $i \geq 0$, the value $E^* = E(W^*)$ of this design will be the minimum among all the designs exhibiting a delay $D = D^*$ and thus it lies on the EEC. More rigorous analytical proofs can be found in [119].

From the above considerations, the indexes i and j (η) define cost functions for optimizing hardware under a fixed load and supply voltage. According to [153, 157, 158], the value j/i (η) is named “hardware intensity”. Basically, j/i (η) quantifies the effort to be spent in sizing a circuit to optimize the speed of the circuit at the expense of its energy consumption. The higher j/i (η), the higher the effort (i.e., energy) to further improve the speed. The set of points of the EEC curve where metrics $E^i D^j$ is minimized for $j > i$ ($\eta > 1$) is hence called the high-performance region, whereas the region with minimum $E^i D^j$ with $j < i$ ($\eta < 1$) is called the low energy region. The former is featured by lower and lower delay gains achieved at the cost of larger and larger increments in energy as long as the delay diminishes. Dual considerations are valid for the low energy region.

The graphical interpretation of hardware intensity is shown in Fig. 2.3 [153, 158]. The solid line plots a typical EEC for a generic circuit. Dotted curves show several contours of the cost function $E^i D^j$ for three values of the hardware intensity. The point in the E-D space at which the EEC tangents the lowest of the contours corresponds to the energy-efficient implementation of the circuit for that specific hardware intensity value [157, 158].

Accordingly, the analytical interpretation of hardware intensity is related to the energy-to-delay sensitivity evaluated in correspondence of the design points optimizing the $E^i D^j$ (ED^η) metrics [7, 8, 157, 158]. Indeed, the design point minimizing $E^i D^j$ for a given (i, j) leads to a zero derivative of $E^i D^j$ with respect to D and E [7, 8, 119]

$$\frac{\partial(E^i D^j)}{\partial D} \Big|_{E^i D^j_{min}} = \left(iE^{i-1} D^j \frac{\partial E}{\partial D} + jE^i D^{j-1} \right) \Big|_{E^i D^j_{min}} = 0 \quad (2.15)$$

$$\frac{\partial(E^i D^j)}{\partial E} \Big|_{E^i D^j_{min}} = \left(iE^{i-1} D^j + jE^i D^{j-1} \frac{\partial D}{\partial E} \right) \Big|_{E^i D^j_{min}} = 0 \quad (2.16)$$

Solving the set of equations (2.15)–(2.16), one finds

$$S_D^E \Big|_{E^i D^j_{min}} = \left(\frac{\partial E}{\partial D} \frac{D}{E} \right) \Big|_{E^i D^j_{min}} = -\frac{j}{i} \quad (2.17)$$

Equivalently, when carrying out analogous calculations by referring to the ED^η metrics, the results is simply $-\eta$. Anyhow, the adoption of the two indexes i and j

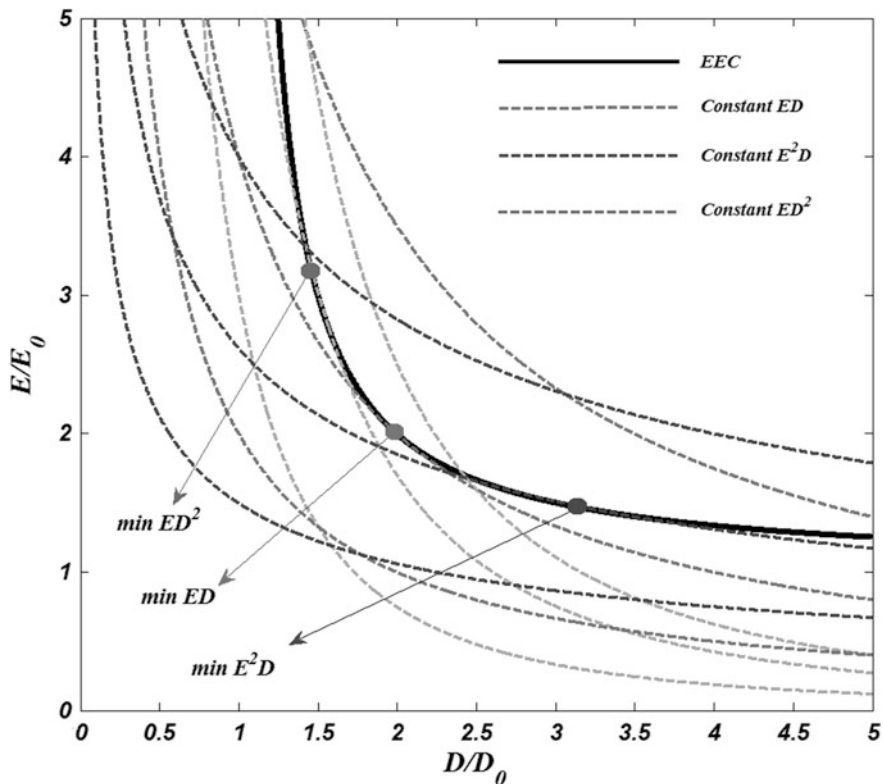


Fig. 2.3 Typical energy-efficient curve and constant cost function contours for $j/i = 1.0$, $j/i = 0.5$ and $j/i = 2.0$

has a clearer physical meaning. Indeed, in the neighborhood of the optimum $E^i D^j$ design point, a $j\%$ speed increase is traded for a $i\%$ energy increment and vice versa. Finally, from (2.17) it is apparent that metrics leading to the same j/i ratio are not distinguishable.

2.2.3 Voltage Intensity and Generalization of the Sensitivity Criterion

So far we have focused on hardware, i.e. transistors sizing, optimization. However, other tuning variables are available at the circuit level, such as the supply voltage V_{DD} and the transistors threshold voltages.

As concerns supply voltage, by introducing the dimensionless derivatives of energy and delay with respect to the supply voltage v ,

$$E_v = \frac{v}{E} \frac{\partial E}{\partial v} \quad (2.18)$$

$$D_v = -\frac{v}{D} \frac{\partial D}{\partial v} \quad (2.19)$$

and taking their ratio, one can define “voltage intensity” β as the energy-to-delay sensitivity relative to the variation of v at a fixed hardware intensity j/i [157, 158]. Hence, just as j/i represents the negative of the ratio between the energy (delay) relative gain and the cost of the relative increase in delay (energy) achievable by sizing transistors under a fixed v , β represents the energy (delay) relative increase (decrease), achievable by increasing v under a fixed w [157, 158]

$$\beta = -\left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{v \text{ variable}-w \text{ fixed}}. \quad (2.20)$$

The E_v and D_v values cannot be simply determined through classical $E \propto V_{DD}^2$ and $D \propto 1/(V_{DD} - V_{TH})^2$, given the impact of leakage and short-circuit currents on energy and the complexity of $I_D = f(V_{GS}, V_{DS})$ relationship featuring nanometer transistors. Therefore, it is necessary to develop comprehensive models of energy and delay as functions of the supply voltage [9] (similarly to those relative to transistors sizing that were discussed in the previous paragraph) or extract E_v and D_v for the various gates in a circuit through simulations. To have an idea of the main trends [158], the voltage intensity β almost linearly increase with V_{DD} for typical CMOS circuits.

