Design and Optimization of Multi-Threshold CMOS (MTCMOS) Circuits

Mohab Anis, *Member, IEEE,* Shawki Areibi, *Member, IEEE,* and Mohammed Elmasry *Fellow, CAE*

***Abstract*— Reducing power dissipation is one of the most important issues in VLSI design today. Scaling causes subthreshold leakage currents to become a large component of total power dissipation. Multi-Threshold technology has emerged as a promising technique to reduce leakage power. This paper presents several heuristic techniques for efficient gate clustering in MTCMOS circuits by modeling the problem via Bin-Packing (BP) and Set-Partitioning (SP) techniques. The SP technique takes the circuit's routing complexity into consideration which is critical for Deep Sub-Micron (DSM) implementations. By applying the technique to six benchmarks to verify functionality, results obtained indicate that our proposed techniques can achieve on average 84% savings for leakage power and 12% savings for dynamic power. Furthermore, four hybrid clustering techniques that combine the BP and SP techniques to produce a more efficient solution are also devised. Ground bounce was also taken as a design parameter in the optimization problem. While accounting for noise, the proposed hybrid solution achieves on average 9% savings for dynamic power and 72% savings for leakage power dissipation at sufficient speeds and adequate noise margins.**

***Index Terms*— Multi-threshold voltage, low-power, leakage power, ground bounce.**

I. Introduction

With the advent of technology, the reduction of the supply voltage (V_{dd}) has become vital to reduce dynamic power and avoid reliability problems in Deep Sub-Micron (DSM) regimes. However, reducing V_{dd} alone causes serious degradation to the circuit's performance. One way to maintain performance is to scale down both V_{dd} and the threshold voltage V_{th}. However, reducing V_{th} increases the subthreshold leakage current exponentially. This problem escalates in DSM technologies. The leakage current $I_{leakage}$ can be approximately formulated as

$$I_{leakage} = I_0 e^{(V_{gs}-V_{th})/nV_T} \quad (1)$$

where $I_0 = \mu_0 C_{ox}(W/L){V_T}^2 e^{1.8}$, C_{ox} is the gate oxide capacitance, (W/L) is the width to length ratio of the leaking MOS device, μ_0 is the zero bias mobility, V_{gs} is the gate to source voltage, V_T is the thermal voltage which is about 26mV at T=300K, and n is the subthreshold swing coefficient given by 1+ $\frac{C_d}{C_{ox}}$ with C_d being the depletion layer capacitance of the source/drain junctions. From Eq. 1 it is evident that the leakage current is exponentially proportional to ($V_{gs} - V_{th}$). Therefore, leakage could be reduced by increasing V_{th} or reducing V_{gs}. Over the past decade, several techniques have been proposed to reduce leakage power during the standby mode by increasing V_{th}. In the Variable Threshold CMOS (VTCMOS) approach [1], the threshold voltage is controlled dynamically through varying the substrate bias voltage. In this scheme, all transistors have Low Threshold voltage (LVT) and the substrate bias is altered so as to: (1) compensate for V_{th} fluctuations in the active mode and accordingly minimize delay variations, and (2) reduce leakage current in the standby mode. Two drawbacks to the VTCMOS approach, are (1) since V_{th} is proportional to the square root of the substrate voltage, a large change in the later is thus required to change V_{th} by effective values, and (2) VTCMOS requires a triple-well structure as well as a charge-pump circuit to produce the substrate voltage.

Another technique is the Multi-Voltage CMOS (MVCMOS) scheme [2]. The MVCMOS technique employs LVT transistors whose gate voltages are driven in the sleep mode to larger than V_{dd} and smaller than V_{ss} for the PMOS and NMOS respectively. This creates a negative gate-to-source biasing (V_{gs}) and so reduce leakage current substantially (in agreement with Eq. 1). This scheme however will suffer from supply bounce problem, and in addition, it requires positive and/or negative charge pump(s).

A different technique to reduce leakage power is to set the primary inputs of a certain module to the $vector$ that best minimizes power in the sleep mode [3] [4] [5]. This technique takes advantage of the fact that leakage current of a CMOS gate can vary broadly with input combinations, due to the stacking effect and differences in the leakage of PMOS and NMOS devices. Drawbacks to this technique include: (1) the stacking effect is not very effective in circuits with deep logic since on average the greater the number of logic levels, the less sensitive leakage power is to primary input combinations, and (2) exhaustive circuit simulations that are controlled using random [3] or generic [4] techniques are made to search for the "best" vector.

Over the past few years, a technique that has emerged increasingly popular is the use of the Multi-threshold CMOS (MTCMOS) technology [6]. MTCMOS circuits reduce leakage power during the standby mode, while attaining high speed in the active mode. The MTCMOS can be either implemented statically or dynamically. In the static approach, devices switching in the critical path are assigned LVT thus attaining the circuit's high speed, while those not in the critical

Manuscript received May 22, 2002; revised April 17, 2003. This paper was recommended by Associate Editor Farid Najm.

M. Anis and M. Elmasry are with the Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario.

S. Areibi is with the School of Engineering, University of Guelph, Guelph Ontario. His research is supported by an NSERC operating grant.

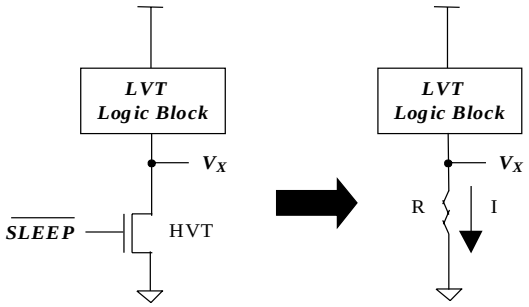
path have High Threshold voltage (HVT) to minimize leakage power [7] [8]. This technique requires accurate assignment of LVT and HVT, but has the advantage of preserved speed. In the dynamic approach, the logic gates are implemented using LVT devices and are connected to a virtual ground line. This line is linked to the main ground rail through an HVT transistor, called a "sleep transistor" [9]. The sleep transistor is controlled by a $SLEEP$ signal used for active/standby mode control ($SLEEP$=1,0 during standby and active modes respectively) (Figure 1(a)). Utilizing LVT transistors permits operating at low supply values with sufficient speed during the active mode. In the standby mode, the $SLEEP$ signal is activated to turn off the HVT device. This will cause the virtual ground line to float and so limit the leakage current to that of the HVT transistor, which is very small. The design cycle is usually short, but at the expence of a slight speed loss. Proper sleep transistor sizing is therefore a key issue that affects the performance as well as the dynamic and leakage powers of the entire circuit.