The most important aspect of this discussion is that hardware and voltage intensities are related when optimizing a circuit in the E-D space. If we consider a sub-circuit that has to satisfy a given maximum delay constraint (e.g., a pipeline stage), this requirement can be achieved at different combinations of the j/i and β values. However, the energy-efficient implementation, i.e. that with the minimum energy, is the one featured by

$$j/i = \beta. \quad (2.21)$$

Indeed, energy and delay are functions of the variables (w, v) , and, by solving the problem of minimizing $E(w, v)$ under the constraint $D(w, v) = D^*$, one finds [30, 157, 158]

$$\frac{\partial D}{\partial w} \frac{\partial E}{\partial v} = \frac{\partial D}{\partial v} \frac{\partial E}{\partial w} \quad (2.22)$$

which means $j/i = \beta$. Hence, to reach an energy-optimal balance between the supply voltage and the transistors sizing, the relative speed gain achieved at the cost of a given relative energy increase due to an increment in the supply voltage must equal the relative speed gain achieved at the cost of a given relative energy

increase due to a larger transistors sizing [158]. This result disproves the common misconception that the lowest energy can be achieved by designing circuit for the highest speed and then reducing the power supply up to the lowest value that satisfies the delay requirement.

Further generalizing the above analysis to any kind of design variable (e.g., threshold voltages) [38, 77], and to the sensitivity $S_x(X)$ of energy to delay with respect to a change in variable x , as in (2.18)–(2.19), the minimum energy under a given delay constraint is achieved when [92]

$$S_x(X) = \left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{x \text{ variable}} = S_y(Y) = \left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{y \text{ variable}} \quad \forall x, y \quad (2.23)$$

being x and y design variables. In other words, the energy-efficient corresponds to the design with $x = X$, $y = Y$ and so on.

2.3 Energy-Efficient Design of Digital Circuits

In this section we discuss practical optimization techniques to achieve the energy-efficient design of digital circuits at the circuit level, by considering various levels of complexity. In particular, we first provide some preliminary remarks concerning the role played by the input capacitance of the circuit and the definition of design space bounds, both essential regardless of the actually employed optimization technique. Then, we consider the case of simple basic blocks whose complexity allows a simulations-based optimization and finally with large designs that can be dealt with by resorting to convex optimization and exploiting simple E-D models.

2.3.1 The Role of the Input Capacitance

As shown in [30, 111], the input capacitance C_{IN} of a logic circuit cannot be simply assumed as fixed, when designing for energy efficiency. Granted that the adopted C_{IN} value is also related with the architectural-level design strategies [30], differently from [92, 111, 157, 158], here we consider C_{IN} as an additional design variable to be fully optimized like any other transistor size. This choice is justified by considering that C_{IN} strongly affects the E-D tradeoff, as discussed below.

A second assumption, differently to [30, 92, 111, 157, 158], is to include the energy dissipated in the (dis)charge of C_{IN} and to exclude the energy dissipated in the (dis)charge of C_L . Indeed, the first term is inherently related to the adopted circuit sizing, whereas the latter term does not depend on the features of the topology [41, 131].

In general, throughput improvements can be achieved by means of an increase in the degree of parallelism and/or a more aggressive sizing of all the gates in the

logic paths (e.g., when the sequential part of code running on a microprocessor system is dominant, hence parallelization is not effective). In the latter case, if C_{IN} is increased with respect to a reference design, it means that the topology is being sized up to achieve a higher speed at the cost of higher energy. Even if the circuit imposes a larger load on the preceding logic stage (e.g. in a pipeline), in high-speed applications the speed penalty of the preceding logic stages could be exceeded by the speed improvement in the considered topology. This tradeoff cannot be explored if one does not assume a fully variable C_{IN} . Similar considerations hold in regard to low-energy designs. Obviously, there always exist practical limits on the adoptable C_{IN} values. Nevertheless, once the full EEC is extracted, the designer can easily select the portion of interest according to practical constraints in terms of maximum allowable C_{IN} , so that the interaction between the delays of adjacent stages through the C_{IN} of the successive one is captured (as it is typically done by setting a fixed C_{IN}).

Finally, it is worth highlighting that, when referring to the “first stage”, we mean the first gate in the path of the circuit whose sizing is assumed as a reference in terms of timing criticality. Indeed, several input-to-output paths coexist in real circuits, and the target must obviously be that of equaling these delays. Among the various paths, it is then possible to identify one (typically the longest) that is adopted as the reference to identify the C_{IN} of the circuit. Note that, since C_{IN} is fully varied and the optimization targets the equality of the various concurrent delays, the input capacitances of the first stages of all the other paths will be optimized and fully explored as well.

2.3.2 Derivation of Design Space Bounds

Regardless of the methodology employed for EEC extraction, practical design space bounds need to be identified to limit the space of solutions. As will be shown successively, this issue is particularly important in the case of simulations-based procedures and nonlinear optimizations, to keep the computational effort within reasonable bounds. On the contrary, this issue becomes less relevant when one adopts simple gate-level E-D models and resorts to convex optimization.

In [30] it is shown that Logical Effort designs lie above the EEC, i.e. they are not the most efficient possible designs. Even if C_{IN} is assumed as a design variable unlike [30], the same result still holds.⁴ Nevertheless, the energy-to-delay sensitivity of Logical Effort designs can be used as an indicator to determine design

⁴ Indeed, as explained in the previous section, the minimum energy under a given speed constraint is reached when the sensitivity with respect to “all” the tuning variables is the same. Logical Effort designs are featured by an infinite energy-to-delay sensitivity with respect to the sizes of internal transistors (since delay cannot be further reduced given a fixed C_{IN}), but not with respect to the size of transistors defining C_{IN} . Hence, the condition in (2.23) is not satisfied for Logical Effort designs, which thus are not energy-efficient [30]. Only when C_{IN} approaches

space bounds. More specifically, let us assume that the designer is interested in the portion of the EEC up to a certain minimum- $E^i D^j$ design point with $j/i = X$, i.e. the portion of the EEC made up by energy-efficient designs that minimize FOMs with j/i less than or equal to X . In this case, the range of the transistor sizes to be explored can be defined by upper bounding the energy sensitivity to delay to a given value X , which translates into the upper bound $C_{IN,max}$ of C_{IN} that satisfies the following condition [7, 8]

$$S_{D|C_{IN}}^E = \frac{S_{C_{IN}}^E}{S_{C_{IN}}^D} = -\frac{j}{i} = -X. \quad (2.24)$$

The definition of $C_{IN,max}$ leads also to the definition of the upper bounds for the other design variables (i.e. transistors sizes) that are clearly defined by the Logical Effort sizing with $C_{IN} = C_{IN,max}$.

Equation (2.24) can be analytically evaluated thanks to the properties of Logical Effort designs. In particular, as discussed in Chap. 1, given C_{IN} and C_L , the minimum normalized path delay D_{TOT} of a circuit simply made up by a path of N cascaded gates is

$$D_{TOT} = N \sqrt[N]{GBH} + P = N \sqrt[N]{F} + P \quad (2.25)$$

which can be rewritten as

$$D_{TOT} = P(1 + k) \quad (2.26)$$

where

$$k = \frac{N \sqrt[N]{GB} \sqrt[N]{C_L}}{P \sqrt[N]{C_{IN}}} \quad (2.27)$$

is the relative delay increment with respect to the ideal and practically unachievable minimum path delay (i.e., the path parasitic delay P). From (2.26)–(2.27), the sensitivity of the optimized path delay D_{TOT} to C_{IN} is given by

$$S_{C_{IN}}^{D_{TOT}} = \frac{\partial D_{TOT}}{\partial C_{IN}} \frac{C_{IN}}{D_{TOT}} = -\frac{1}{N} \frac{k}{k + 1} \quad (2.28)$$

which is a function of C_{IN} only.

As for the path delay D_{TOT} , it is possible to unambiguously determine the energy E_{TOT} of a single path circuit sized through Logical Effort for a given C_{IN} and C_L . According to (1.33) and (1.34), the input capacitance C_N and the energy E_N of the N th gate are respectively given by

(Footnote 4 continued)

infinity, the Logical Effort design will be featured by an equal (tending to infinity) energy-to-delay sensitivity with respect to all the tuning variables.

$$C_N = \frac{g_N b_N \sqrt[N]{C_{IN}}}{\sqrt[N]{GBC_L}} C_L \quad (2.29)$$

$$E_N = \chi_N C_N. \quad (2.30)$$

By iterating the above reasoning and going backwards through the path, one finds that the input capacitance and energy of the i th gate under classical Logical Effort design are

$$C_i = \frac{\left(\prod_{j=i}^N g_j\right) \left(\prod_{j=i}^N b_j\right) (C_{IN})^{\frac{N-i+1}{N}}}{(GBC_L)^{\frac{N-i+1}{N}}} C_L \quad (2.31)$$

$$E_i = \chi_i \cdot C_i \quad (2.32)$$

and $C_1 = C_{IN}$.

Therefore, the overall dissipation of the reference path is

$$E_{TOT} = \sum_{i=1}^N \left[\chi_i \frac{\left(\prod_{j=i}^N g_j\right) \left(\prod_{j=i}^N b_j\right) (C_{IN})^{\frac{N-i+1}{N}}}{(GBC_L)^{\frac{N-i+1}{N}}} C_L \right] \quad (2.33)$$

and the sensitivity of the overall energy E_{TOT} to C_{IN} can be again expressed as a function of the only C_{IN} :

$$S_{C_{IN}}^{E_{TOT}} = \frac{\sum_{i=1}^N \left[\chi_i \frac{\left(\prod_{j=i}^N g_j\right) \left(\prod_{j=i}^N b_j\right) (C_{IN})^{\frac{N-i+1}{N}}}{(GBC_L)^{\frac{N-i+1}{N}}} \left(\frac{N-i+1}{N}\right) C_L \right]}{\sum_{i=1}^N \left[\chi_i \frac{\left(\prod_{j=i}^N g_j\right) \left(\prod_{j=i}^N b_j\right) (C_{IN})^{\frac{N-i+1}{N}}}{(GBC_L)^{\frac{N-i+1}{N}}} C_L \right]} \quad (2.34)$$

Finally, (2.28) and (2.34) can be combined to evaluate (2.24) and determine $C_{IN,max}$.

Unfortunately, formula (2.24) cannot be always applied straightforwardly since g_i , h_i , b_i and p_i are often not available in a closed-form⁵ as function of C_{IN} . Rather, g_i ,

⁵ There are three main reasons for this issue:

- (1) there are various sources of non-linearity, as listed in the Sect. 2.2, which imply the need for iterative procedures to determine the Logical Effort sizing.
- (2) not all the transistor sizes need to be optimized. Actually, only transistors lying in input-to-output paths should represent variables to be optimized in the E-D space, since they affect both consumption and speed. On the contrary, there can exist some parts of the circuit whose size is simply the minimum guaranteeing correct operation, since they affect only energy. This is the case for instance of keepers, pulse generators, and so on. However, these gates have a size dependent on the design variables to be optimized (to guarantee the correct operation) and hence affect the branching effort b_i in a non-linear way.

h_i , b_i and p_i themselves can be found only by numerically solving a set of complex non-linear equations when applying the Logical Effort method for a given C_{IN} (see Footnote 5). Furthermore, when the circuit is not simply made up by a single path, also the energy of the circuit is not simply that in (2.32) (see Footnote 5), and it is not always possible to find closed form relationships describing the energy of the other gates as functions of C_{IN} . Nevertheless, one has to keep in mind that the energy still depends on the only variable C_{IN} , when sizing for maximum speed.

From the above considerations, the following iterative procedure can be adopted to define the range of C_{IN} for a targeted maximum sensitivity $-j/i$ [4, 7, 8]:

- (a) under the current C_{IN} (re)apply the Logical Effort method to find the transistor sizes leading to the minimum delay of all the concurrent paths in the circuit (a non-linear set of equations must be solved, see Footnote 5)
- (b) (re)simulate energy and delay
- (c) (re)extrapolate the fitted curve of E_{TOT} versus C_{IN} and the fitted curve of D_{TOT} versus C_{IN} fitted curves, and (re)compute the sensitivity (2.24) around the current C_{IN} value
- (d) (re)compare such sensitivity with the desired one $-j/i$; if $|S_{D_{TOT}}^{E_{TOT}}|_{C_{IN}}| < |j/i|$, C_{IN} is increased and another iteration starting from (a) is needed. Otherwise, no more iteration is needed, hence $C_{IN,max}$ and the upper bound for all transistor sizes is found.

To exemplify the above procedure, let us consider a 4-bit Ripple-Carry Adder in a 65-nm technology, whose schematic is shown in Fig. 2.4, under a load equal to 16 minimum inverters and $V_{DD} = 1V$. In Fig. 2.5 we show the energy-to-delay sensitivity relative to the variation of C_{IN} . The x -axis corresponds to the value of the transistor width w_1 (normalized to the minimum W_{min}) determining the size of the first stage of the circuit (i.e., C_{IN}), while other four transistors normalized widths $w_2 - w_5$ are further tuning variables (see Fig. 2.4 and [7] for details).

From inspection of Fig. 2.5, according to the above discussed procedure, one has that the minimization of the ED^3 metric (i.e., $-j/i = -3$) requires $w_1 > 15$,

(Footnote 5 continued)

- (3) The possible presence of reconvergent paths or multiple outputs. Indeed, transistors in the paths that lie nearby to the path assumed as the reference one will affect speed too, since they must be sized so that all concurrent paths exhibit the same delay. When formulating the Logical Effort equations, besides satisfying (1.35) for stages in the reference path, additional equations relate the delays of the various concurrent paths. This makes the problem of finding the minimum delay design even more complex and nonlinear.