In this paper we introduce two models and several hybrid heuristic techniques that cluster logic gates at a fixed sleep transistor size, which will prove to be power efficient compared to the literature ([9], [10]) while maintaining adequate performance. Furthermore, noise associated with the virtual ground rail is taken as a design criterion in later sections. The paper starts with a brief review on the concept of sleep transistors. Section III-A explains the technique we used for calculating the size of the sleep transistor. Section 3.2 introduces novel techniques used for gate clustering and assignment. Our results are summarized in Section IV-D. In Section V, four hybrid clustering techniques are proposed, and compared. Section VI introduces the noise on the virtual ground rail as a design criterion and results for the techniques are shown in Section VII when noise is taken into account. The paper concludes with some comments on how the different clustering methods performed and performance enhancements attributed to the proposed techniques.

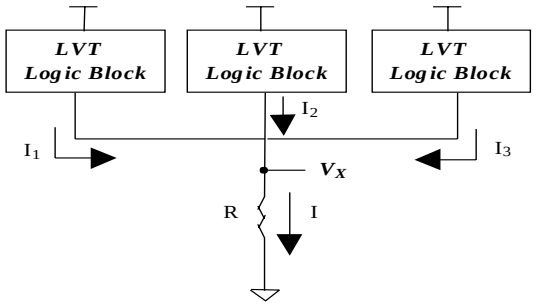
II. Background

During the active mode, the sleep transistor could be realized as a resistor (**R**) as shown in Figure 1(a) [11]. This generates a small voltage drop V_X equal to ($I \times R$), where I is the current flowing through the sleep transistor. The voltage drop across R has two effects, firstly it reduces the gate's driving capability from V_{dd} to V_{dd}-V_X and secondly it causes the threshold voltage of the LVT pull-down devices to increase due to the body effect [4] [5]. Both effects degrade the speed of the circuit. Therefore, the resistor should be made small and consequently the size of the sleep transistor large. This comes at the expence of extra area and power. On the other hand, if the resistor is made too large (i.e. the sleep transistor is sized small), the circuit speed will degrade. Therefore, a trade-off exists between achieving sufficient performance and low power values. This trade-off will become even more evident in the DSM regime. In DSM technologies, the supply voltage is scaled down aggressively, causing the resistance of the sleep transistor to increase dramatically, requiring even larger size sleep devices. This will cause leakage and dynamic power to significantly mountain in the standby and active modes respectively. Therefore, an important design criterion is sizing the sleep transistor to attain sufficient performance. In other words, the current "I" flowing through the sleep transistor must be satisfactory to achieve the required speed.

The worst case design scenario takes place if all the gates supported by the sleep transistor are simultaneously switching in time (Figure 1(b)). The sleep transistor exhibits maximum current when (I=I_1+I_2+I_3)(Case I). In this case, the sleep transistor is sized up to contain the high current. If the gates are discharging mutually exclusive, the sleep transistor is sized according to the maximum current of the mutually exclusive discharging gates ($I = \max\{I_1, I_2, I_3\}$) (Case II). The sleep transistor is a lot smaller in this case. If a current-time graph is constructed of the discharged currents, I_1, I_2 and I_3 overlap in time in Case I. On the other hand, no overlap in time occurs for Case II. An intermediate case occurs when the discharged currents "partially" overlap, if the LVT Logic Blocks have slightly different discharge times.



(a) Sleep Transistor modeled as resistor



(b) Worst case discharging scenario

Fig. 1. Sleep Transistor in MTCMOS Circuits

A single sleep transistor to support the whole circuit was proposed in [9]. The logic gates ground rails were connected to a virtual ground rail which has potential slightly higher than ground. The real and virtual ground rails are then linked by the sleep transistor. In another work [10], the sleep transistor was sized according to an algorithm based on mutual exclusive discharge pattern. In [10], cascaded gates are clustered together because simultaneous current discharge can never take place. This approach may be efficient for balanced circuits with tree

configurations, where mutually exclusive discharging gates are easily detected. However, this methodology would not be as efficient for circuits with complicated interconnections and unbalanced structures. Sleep transistor assignments can therefore be wasteful, and would cause dynamic and leakage power to rise. Finally, the sets of sleep transistors in [10] are merged into a single large sleep transistor to accommodate the whole circuit as in [9]. In addition to these drawbacks, sharing a single sleep transistor for the whole circuit would increase the interconnect resistance for distant blocks. As a result, the sleep transistor would be sized even larger than expected to compensate for the added interconnect resistance. Excessively large sleep transistors again augment dynamic and leakage power as well as area. This drawback would be even more severe in DSM regimes, where interconnects would have a large impact on the circuit performance [12]. Our proposed methodologies presented in the upcoming sections solve this problem, and not only cluster gates with exclusive discharge patterns, but with "partially" overlapping discharged currents as well. The first step in our techniques is to calculate the size of the sleep transistor.

III. Proposed Techniques

A. Sizing The Sleep Transistor

To estimate the size of the sleep transistor, the delay of a single gate (τ_d) at the absence of a sleep transistor can be expressed as [9] [11]

$$\tau_d \propto \frac{C_L V_{dd}}{(V_{dd} - V_{tL})^{\alpha}} \tag{2}$$

where C_L is the load capacitance at the gate output, V_{tL} is the LVT=350mV, V_{dd}=1.8V and α is the velocity saturation index which is equal to $\approx$ 1.3 in 0.18μm CMOS technology. In the presence of a sleep transistor, the delay of a single gate τ_d^{sleep} can be expressed as

$${\tau_d}^{sleep} \propto \frac{C_L V_{dd}}{(V_{dd} - V_X - V_{tL})^{\alpha}} \tag{3}$$

where V_X is the potential of the virtual ground. Assuming the circuit could tolerate a 5% degradation in performance due to the presence of the sleep transistor, therefore

$$\frac{\tau_d}{\tau_d^{sleep}} = 95\% \tag{4}$$

Substituting for τ_d and τ_d^{sleep}, and assuming $\alpha = 1$ for simplicity, we get

$$1 - \frac{V_X}{(V_{dd} - V_{tL})} = 95\% \tag{5}$$

Therefore V_X can be formulated as

$$V_X = 0.05(V_{dd} - V_{tL}) \tag{6}$$

The current flowing through the "linearly-operating" sleep transistor is expressed as:

$$\begin{aligned} I_{sleep} &= \mu_n C_{ox} (W/L)_{sleep} [(V_{dd} - V_{tH}) V_X - {V_X}^2/2] \\ &\approx 0.05 \mu_n C_{ox} (W/L)_{sleep} (V_{dd} - V_{tL})(V_{dd} - V_{tH}) \end{aligned} \tag{7}$$

where μ_n is the N-mobility, C_{ox} is the oxide capacitance and V_{tH} is the HVT=500mV. The size of the sleep transistor can be therefore expressed as

$$(W/L)_{sleep} = \frac{I_{sleep}}{0.05 \mu_n C_{ox} (V_{dd} - V_{tL})(V_{dd} - V_{tH})} \tag{8}$$

Values for I_{sleep} and consequently $(W/L)_{sleep}$ are chosen to exhibit low power dissipation. Thus, an optimization problem exists to find the value for I_{sleep} and consequently $(W/L)_{sleep}$ to dissipate minimal dynamic and leakage power. This will be illustrated in Section IV-B. For the time being, and to illustrate the basic idea behind the proposed techniques, a value for I_{sleep} is chosen to be 250μA, leading to a $(W/L)_{sleep} \approx 6$ (Eq. 8) for 0.18μm CMOS technology. This constant size $(W/L)_{sleep}$ = 6 will be first used for illustrating the first two proposed methodologies i.e Bin-Packing (BP) and Set-Partitioning (SP) techniques. To ensure correct functionality, agreeable delay, power and leakage values to analytical calculations were verified for the LVT and HVT HSPICE models. Leakage current increases by an order of magnitude for every 85mV reduction in V_{th}. Furthermore, due to the approximation of the velocity saturation index α from 1.3 to 1 in our analysis, there is a small difference between the simulated current and the modeled current from Eq. 8 which is in the order of 10%. This current difference may slightly shift the results later. However, the proposed clustering technique will not change and substantial leakage power savings are still achieved. More over, more accurate results could be obtained by taking the simulated current values instead of the modeled value in Eq. 8.