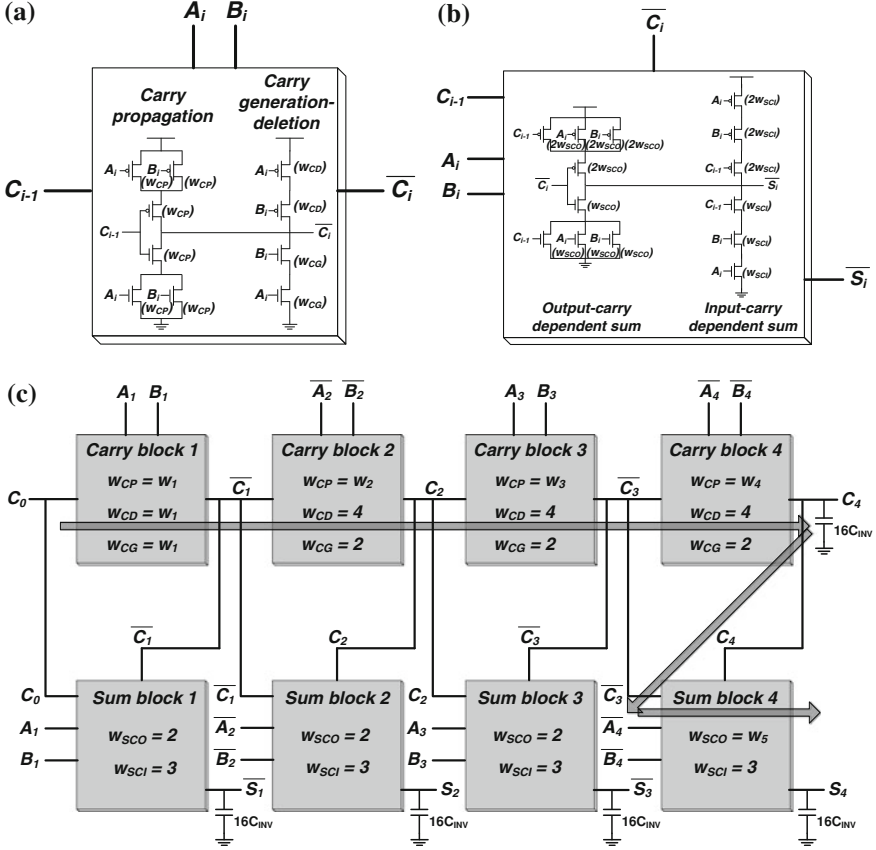


Fig. 2.4 4-bit RCA: carry block (a), sum block (b), whole structure (c)

while the minimization of the ED^4 metric requires $w_1 > 31$. The corresponding bounds on the other variables $[w_2, w_3, w_4, w_5]$ are $[17, 18, 17, 7]$ for the ED^3 metric and $[31, 30, 25, 9]$ for the ED^4 metric [7]. These are clearly upper bounds to the transistors sizes that optimize the two metrics, which are equal to $[15, 17, 17, 16, 6]$ and $[29, 30, 30, 18, 10]$, respectively [7].

Summarizing, these results confirm the effectiveness of such a procedure, which aims at practically bounding the design space through the analysis of the energy-to-delay sensitivity relative to the variation of C_{IN} , while still using classical Logical Effort.

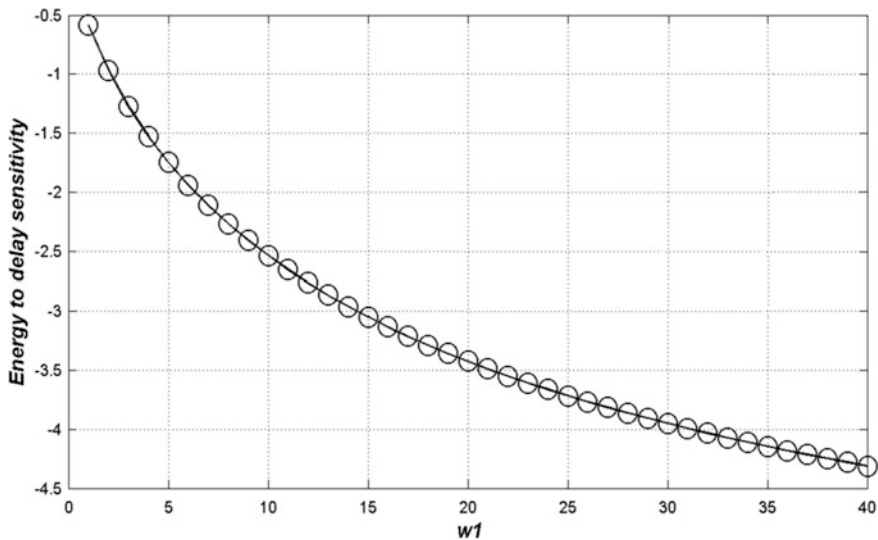


Fig. 2.5 4-bit RCA: energy-to-delay sensitivity of Logical Effort designs as a function of the first stage size

2.3.3 Simulation-Based Optimization of Small-Sized Circuits

When dealing with small circuits featured by few design variables (i.e., simple basic circuit blocks), the energy-delay optimization can be carried out by employing a simulation-based procedure [4–8].

From the properties of the $E^i D^j$ metrics discussed in the previous section, from a practical perspective the EEC of a circuit can be extracted by simply minimizing $E^i D^j$ for a limited number of pairs (i, j) and interpolating such optimum points. In particular, a binary search can be employed to identify minimum- $E^i D^j$ designs because $E^i D^j$ functionals are nearly convex in the design space [4]. Moreover, the transistor size space to be explored can be progressively reduced through an iterative procedure described in the following. Indeed, assuming $j_1/i_1 < j_2/i_2$, a design optimizing $E^{i_1} D^{j_1}$ will always lead to sizes that are smaller than optimizing $E^{i_2} D^{j_2}$. Therefore, one can start from the metric with the highest j/i ratio. Then, once it is optimized with a sizing W' , the optimization of the successive metric (with decreased j/i value) will be constrained by bounding the design space with the sizing W' , and so on [4].

To exemplify the above search algorithm, we report the results relative to the simulations-based extraction of the EEC for the 4-bit adder previously mentioned. In Fig. 2.6 the design points explored in the search space are depicted with small circles, while the energy-efficient ones minimizing some $E^i D^j$ metrics are highlighted. It is apparent that the explored designs crowd near the EEC, thus highlighting the search algorithm effectiveness.

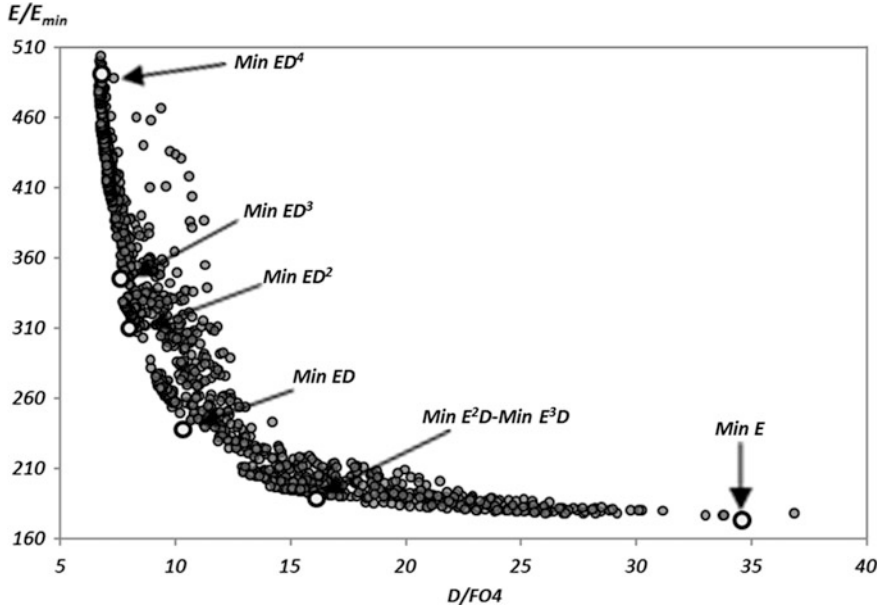


Fig. 2.6 Energy-delay space exploration for the 4-bit RCA ($C_L = 16C_{INV}$, $V_{DD} = 1V$)

Table 2.1 4-bit RCA:
minimum E^iD^j designs

Sizing ↓	D [FO4]	E [E_{min}]	S_D^E	$-j/i$
Min ED^4	6.79	490.89	-4.24	-4.00
Min ED^3	7.62	345.12	-2.78	-3.00
Min ED^2	7.99	310.12	-1.85	-2.00
Min ED	10.32	238.00	-0.92	-1.00
Min E^2D	16.11	188.62	-0.37	-0.50
Min E^3D	16.11	188.62	-0.37	-0.33
Min E	34.59	173.43	-0.08	-0.00

As a further validation, we also evaluate the energy-to-delay sensitivity in the minimum E^iD^j points and compare with the theoretically expected $-j/i$ value, as shown in Table. 2.1. Results confirm again that the described search algorithm allows for identifying the minimum E^iD^j points with adequate accuracy for design purposes.

2.3.4 Nonlinear and Convex Optimization of Large Size Circuits

When dealing with circuits of fairly large complexity (e.g., from several tens to several thousands of design variables), a simulations-based optimization becomes

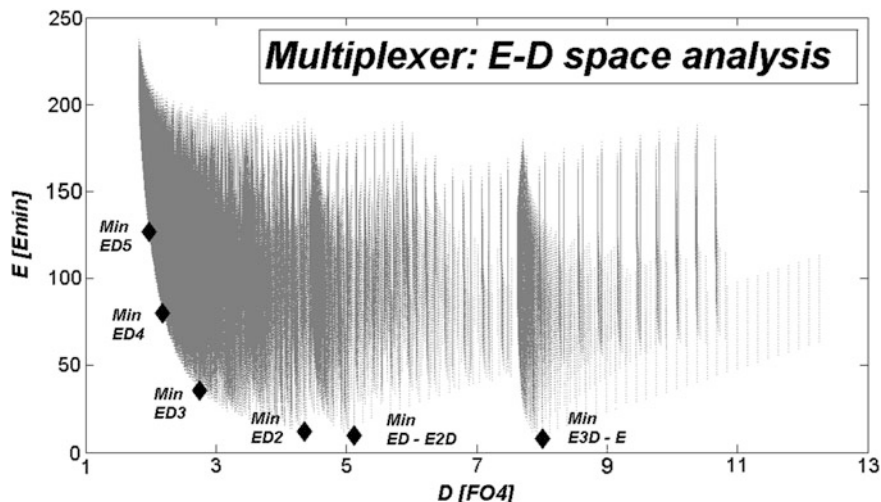


Fig. 2.7 Full E-D space exploration for a buffered 2:1 multiplexer

infeasible because of its prohibitive computational effort. To give an idea, the full E-D space exploration of a simple buffered 2:1 multiplexer, featured by five design variables (transistors widths swept with a W_{min} step), takes nearly a minute on a current desktop computer when using the E-D models in the second paragraph and the previous procedure to determine the design space bounds. The tens of millions designs explored are shown in Fig. 2.7. Considering larger circuits, the complexity grows exponentially and a full exploration soon becomes infeasible.

If the objective function to be minimized (e.g., energy) and constraints functions to be satisfied (e.g., delay) do not have any special feature (e.g., convexity), the optimization problem is said to be a “nonlinear optimization” or a “nonlinear programming” [17]. This is actually the case when both energy and delay are very accurately modeled by accounting for several effects even in complex ways (e.g., short-circuit currents, impact of input slope on the delay, dependence of leakage on the threshold voltages, etc.).

As long as the design variables are no more than several tens, global optimization algorithms, ensuring that the true global optimum solution is found, can be applied while still maintaining the computational effort feasible, i.e. from hours to no more than few days [17]. Obviously, the accuracy in E-D estimation is worse than simulation-based approaches, but a much broader exploration of the design space can be performed in a comparable time [115]. Note that in such a case, the above discussed definition of proper design space bounds is still of great importance, as it is in simulation-based approaches.

When dealing with circuits featured by more than a hundred design variables, non-linear programming is no longer able to reliably determine the solution of the optimization problem (i.e., the actual global optimum is missed). Therefore, the

focus must be on the adoption of the most accurate possible E-D models leading to optimization problems that can be reliably solved in a feasible time.

A class of problems that can be reliably and fast solved is the convex optimization, where both the objective and constraint functions are convex [17] (see the Appendix at the end of the chapter). There is in general no analytical formula for the solution of convex optimization problems, but there are very effective methods for solving them like interior-point methods [17] or other custom methods. For instance, the method proposed in [61] is claimed to size circuit with a million gates in nearly an hour. Furthermore, thanks to the properties of the above solving methods, the definition of design space bounds and of the initial point becomes irrelevant.

Hence, it is apparent that the required computational effort is incomparably lower than that required in the previous cases. The other side of the coin is that the formulation itself requires a simplification of the E-D models that lowers the accuracy in their estimation. Nevertheless, this is the only feasible approach when the circuit size is large enough.

A class of convex optimization problems that really suits the problem of digital gate sizing for energy efficiency is called “generalized geometric programming (GGP)”, where the objective and constraint functions take the special form of “generalized posynomials” (“monomials” for the equality constraints). A brief overview on convex optimization and GGP is reported in the Appendix of the chapter, while a detailed and full mathematical treatment can be found in [17] and [16]. In particular, in [15] a comprehensive list concerning the applicability of GGPs to the design of digital circuits can be also found, which includes:

- the minimization of energy/power (or area) of logic circuits under speed (e.g., delay, clock frequency) constraints, i.e. the energy-efficient design;
- wires sizing in *RC* tree networks;
- statistical optimization under PVT variations.

As previously discussed, energy and delay have to be modeled with adequate accuracy to ensure convergence to the actual global optimum. As concerns delay, *RC* based models linearly including the impact of input slope (see Sect. 2.2) are typically adopted [21, 32, 118], while energy is typically modeled as proportional to gates sizes [see (2.1)].

2.4 Design of Energy-Efficient Pipelined Systems

When dealing with custom datapaths, the design of energy-efficient pipelined systems is essential to achieve the desired throughput (or clock frequency) while keeping energy minimum.

Convex optimization methods allow for optimizing any kind of digital circuits featured by several concurrent constraints, as in the case of pipelined systems. However, formulating the problem as (for instance) GGP and solving it by relying

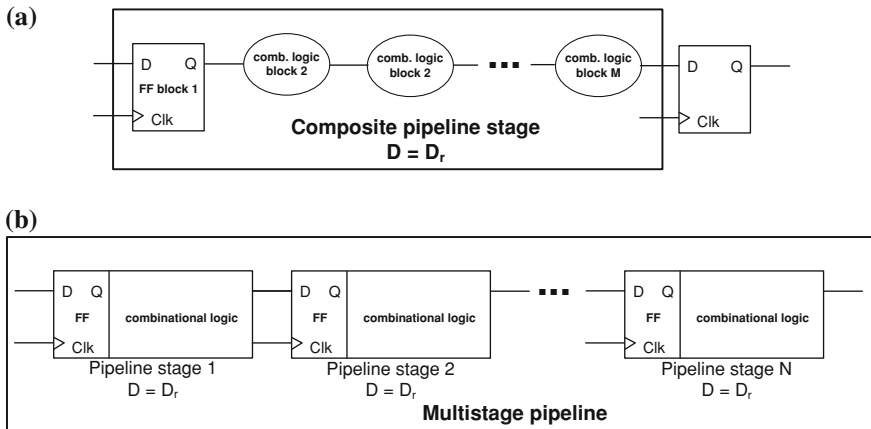


Fig. 2.8 Composite pipeline stage (a) and multistage pipeline (b)

upon the related mathematics, makes one lose sight of the relevant aspects pertinent to the design of an energy-efficient pipeline. In such sense, the papers by Zyuban and Strenski [157, 158] and a subsequent work (e.g., [30]) attempt to provide a deeper insight into pipelines.