IV. Proposed Clustering Technique

To illustrate our techniques, six benchmarks are used as test vehicles; a 4-bit Carry Look Ahead (CLA) adder, a 32-bit priority checker, a 6-bit array multiplier design, a 4-bit ALU/Function Generator (74181 ISCAS-85 benchmark), a 32-Single Error Correcting circuit (C499 ISCAS-85 benchmark), and finally a 27-bit Channel Interrupt Controller (CIC) (C432 ISCAS-85 benchmark). These benchmarks have been chosen to offer a variety of circuits with different structures employing various gates with different fanouts. The 4-bit CLA adder will be first used to demonstrate the proposed techniques. Results pertaining to all other benchmarks will be provided in section IV-D.

Figure 2 shows a schematic diagram of the CLA adder, which consists of 28 gates (G_1-G_{28}). All gates are implemented in 0.18μm CMOS technology. To illustrate our proposed technique, a preprocessing stage of gate currents is described in the next section. This stage will be utilized in solving the BP problem and later in solving the SP problem.

A. Preprocessing of Gate Currents

The main objective of the preprocessing stage is to group gates into sub-clusters such that their combination would not exceed the max current of any gate within the cluster.

All the gates used in the implementation of the benchmarks were based on the 0.18μm Standard Cell Library by $Virtual$

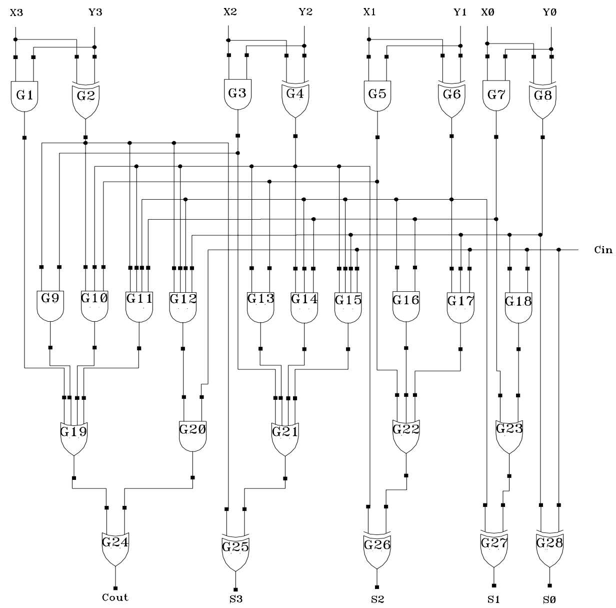


Fig. 2. 4-bit Carry Look-Ahead Adder

Silicon Technology Inc. using the 0.18μm *TSMC* Process. In this library, 250 logic gates are defined. The propagation delay of each of these logic gates is documented. For each logic gate, different propagation delays are documented according to: (1) the input vector applied to the gate, (2) the kind of transition occuring at the input of the gate, (3) the amount of fanout associated with the gate, and (4) whether the signal at the output of the gate is falling or rising. In summary, the standard cell library carries all the gate information. The documented propagation delays for the Standard Cell Library have been verified through HSPICE simulations under the same environment (i.e. same 0.18μm CMOS technology provided by *TSMC*; V_{dd}=1.8V and temperature=25^{o}C).

In our analysis, only the propagation delay for an output falling edge signal is taken into consideration, because it takes into account the current flowing through the sleep transistor.

The tested benchmarks are composed of gates having different fanouts. Each of the gates composing a given benchmark is mapped to the documented Standard Cell Library according to its functionality and fanout. For each gate, all input combinations are applied (for example: 00,01,10,11 for a 2-input XOR gate), and the highest discharging current at the output of every gate is monitored (worst case discharge current) while taking the gate's fanout into consideration. The discharge current as well as the short-circuit currents are monitored because this is the current that flows through the sleep transistor and eventually *ground*. The probability that discharging takes place (switching activity) at the output of each gate is calculated and multiplied by the corresponding discharge peak current. This gives an *expected* discharge current value. The switching activity of a gate is computed by multiplying the probability that the output of the gate will be at zero, by the probability it will be at one [13]. If the switching activity is not accounted for, the design problem would be very pessimistic and the sleep transistor will be oversized, causing substantial increase in leakage and dynamic power dissipation as well as die size. It is very unlikely that the clustered gates would have their worst case current discharge at the same time. This has been deduced by exhaustively applying all input vectors to the CLA adder benchmark. The monitored current is composed of the discharge and the short-circuit currents that take place during switching. Sleep transistors should be sized to also accommodate the short-circuit currents, otherwise speed will degrade.

The peak current value and time at which the switching occurs as well as its duration are monitored. The time the switching takes place depends on the gate propagation delay and input pattern, while the current duration depends on the slope of the input signal as well as the fanout of the gate. The larger the input slope and/or gate fanout, the longer the switching duration. The discharge current of each gate takes a triangular shape, whose peak occurs at a time equal to the gate delay, and spans a time, mainly function in the fanout of the gate. Since the switching activity of a gate is a constant number, multiplying it by the triangular shaped discharge current would also produce a triangular shape spanning the same time duration, but with a smaller peak value.

To facilitate vector comparisons and to offer an automated design environment, every discharge current at the output of a gate is represented by a vector. The time axis is divided into time slots each equal to 10psec as shown in Figure 3. A time slot of 10psec is sufficient in 0.18μm CMOS technology to offer relatively good accuracies. Each time slot holds a value that represents the magnitude of the discharge current at that specific time which constitutes an element in the vector. In order to illustrate this idea, Figure 3 shows a 2-input AND gate (G1) with a fanout of 2 driving a 2-input OR gate (G2) with a fanout of 4. Furthermore, a load of 6fF is applied to the outputs of each circuit contributing for wiring capacitance. The discharge currents of G1 and G2 (I_1 and I_2) are presented as a vector.