In this paragraph, we refer to pipelines that are made up of stages made up of circuit blocks of different complexity and granularity (e.g., flip-flops, an adder, a multiplier, etc.). Finally, we refer to a pipeline where a block comprises only basic logic gates (e.g., inverters, NAND gates, NOR gates, etc.).

2.4.1 Zyuban and Strenski's Hardware-Voltage Intensity Criteria

According to (2.21), the minimum energy of a single circuit under a given delay constraint is achieved when hardware intensity η and voltage intensity β are equal. The analysis can be extended to the cases of:

- a composite pipeline stage made up of several blocks (see Fig. 2.8a). The speed constraint is expressed in terms of the overall stage delay, as in the case of a single circuit. However, here we are separately targeting the energy and delay contributions from the various underlying blocks.
- A multistage pipeline with composite stages (see Fig. 2.8b), i.e. various pipeline stages subject to the same delay constraint.
- a multistage pipeline with composite stages, i.e. various pipeline stages subject to the same delay constraint, where the energy and delay contributions from the various underlying blocks are separately targeted.

(a) *A composite pipeline stage*

Assuming the pipeline stage is made up of M cascaded sub-blocks, we have to minimize the overall energy

$$E(w_1, w_2, \dots, w_M, v) = \sum_{i=1}^M E_i(w_i, v) \quad (2.35)$$

being w_i the normalized sizes of the various blocks and v the supply voltage, under the constraint that the overall delay is equal to a given value D_r

$$D(w_1, w_2, \dots, w_M, v) = \sum_{i=1}^M D_i(w_i, v) = D_r. \quad (2.36)$$

The solution of the problem can be easily found by using the Lagrange multipliers method [30], and corresponds to the condition

$$\frac{e_i}{d_i} \eta_i = \beta \quad \forall i = 1 \dots M \quad (2.37)$$

where $e_i = E_i/E$ and $d_i = D_i/D$ are the energy and delay fraction of the i th block relative to the entire pipeline stage, η_i is the hardware intensity of the i th block and β is the stage voltage intensity, i.e.

$$\eta_i = - \left. \frac{\partial E_i}{\partial D_i} \frac{D_i}{E_i} \right|_{w_i \text{ variable}/(w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_M, v) \text{ fixed}} \quad (2.38)$$

$$\beta = - \left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{v \text{ variable}/(w_1, w_2, \dots, w_M) \text{ fixed}}. \quad (2.39)$$

Thus, in a pipeline stage with multiple cascaded blocks that have been designed independently, those that have lower energy weight and higher delay weight should be designed more aggressively than blocks with lower delay weight and higher energy weight.

The aggregate hardware intensity of the whole pipeline stage cannot be in general related to the hardware intensities of the underlying blocks, given that one has [158]

$$\frac{\partial E}{E} = - \sum_{i=1}^M \left[\frac{e_i}{d_i} \eta_i \frac{\partial D_i}{D} \right]. \quad (2.40)$$

However, when condition (2.37) is satisfied (i.e., all blocks are jointly optimized), from (2.40) one finds that the aggregate hardware intensity of the whole pipeline stage is equal to those of the various blocks, i.e.

$$\eta = - \left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{(w_1, w_2, \dots, w_M) \text{ variable}/v \text{ fixed}} = \frac{e_i}{d_i} \eta_i = \beta \quad \forall i = 1 \dots M. \quad (2.41)$$

(b) *A multistage pipeline*

Practically, different stages of the pipeline usually have different complexity and energy-delay tradeoff, hence it would be incorrect to tune all of them for the same value of the hardware intensity.

Assuming the pipeline is made up of N stages, we have to minimize the overall energy

$$E(W_1, W_2, \dots, W_N, v) = \sum_{i=1}^N E_i(W_i, v) \quad (2.42)$$

being W_i the sizes of the various stages, under the constraint that the delays of the various stages are all equal to a given value

$$D_i(W_i, v) = D_r \quad \forall i = 1 \dots N \quad (2.43)$$

Note that each i th stage is in turn made up of M_i blocks and hence the sizing W_i should be more properly expressed as

$$W_i = (w_{i,1}, w_{i,2}, \dots, w_{i,M_i}) \quad (2.44)$$

The solution of the problem can be again easily found by using Lagrange multipliers [30], and corresponds to the conditions

$$\sum_{i=1}^N e_i \eta_i = \beta \quad \forall i = 1 \dots N. \quad (2.45)$$

The above relationship can be used to reevaluate the choice of the supply voltage and the clock cycle target, and possibly the partitioning of the pipeline into stages.

This time the aggregate hardware intensity of the whole multistage pipeline can be computed from the hardware intensities of the various stages and corresponds to the left side of (2.45) equation [158], i.e.

$$\eta = - \left. \frac{\partial E}{\partial D} \frac{D}{E} \right|_{(W_1, W_2, \dots, W_N) \text{ variable}/v \text{ fixed}} = \sum_{i=1}^N e_i \eta_i. \quad (2.46)$$

(c) *A multistage pipeline with composite stages*

Assuming the pipeline is made up of N composite stages and the i th stage is made up of M_i blocks, we have to minimize the overall energy

$$E(w_{1,1}, w_{1,2}, \dots, w_{1,M_1}, w_{2,1}, w_{2,2}, \dots, w_{2,M_2}, \dots, w_{N,1}, w_{N,2}, \dots, w_{N,M_N}, v) \\ = \sum_{i=1}^N \left\{ \sum_{j=1}^{M_i} [E_{ij}(w_{ij}, v)] \right\} \quad (2.47)$$

where the subscripts i and j refer to the i th pipeline stage and to the j th block within it, under the constraint that the overall delays of the various stages are all equal to a given value

$$D_i(w_{i,1}, w_{i,2}, \dots, w_{i,M_i}, v) = \sum_{j=1}^{M_i} [D_{ij}(w_{ij}, v)] = D_r. \quad (2.48)$$

The solution of the problem, as in the previous cases, can be found by using Lagrange multipliers and corresponds to the conditions

$$\frac{e_{ij}}{d_{ij}} \eta_{ij} = \frac{e_{i,k}}{d_{i,k}} \eta_{i,k} \quad \forall j, k = 1 \dots M_i \quad (2.49)$$

$$\sum_{i=1}^N \frac{e_{ij}}{d_{ij}} \eta_{ij} = \beta \quad \forall j = 1 \dots M_i \quad (2.50)$$

Again, the aggregate hardware intensity of the whole pipeline stage cannot be in general related to the hardware intensities of the underlying blocks, given that one has [158]

$$\frac{\partial E}{E} = - \sum_{i=1}^N \left\{ \sum_{j=1}^{M_i} \left[\frac{e_{ij}}{d_{ij}} \eta_{ij} \frac{\partial D_{ij}}{D} \right] \right\}. \quad (2.51)$$

However, when condition (2.49) is satisfied, from (2.51) one finds that the aggregate hardware intensity of the whole multistage pipeline is equal to

$$\eta = \sum_{i=1}^N \frac{e_{ij}}{d_{ij}} \eta_{ij} \quad \forall j = 1 \dots M_i \quad (2.52)$$

2.4.2 Practical Guidelines to Design Energy-Efficient Pipelines

The optimal criteria given by Zyuban and Strenski have two primary limitations: their hard-to-use coarse-tuning approach and the restricted assumption of energy and delay dependency among blocks/stages [30]. Indeed, the optimal criteria are difficult to apply, and they are more useful for the verification of optimality after performing the design. If the design is not optimal, the criteria may suggest modifications to energy, delay, hardware intensity, or supply voltage, but it is not immediately clear how to change each of these quantities [30].