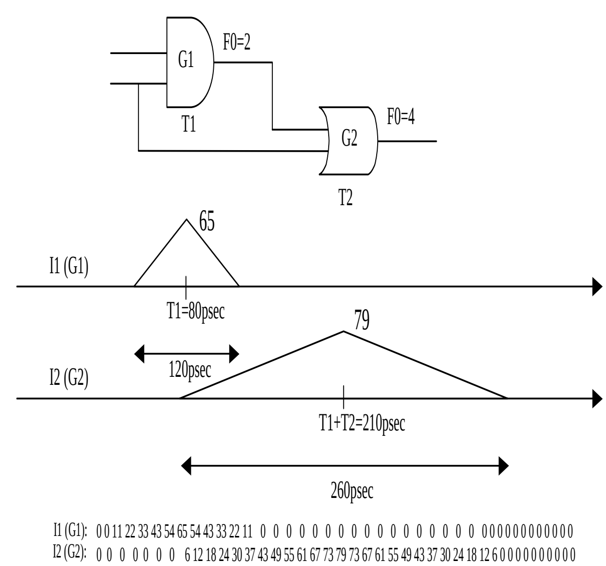
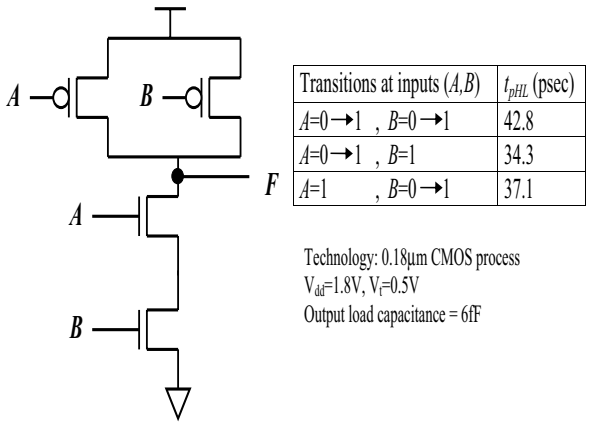


Fig. 3. Timing Diagram

Each element in the vector presents the magnitude of current at this 10psec time slot. The peak of the discharge current for G1 occurs at the gate's delay time ($T_1 = 80psec$), while the discharge current I_2 occurs at time ($T_1 + T_2 = 210psec$), because G2 will not discharge until G1 discharges. The peak currents of gates G1 and G2 are $65\mu A$ and $79\mu A$ respectively. The triangular shaped currents are converted into vectors as shown in Figure 3. Since G2 has a large fanout of 4, the duration of the discharge current is long (260 psec), while the duration of the discharge current in G1 is short due to the small fanout of 2 (120 psec). Therefore, for every gate in the circuit, a vector is constructed that carries information about the delay of the gate (when the peak occurs), the fanout of the gate (the duration at which the current lasts) and the magnitude of the current in each time slot. By constructing a vector for each gate, a series of vectors (28 in this case) are produced, that carry information about the whole circuit.

1) Trapezoidal Current Discharge: In Figure 3, the current I_1 is assumed to be discharged at a time equal to the gate G1 average delay; T_1 (provided that G1 is a primary output gate in the circuit). Similarly, current I_2 is assumed to be discharged at time $T_1 + T_2$, where T_2 is the gate G2 average delay. However, the delay of any gate, and consequently the delay of the whole circuit, changes with the input vectors. For example, consider the 2-input NAND gate in Figure 4. The high-to-low propagation delay (affected by sleep transistor) varies with different changes in the input vector[1].



Transitions at inputs (*A*,*B*)	t_{pHL} (psec)
A=0→1 , *B*=0→1	42.8
A=0→1 , *B*=1	34.3
A=1 , *B*=0→1	37.1

Technology: 0.18µm CMOS process
V_{dd}=1.8V, V_t=0.5V
Output load capacitance = 6fF

Fig. 4. 2-input NAND example

Thus, a method should be provided that insures that current discharge is taken into account with any input vector combination. Accounting for the discharge current over all input vector combinations guarantees that the sleep transistor would be sized properly, and that the circuit would meet the target performance. We therefore apply, what we call the min/max technique. The objective of this technique is to determine the earliest and latest time that a current discharge takes place at the output of a certain gate. Relating to the 2-input NAND example in Figure 4, t_{min}=34.3 psec (earliest time), while t_{max}=42.8psec (latest time), where t_{min} and t_{max} are the minimum and maximum high-to-low propagation delays of the NAND gate. The min/max technique is further explained through the following general example.

[1]Explanation of the propagation delay values can be found in [14].

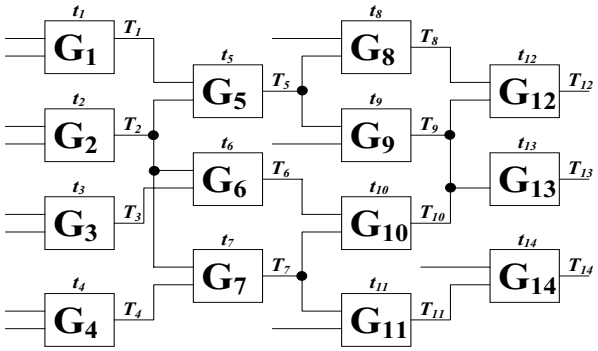


Fig. 5. Random logic network example

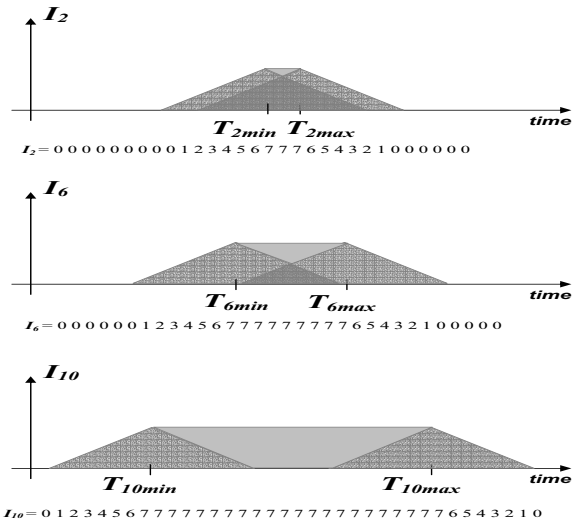


Fig. 6. Discharge currents in the random logic example

Consider the random logic circuit in Figure 5. We define a parameter T_k, which denotes the accumulative delay at the output of gate G_k. The accumulative delay of gate G_k has a minimum value $T_{k_{min}}$ and a maximum value $T_{k_{max}}$, whose values depend on the accumulative delays of the preceding gates. For a primary output gate G_k such as G1, G2, G3 and G4, $T_{k_{min}}$ is equal to $t_{k_{min}}$, where $t_{k_{min}}$ is the minimum intrinsic gate delay of G_k. On the other hand $T_{k_{max}}$ is equal to $t_{k_{max}}$, where $t_{k_{max}}$ is the maximum intrinsic gate delay of G_k.