The other limitation arises since these optimal criteria are derived assuming that changes in a particular block/stage do not affect the energy and delay of neighboring ones. While this assumption might be reasonable in coarse tuning of separate macro-modules, it is certainly inadequate in cascaded pipeline stages since the input (output) capacitances of each stage/block affect the performance of the preceding stage/block (of the stage/block itself). Therefore, the energy and delay dependencies between adjacent blocks/stages should be added to the previous derivations. However, due to the non-analytical form of these dependencies, their inclusion does not lead to an analytical solution [30].

To partially overcome the above issues, a thorough methodology consisting of several iterative steps has been proposed in [30]. This methodology targets the minimization of the energy of a multistage pipeline under a given delay constraint and without neglecting the mutual influence between the design of the various stages.

The convention adopted in [30] is to exclude the energy dissipation related to the (dis)charge of the input capacitance of a stage and to include that related with the (dis)charge of the output load capacitance.

The iterative procedure leading to the optimum designs of all the combined pipeline stages is based on the optimization of the stages themselves under various input/output capacitance conditions. Indeed, three different optimizations can be performed on a single stage:

- (1) the stage can be designed to achieve the minimum energy under a given delay constraint and with a fixed input and load capacitances. This is the problem discussed in the rest of this chapter and can be dealt with by resorting to simulation- or model-based optimizations (e.g., with generalized geometric programming). When exploring different delay constraints, an energy-efficient curve can be extracted and it reaches a well-defined minimum delay point corresponding to the Logical Effort design. This case is exemplified in the case of a 64-bit Kogge-Stone adder in Fig. 2.9 [30].
- (2) Given the convention adopted on input capacitance-related dissipation, the delay of the stage can be improved without worsening the energy by simply increasing the input capacitance as shown in the case of the 64-bit Kogge-Stone adder in Fig. 2.10 [30]. Obviously, such an increase negatively affects

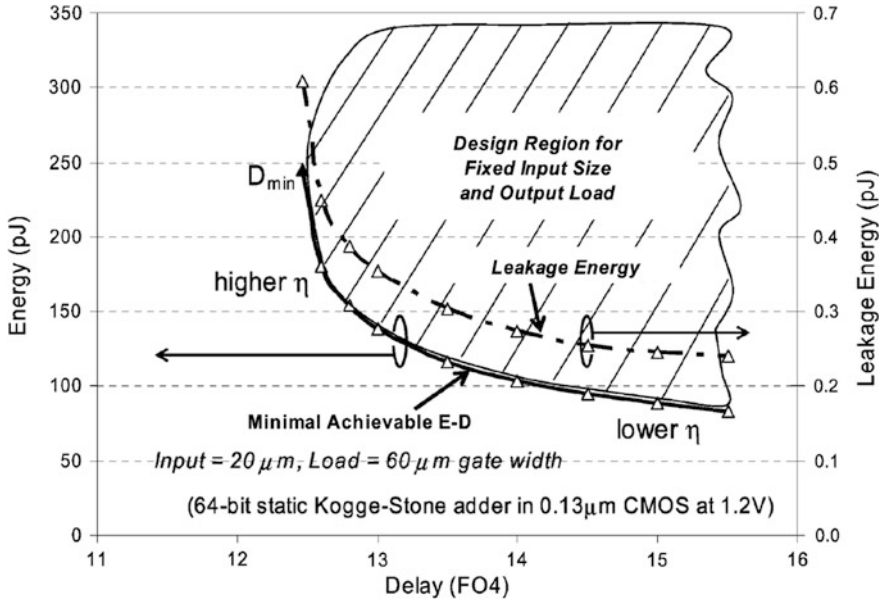


Fig. 2.9 64-bit Kogge-Stone adder: energy-delay optimization under fixed input capacitance and output load [30]

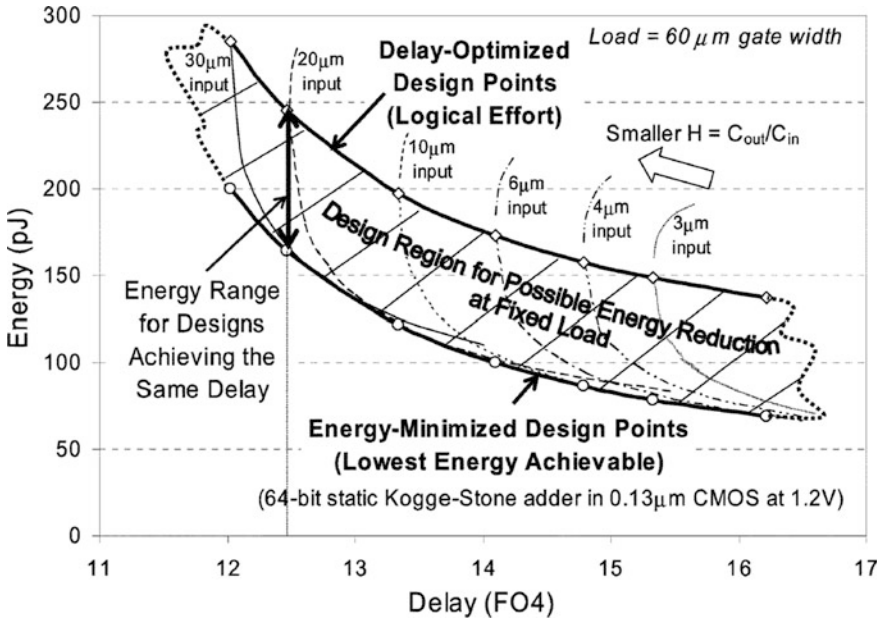
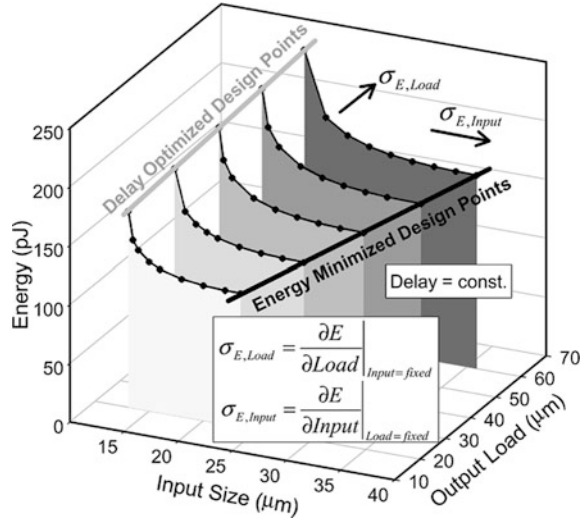


Fig. 2.10 64-bit Kogge-Stone adder: design region for possible energy-delay reduction under varying input capacitance and fixed output load [30]

Fig. 2.11 Optimized energy of a pipeline stage versus input capacitance under fixed load and versus load under fixed input capacitance [30]



the delay of the stage preceding the considered one given that its load increases.

- (3) Given the convention adopted on input capacitance related dissipation, the energy of the stage can be improved without worsening delay by simply increasing the input capacitance as shown in Fig. 2.10 [30]. Indeed, a larger input capacitance allows to reach the same delay with a smaller sizing (and hence a smaller dissipation) of the other gates within the stage. Obviously, such an increase negatively affects the energy of the stage preceding the considered one given that its load increases.