As an example, for a primary output gate like G2, $T_{2_{min}}$=$t_{2_{min}}$, while $T_{2_{max}}$=$t_{2_{max}}$. If G2 was a 2-input NAND gate for example, $T_{2_{min}}$=$t_{2_{min}}$=34.3psec and $T_{2_{max}}$=$t_{2_{max}}$=42.8psec (Figure 4).

For non-primary output gates, $T_{k_{min}}$ and $T_{k_{max}}$ would need to consider the minimum and maximum delays for each input to that non-primary output gate. For example, gate G6 is a non-primary output gate fed by gates G2 and G3. Therefore $T_{6_{min}}$ and $T_{6_{max}}$ can be written as:

$$T_{6_{min}} = min\{(T_{2_{min}} + t_{6_{min}}), (T_{3_{min}} + t_{6_{min}})\} \quad (9)$$

$$T_{6_{max}} = max\{(T_{2_{max}} + t_{6_{max}}), (T_{3_{max}} + t_{6_{max}})\} \quad (10)$$

Similarly, the accumulative delay T_{10} of the non-primary output gate G10 is:

$$T_{10_{min}} = min\{(T_{6_{min}} + t_{10_{min}}), (T_{7_{min}} + t_{10_{min}})\} \quad (11)$$

$$T_{10_{max}} = max\{(T_{6_{max}} + t_{10_{max}}), (T_{7_{max}} + t_{10_{max}})\} \quad (12)$$

In general, the accumulative delay T_k of gate G_k is expressed as:

$$T_{k_{min}} = min\{(T_{i_{min}} + t_{k_{min}}), (T_{j_{min}} + t_{k_{min}}), ...\} \quad (13)$$

$$T_{k_{max}} = max\{(T_{i_{max}} + t_{k_{max}}), (T_{j_{max}} + t_{k_{max}}), ...\} \quad (14)$$

given that output of gates G_i, G_j, ... are inputs to G_k.
Equations 13 and 14 can be rewritten as:

$$T_{k_{min}} = min\{(T_{i_{min}} + t_{k_{min}})\} \forall i = 1, 2, 3, ...n \quad (15)$$

$$T_{k_{max}} = max\{(T_{i_{max}} + t_{k_{max}})\} \forall i = 1, 2, 3, ...n \quad (16)$$

where n is the number of inputs to gate k.

The general expression for the accumulative min and max delays $T_{k_{min}}$ and $T_{k_{max}}$ in Equations 13 and 14 are valid also for the special case that gate G_k is a primary output gate. In that case, $T_{i_{min}}$, $T_{j_{min}}$...., and $T_{i_{max}}$, $T_{j_{max}}$...., are equal to $zero$, leading to $T_{k_{min}}$=$t_{k_{min}}$ and $T_{k_{max}}$=$t_{k_{max}}$.

In Figure 3, the peak discharge current is assumed to occur at a single time value. Now that the delay of a gate changes with the input vector, the discharge current must be taken into account during the time period from $T_{k_{min}}$ to $T_{k_{max}}$ for each gate G_k. This guarantees that regardless of the input vector, the discharge current is taken into account, and the speed of the circuit is attained. To illustrate this, Figure 6 shows a rough diagram of the discharge currents for gates G2,G6, and G10 in Figure 5.

Figure 6 is for illustration purposes only, and does not take into account the actual discharge current values or how fanout changes the duration of discharge. The triangular shaped discharge current previously shown in Figure 3 sweeps the time range from $T_{k_{min}}$ to $T_{k_{max}}$. Therefore, the discharge current is no longer modeled as a triangle, but as a trapezoid which takes into consideration the variation in delay due to changes in the input vector. The corresponding vectors for currents I_2, I_6 and I_{10} are also shown in Figure 6. It is interesting to note from Figure 6 that the first stage gates (primary output gates); G1, G2, G3 and G4 would have discharge currents that span short time durations. As we move deeper into the circuit, the discharge current would span longer time durations for gates located in later stages. This is attributed to the increase in accumulated delay in gates located in later stages.

This method insures that current discharge is taken into account with any input vector combination. As previously explained, accounting for the discharge current over all input vector combinations guarantees that the sleep transistor would be sized properly, and that the circuit would meet the target performance. Although, the approach presented may seem to pessimistically model the discharge currents, which may increase the size or number of sleep transistors, it actually proves useful in accounting for two important factors:

- *Intrinsic gate and interconnect delays*: The delay of a circuit varies with variation in the intrinsic gate delay as well as the interconnect delay. These variations are more severe in the deep-submicron regime. Intrinsic gate delays vary due to variations such as threshold-voltages, transistor dimensions, doping concentrations, input signal slope variations [15]. On the other hand, interconnect delays may vary due to variations in wire dimensions and coupling noise [16]. Accounting for the discharge current from $T_{k_{min}}$ to $T_{k_{max}}$ as illustrated in Figure 6 would include discharge currents even if variation in circuit delay takes place. This guarantees that performance is not degraded with any delay variations.
- *Glitching currents*: Glitching currents usually arise at the output of a gate whose inputs do not arrive at the same time. Referring to Figure 3, glitching currents may arise at the output of gate G2 due unbalanced delay paths for the inputs. If these glitching currents are not taken into account when sizing the sleep transistor, performance would be affected at the time these glitching current occur. In our technique, the discharge current is accounted for, from the minimum delay of all the inputs $T_{k_{min}}$ to the maximum delay of all inputs $T_{k_{max}}$. This insures that any glitching currents are taken into consideration, and thus sleep transistors are sized properly to fulfill the target performance.

2) Preprocessing heuristic: Figure 7 illustrates the used preprocessing heuristic that forms a set of sub-clusters of gates that when combined would not exceed the maximum current of any gate within the cluster [17].

The preprocessing algorithm first initializes the current vectors of all the gates as described in Figure 6. At the beginning of the preprocessing algorithm all gates are set free to move to any newly created cluster. Once a gate is collapsed into a cluster it is locked and is unable to participate in the formation of new clusters. The criteria used to append a gate to a cluster is based on the maximum current capacity of the current cluster. As shown in step (3) of the algorithm, a cluster is initially seeded with a free gate after which all other gates are appended to the cluster (according to the criteria explained above). Once a gate is appended to a cluster it is locked and the cluster information (in terms of the number of gates, current, maximum current) is updated. The preprocessing algorithm terminates when it is not possible to append any further gates to a cluster. Table I shows the

```
PREPROCESSING HEURISTIC
1. Initialize current vectors
2. Set all Gates free; to move to subcluster;
3. For  all gates in circuit
        If gate G_i is not clustered yet
          assign gate G_i to new cluster C_k
          update cluster current vector
          calculate max current,start,end time
        End If
        For  all other gates in circuit
          If (gate G_j is not clustered yet)
            add current of gate G_j to cluster C_k
            If (combination ≤ max current)
              append gate to cluster
              update cluster info
              set gate G_j locked in cluster C_k
            End If
        End For
    End For
4. Return all clusters formed.
```

Fig. 7. Heuristic for Forming Sub-clusters

results of applying the preprocessing heuristic to the 4-bit carry look-ahead adder that was presented in Figure 2 where 14 sub-clusters are formed. For example the second column in Table I (I_{EQ_2}) represents a sub-cluster formed by combining Gates G_3,G_4, G_{27} and G_{28} which has a maximum current of 80μA of the partially overlapped discharging gates (I_{G3}^{max} = 65μA, I_{G4}^{max} = 80μA, I_{G27}^{max} = 34μA and I_{G28}^{max} = 30μA). $I_{overlap}^{2} = max\{I_{G3}^{max}, I_{G4}^{max}, I_{G27}^{max}, I_{G28}^{max}\}$=$I_{G4}^{max}$=80$\mu$A. The objective is then to group as much current (as many gates) as possible without exceeding the current limit of the sleep transistor (250μA), while minimizing the number of sleep transistors used. This is shown in Table II of Section IV-B. This problem presentation is analogous to the Bin-Packing problem in operations research.

B. The Bin-Packing Technique

The Bin-Packing (BP) problem [18] can be described as follows. Given n $items$ (a set of equivalent $currents$ in this case) and m $bins$ ($sleep$ $transistors$ in this case), with

$$I_{EQ_j} = \text{equivalent } current \text{ of gate } j,$$
$$I_{max} = capacity \text{ of each } sleep\ transistor = 250\mu\text{A}$$

The objective is to assign each I_{EQ} to one bin so that the total current in each bin does not exceed I_{max} and the number of bins used is minimized. It is important to notice that the peak current of a combination of logic gates "subcluster" (as we will describe later on) is directly related to the peak current of the individual logic gates (Tables I and II).

The mathematical formulation of the problem is as follows

$$Minimize\ z = \sum_{i=1}^{m} y_i \tag{17}$$

Subject to

$$\sum_{j=1}^{n} I_{EQ_j} x_{ij} \leq I_{max} y_i, \qquad i \in \{1, ..., m\},$$
$$\sum_{i=1}^{m} x_{ij} = 1, \tag{18}$$

where

$$y_i = \begin{cases} 1, & \text{if bin } i \text{ is used} \\ 0, & \text{otherwise} \end{cases} \qquad x_{ij} = \begin{cases} 1, & \text{if items } j \in \text{bin } i; \\ 0, & \text{otherwise} \end{cases}$$

This model is a pure Binary Integer Programming problem (BIP). The objective function to be minimized; z, is analogous to the minimum number of sleep transistors used. y_i is analogous to the sleep transistors available. x_{ij} takes a value of "1" if current I_{EQ_j} is assigned to bin i. CPLEX 7.5; a commercial ILP solver, was used to solve this BP problem, to determine which currents should be grouped together, and to which sleep transistor they are assigned. A summary of the current assignments is shown in Table II.

It is clear from Table II that three sleep transistors will be needed to contain all the gates in the circuit ($z = 3$). It should be noted that the total current of any cluster must never exceed the maximum current limit of the sleep transistor, which is 250μA.

Now that the basic idea for the BP technique has been illustrated through I_{sleep}=250μA, we should find the optimum I_{sleep} value which dissipates the least dynamic and leakage power. Six values for I_{sleep} are considered; I_{sleep}= 150, 200, 250, 300, 350, 400μA. Table III shows the different values for I_{sleep} and the corresponding $(W/L)_{sleep}$ and W_{sleep} using Eq. 8, where L_{sleep} is taken as 180nm for a 0.18μm CMOS technology.

The six benchmarks previously mentioned were simulated with different sleep transistor sizes in Table III. In each case, both the dynamic and leakage power ($P_{dynamic}$ and $P_{leakage}$) are calculated, and a Figure Of Merit (FOM) is generated which is the product of $P_{dynamic}$ and $P_{leakage}$.

Dynamic power is calculated during the active mode (SLEEP signal controlling sleep transistor=0 Fig. 1(a), ie. sleep transistor is ON). The current drawn from the supply is monitored for all generated input vectors and averaged, and finally multiplied by V_{dd} to produce the average $P_{dynamic}$. The dynamic power dissipated due to the *on* and *off* switching of the sleep transistors is ignored, since the standby/sleep time period is significantly longer than the active period [19]. On the other hand, leakage power is calculated during the standby mode (SLEEP signal=1, sleep transistor is OFF). All input vector combinations are applied to the circuit inputs, and the measured leakage current is monitored for each input vector, and then is finally averaged. The average leakage current is then multiplied by V_{dd} to get the average $P_{leakage}$. The reason leakage power was averaged, is because $P_{leakage}$ varies with the input vector [3].

This FOM is plotted for different sleep transistor sizes for all benchmarks. The sleep transistor size which achieves

minimum FOM is recorded. Figure 8 plots the normalized FOM ($P_{dynamic} \times P_{leakage}$) versus I_{sleep} (W_{sleep}) for the 6-bit Multiplier. The FOM curve is normalized to its minimum value over different I_{sleep} values. A W_{sleep} of $\approx$ 1.32μm ($I_{sleep}\approx$300μA) achieves minimum FOM for the 6-bit Multiplier case. Furthermore, Figure 8 also plots the normalized $P_{leakage}$ over different sleep transistor sizes. $P_{leakage}$ curve is normalized to its minimum value over different I_{sleep} values. Minimum $P_{leakage}$ is achieved at the same sleep transistor size ($W_{sleep}\approx$ 1.32μm) as the FOM. This is because dynamic power is not effected to the first order by the choice of sleep transistor size.

Since at a sleep transistor size W_{sleep} = 1.32μm, $P_{leakage}$ and FOM are minimized, the values of $P_{dynamic}$ and $P_{leakage}$ at W_{sleep} = 1.32μm are recorded. This cycle is repeated for the remaining five benchmarks. From Figure 8 it can be seen that when W_{sleep} (I_{sleep}) takes a very small value, the number of sleep transistors (ST) increases (ST=6 for W_{sleep}=0.66μm), and consequently both $P_{dynamic}$ and $P_{leakage}$ are augmented. As I_{sleep} takes higher values, more gates can be confined within a sleep transistor. Even though the sleep transistor size increases, the large reduction in sleep transistor number, causes $P_{dynamic}$ and $P_{leakage}$ to drop. Eventually, a stage is reached that as I_{sleep} increases, the saving in ST number is reduced relative to the increase in W_{sleep}. This will cause $P_{dynamic}$ and $P_{leakage}$ to augment again. The curve shown in Figure 8 thus has a point where the $P_{dynamic}$,$P_{leakage}$ product has a minimum value. Furthermore, minimum $P_{leakage}$ is achieved at that same point. This is because $P_{leakage}$ is directly proportional to W_{sleep}, unlike $P_{dynamic}$. It should be also noted that based on the minimum FOM value achieved for each benchmark, W_{sleep} (I_{sleep}) varies from one benchmark to another depending on structure and topology of the circuit. For example, the 4-bit ALU benchmark has a minimum FOM at I_{sleep}=200μA (W_{sleep}=0.88μm). This is different than the 6-bit Multiplier case at hand, where minimum FOM is achieved at I_{sleep}=300μA (W_{sleep}=1.32μm). The values for W_{sleep} achieving minimum FOM for the benchmarks are summarized in Table IV.