According to (2) and (3), for a fixed output load and a variable input capacitance an energy-efficient design region comes out and, as shown in Fig. 2.10, it is located between the minimum energy and minimum delay points [30]. Given a delay constraint, the maximum and minimum values for the input capacitance are found and correspond to the minimum energy and minimum delay point in Fig. 2.10.

The key for overall energy optimization is the analysis for each stage of the sensitivities of the optimized energy to the input capacitance C_{IN} , under a fixed output load C_L , and to the output load under a fixed input capacitance

$$\sigma_{E,C_{IN}} = - \frac{\partial E}{\partial C_{IN}} \Big|_{C_{IN} \text{ variable} / C_L \text{ fixed}} \quad (2.53)$$

$$\sigma_{E,C_L} = \frac{\partial E}{\partial C_L} \Big|_{C_L \text{ variable} / C_{IN} \text{ fixed}} \quad (2.54)$$

Indeed, in this way one can deal with the performance improvement of a stage when increasing its input capacitance and decreasing its output load, and the corresponding decrease in the performance of the preceding and succeeding stages.

The general trends are shown in Fig. 2.11, where it is shown that the sensitivity of the optimized energy of each stage to its C_{IN} under a fixed C_L asymptotically decreases for larger values of C_{IN} itself [30]. The maximum value of C_{IN} leading to the lowest stage energy corresponds to the minimum energy point in Fig. 2.10 and increases for larger C_L . On the contrary, the sensitivity increases when moving towards the minimum delay point (Logical Effort design) by decreasing the C_{IN} value. Again, the minimum delay point is achieved with a larger C_{IN} when C_L increases. Moreover, the optimized energy of the stage under a fixed C_{IN} is a nearly linear increasing function of C_L both when considering the minimum energy and minimum delay points.

It is easy to show that, when considering a multistage pipeline, the overall minimum energy is reached when the sensitivities of the energy of all stages to their input capacitances and output loads are all equal [30].

Basing on the above considerations, an iterative procedure to determine the energy-efficient sizing of a multistage pipeline comes out [30]:

- (1) a set of initial values for the capacitances at the boundaries between the various stages is chosen
- (2) the various stages are optimized for minimum energy given the delay constraint under a fixed input capacitance and output load (the capacitances at the boundary are fixed). This optimization can be performed with any of the methods discussed in this chapter.
- (3) The sensitivities in (2.53)–(2.54) are computed for each stage.
- (4) If the sensitivities are not equal for all the stages, the values of the capacitances at the boundaries between the various stages are properly updated and the procedure starts again at step 2. Otherwise the energy-efficient design for the multistage pipeline is found and the procedure comes to an end.

Appendix: Convex Optimization

When dealing with circuits of large size (e.g., a hundred design variables), non-linear programming is no longer able to ensure the reachability of the global optimum. Since non-linear programming and the related solving algorithms were necessary because of the complexity of quite accurate E-D models, the usage of the latter ones cannot be afforded anymore. On the contrary, the focus must be on the adoption of the most accurate possible E-D models leading to optimization problems that can be reliably solved (i.e., assuring the global optimum is found) in a feasible time.

Two classes of optimization problems that can be reliably and fast solved are “least-squares” problems, where there are no constraints and the objective function

is a sum of square terms, and “linear programming”, where both the objective and constraint functions are linear [17]. Both cases are special subclasses of “convex optimization”, where both the objective and constraint functions are convex [17]. In short, by defining the objective function $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ and the constraint functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ for $i = 1 \dots m$, a convex optimization problem is one of the form [17]

$$\begin{aligned} & \text{minimize} \quad f_0(x) \\ & \text{subject to} \quad f_i(x) \leq b_i \quad i = 1 \dots m \\ & \text{where} \quad f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \end{aligned} \quad (2.55)$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$.

There is in general no analytical formula for the solution of convex optimization problems. However, similarly to linear programming, there are very effective numerical methods for solving them, such as interior-point methods [17] or other custom methods [61]. These methods allow for easily solving problems with hundreds of variables and thousands of constraints on a current desktop computer, in a few tens of seconds or less. For instance, the method proposed in [61] is claimed to size circuit with a million gates in nearly 1 h. Furthermore, thanks to the properties of the above solving methods, the definition of practical design space bounds as well as that of the initial point from which to start the optimization, become irrelevant. Hence, it is apparent that as long as the optimization problem can be formulated in a convex form, the required computational effort is incomparably lower than that required in the previous cases. The other side of the coin is that the formulation itself requires a simplification of the E-D models that lowers the accuracy in their estimation. Nevertheless, this is the only feasible approach when the circuit size is large enough.

A class of optimization problems that really well suits the problem of digital gate sizing (e.g., to determine the energy-efficient designs as in our case) is that called geometric programming [15].

Given a vector $x = (x_1, x_2, \dots, x_n)$ of positive optimization variables (e.g., transistors sizes), a function g of the form

$$g(x) = cx_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \quad (2.56)$$

where $c > 0$ and $\alpha_i \in \mathbb{R}$, is called a “monomial” [16]. Monomials are closed under product, division, positive scaling, power, inverse. A sum of monomials, i.e.

$$f(x) = \sum_{k=1}^t c_k x_1^{\alpha_{1k}} x_2^{\alpha_{2k}} \dots x_n^{\alpha_{nk}} \quad (2.57)$$

where $c_k > 0$ and $\alpha_{ik} \in \mathbb{R}$, is called a “posynomial” [16]. Obviously, a monomial is also a posynomial. Posynomials are closed under sum, product, positive scaling, product and division by monomial and positive integer power. A “generalized posynomial” can also be introduced as a function that can be obtained also by taking the positive fractional power of posynomials or the maximum of some

posynomials [16]. Generalized posynomials are hence closed under sum, product, positive scaling, product and division by monomial, positive power and maximum.

A “(generalized) geometric programming”, (G) GP is an optimization problem with the form [16]

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 1 \quad i = 1 \dots m \\ & && g_i(x) = 1 \quad i = 1 \dots p \end{aligned} \tag{2.58}$$

where f_i are (generalized) posynomials and g_i are monomials. Several extensions, such as $f(x) < g(x)$, $f(x) < a$ or $g_1(x) = g_2(x)$ constraints, can be readily handled. It is also possible to maximize a nonzero monomial objective function, by minimizing its inverse (which is also a monomial).

GPs are not convex optimization problems. However, they can be converted into non-linear but convex optimization problems basing on a logarithmic change of variables and a logarithmic transformation of the objective and constraint functions [16, 17]. Variables x_i are substituted by $y_i = \log(x_i)$ and the log of posynomials and monomials is taken leading to the following optimization problem

$$\begin{aligned} & \text{minimize} && \log[f_0(e^{y_1}, e^{y_2}, \dots, e^{y_n})] \\ & \text{subject to} && \log[f_i(e^{y_1}, e^{y_2}, \dots, e^{y_n})] \leq 0 \quad i = 1 \dots m \\ & && \log[g_i(e^{y_1}, e^{y_2}, \dots, e^{y_n})] = 0 \quad i = 1 \dots p \end{aligned} \tag{2.59}$$

where the objective and inequality constraints are now smooth convex functions and the equality constraint are now affine functions, i.e. linear plus a constant.

Flip-Flop Design in Nanometer CMOS

From High Speed to Low Energy

Alioto, M.; Consoli, E.; Palumbo, G.

2015, XV, 260 p. 123 illus., 5 illus. in color., Hardcover

ISBN: 978-3-319-01996-3