Keeping the 5% speed degradation as a comparison basis, the BP technique is compared to [9] and [10]. The operational frequency is 500MHz, and a load of 6fF is applied to the outputs of each gate in a benchmark.

The results are mentioned in Section IV-D and summarized in Table IV (Normalized to [9]). The BP technique proves to attain high dynamic and leakage power savings and in particular achieving on average 95%, 85% leakage savings compared to [9] and [10] respectively. In addition to the reduction in leakage power, the Bin-Packing method reduces dynamic power by an average of 17% with respect to [9] and 14% compared to [10]. Leakage power has been calculated in the standby mode when the sleep transistors are off ($SLEEP$=1) and inputs are inactive.

The BP technique is particularly efficient when it is applied to small circuits that have unbalanced structures. One limitation is that the BP technique does not take the physical locations of the gates on the chip into consideration. For larger circuits this might cause two gates located far apart to be clustered together which will augment the routing complexity of the circuit, as discussed earlier. The Set-Partitioning technique solves this problem, and consequently reduces the routing complexity of the circuit, unlike [9] and [10].

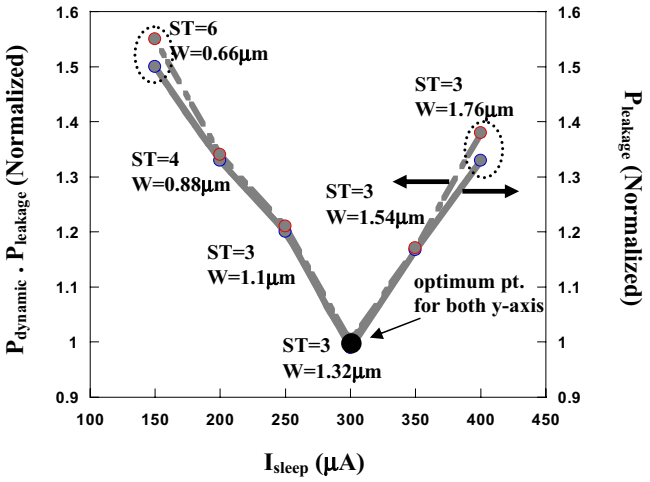


Fig. 8. Different I_{sleep} values for the 6-bit Multiplier

C. The Set-Partitioning Technique

The Set-Partitioning (SP) problem [18] can be described as follows: Similar to the BP problem, m currents (gates) are arranged into groups such that each gate is included only once in a cluster. A cost function; c_j is associated with each group j; S_j. The cost function c_j is evaluated from the physical locations of the gates with respect to each other, which is related to the routing complexity of the circuit as well as the capacity of each cluster.

In order to evaluate the physical locations of the gates, the Cadence Virtuoso Placement and Route tool has been used to produce a compact layout floor-plan from the schematic entry. Once the compact layout floor-plan is constructed, the X,Y coordinates for every gate are extracted and the cost functions are evaluated. Figure 9 shows the floor-plan layout for the 4-bit CLA adder. The V_{dd} and gnd rails are shown and a cavity exists where the sleep transistors are located. The cavity of the sleep transistors has been taken into consideration when extracting the X,Y coordinates of every gate.

In Figure 9, gates G_1 to G_{28} are identified, and the relative distances are computed from the compact layout. The cost function is formulated as follows:

$$c_j = (w_1 \times c_{j1}) + (w_2 \times c_{j2}) \tag{19}$$

where c_{j1} is a distance function (i.e rectilinear distance between gates within a cluster) and c_{j2} represents the difference between the maximum cluster capacity and the sum of all currents of gates within a cluster.

Therefore,

$$c_{j1} = \sum d_{uv} \text{ in a group } S_j \tag{20}$$

where d_{uv} is the distance between the centers of gates G_u and G_v. For example, referring to Figure 10, group S_j is composed

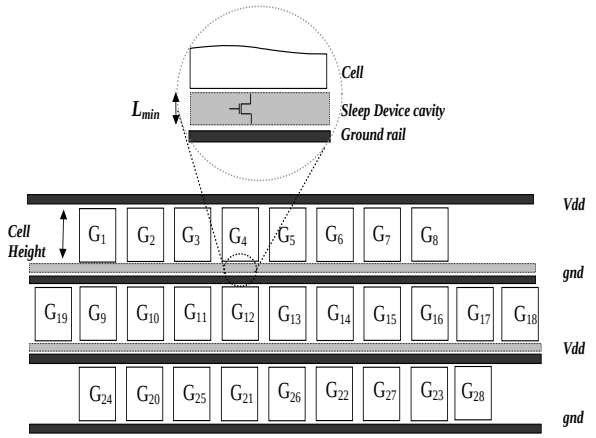


Fig. 9. 4-bit CLA Adder Floorplan

of gates G_u,G_v and G_w. The value of the partial cost function of group S_j is: c_{j1} = d_{uv} + d_{vw} + d_{wu}.

And,

$$c_{j2} = Sleep\_Transistor_{max\_current} - \sum current_i \quad \forall i \tag{21}$$

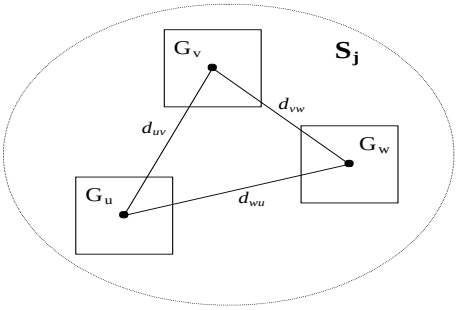


Fig. 10. Cost Function Calculation Example

The weights w_1 and w_2 are the weights associated with the cost of the two constraints i.e distance and capacity of the formed clusters. In this paper, we have assigned equal values to the weights w_1=w_2=0.5, in order to balance the distance and capacity constraints. Gates are grouped, while meeting the constraint that the sum of currents does not exceed I_{max}=250μA. Figure 11 presents a very fast and efficient heuristic to form groups of clusters that will be used by the SP technique. The heuristic forms different types of clusters (i.e clusters consisting of single gates, two gates e.t.c.). This guarantees a feasible solution for the Set Partitioning technique. The target is to select certain groups (clusters) to achieve lowest cost value, while maintaining the I_{max} constraint. The groups must also cover all gates with no repetition. As illustrated in Figure 11, the algorithm calculates the distance between all gates and creates clusters consisting of single gates (this guarantees a solution to the Binary Integer Program (BIP)). In step (4) of the algorithm, the subroutine (**Create_n_Gate_Clusters(cl)**) is utilized for forming clusters of different sizes (according to the parameter **cl** passed). In effect, a certain number (i.e target) of clusters with a specific capacity is created according to a parameter set within the

CLUSTERING HEURISTIC

Create_Clusters()
1. Calculate distances between all gates;
2. Initialize maxgates_per_cluster=n;
3. Create clusters with Single gates;
4. **For** cl=2; cl $\leq$ maxgates_per_cluster
   Create_n_Gate_Cluster(cl)
   End For
5. For all clusters created **calculate_cost()**
6. Return();

Create_n_Gate_Clusters(cl)
1. **For** cluster of type cl
   create_new_cluster()
   While not done
   Choose Gate with minimum distance
   If sum of currents $\leq$ capacity
   append gate to newly created cluster
   End If
   If total gates within cluster $\geq$ limit
   break;
   End While
   End For

Fig. 11. Heuristic for Grouping Gates into Clusters

algorithm. Currently a limit of 10 different clusters of a certain type is set as an upper bound. This number was set empirically to have a balance between the solution quality and CPU time spent to solve the BIP problem (as will be explained later on).

The mathematical formulation of the Set Partitioning problem is as follows

$$Minimize \qquad Z = \sum_{j=1}^{n} c_j S_j \tag{22}$$

subject to

$$\sum_{j=1}^{n} a_{ij} S_j = 1 \quad i = 1, ..., m \tag{23}$$

$$S_j \in 0, 1 \quad j = 1, ..., n \tag{24}$$

$$S_j = \begin{cases} 1, & \text{if the } j\text{th cluster is selected} \\ 0, & \text{otherwise} \end{cases}$$

where n is the number of groups generated and a_{ij} = 0 or 1. In this formulation each row ($i = 1, ..., m$) represents a constraint where module m should belong to. The columns ($j = 1, ..., n$) represent feasible clusters (i.e sleep transistors) that accomodates a set of gates in the circuit. The matrix a_{ij} is constructed as:

$$a_{ij} = \begin{cases} 1, & \text{if gate i is covered by cluster j} \\ 0, & \text{otherwise} \end{cases}$$

Therefore, the objective of the low-power Set Partitioning problem is to find the "best" collection of clusters such that each gate is covered by exactly one cluster. The above model

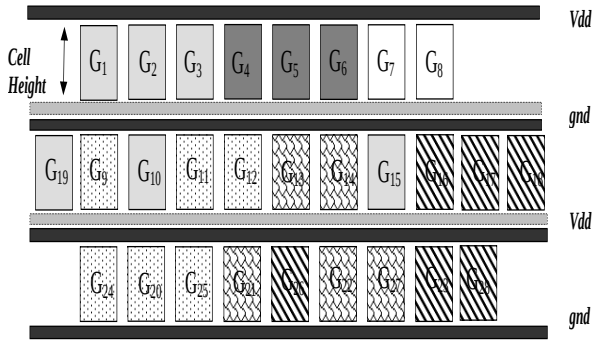


Fig. 12. Results:4-bit CLA Adder Floorplan w=(0.5,0.5)

is also a 0-1 pure integer LP problem, which is again solved using CPLEX version 7.5.

Figure 12 shows the solution of the SP technique with w_1=w_2=0.5, by highlighting the gates that are clustered together with the same pattern or color. It is evident from Figure 12 that gates that are placed closely were clustered together (i.e gates in two consecutive rows) with a specific sleep transistor therefore minimizing the wire-length.

To further illustrate how the floorplan of the clustered gates changes with a different w_1,w_2 values, Figure 13 shows the floorplan for w_1=0.9 and w_2=0.1. Since the distance dependant variable w_1 is given a larger value compared to the capacity dependant variable w_2, it could be seen from Figure 13 that the gates within a cluster are next to each other, where as the number of clusters has increased. This means that the efficiency to pack gates in a cluster has greatly degraded. In this case, the Set Partitioning modeling of the problem favors minimum distance to full capacity clustering.

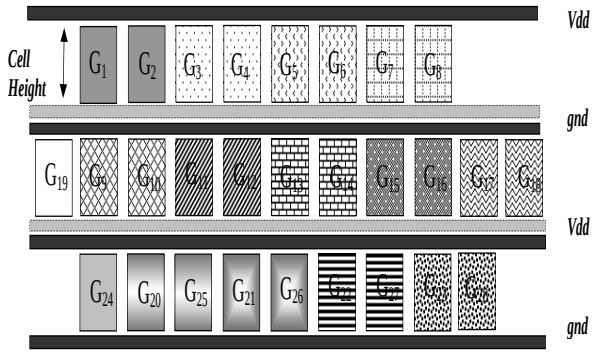


Fig. 13. Results:4-bit CLA Adder Floorplan w=(0.9,0.1)

Furthermore, Figure 14 shows the normalized $P_{dynamic}$ and $P_{leakage}$ dissipation as a function of w_1,w_2. The two curves are normalized to their minimum value. Minimum $P_{dynamic}$ is achieved at w=(w_1=0.6, w_2=0.4) while minimum $P_{leakage}$ is achieved at w=(w_1=0.5, w_2=0.5).

In general, minimum power dissipation takes place at a balanced "minimum distance"-"full capacity" clustering. On the other hand, favoring the distance constraint over the capacity constraint (ie. $w_1 \gg w_2$) causes large $P_{dynamic}$ and

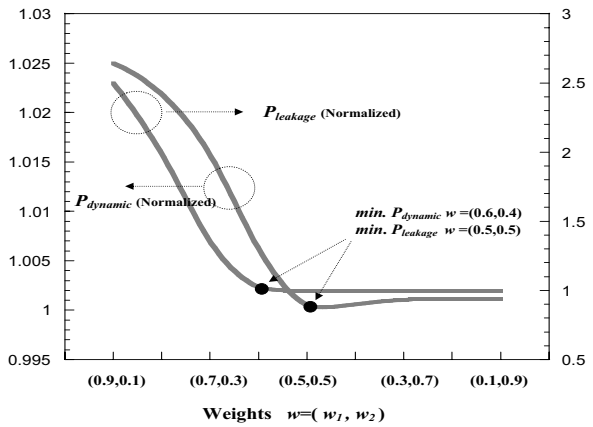


Fig. 14. Impact of weights over $P_{dynamic}$ and $P_{leakage}$

$P_{leakage}$ dissipation. Therefore, equal w_1 and w_2 values are taken throughout this work to balance the "minimum distance" to "full capacity" clustering constraints.

D. Results and Discussion

Table IV compares the SP and BP techniques to the literature. All results are normalized to [9]. The BP & SP techniques employ sleep transistors which are sized to achieve minimum FOM (as shown in Figure 8). A 5% degradation in circuit speed is achieved, where the frequency of operation is set at 500MHz. LVT and HVT are set to 350mV and 500mV respectively in the 0.18μm CMOS technology. Furthermore, the interconnects to link the sleep transistors with the gates have been taken into consideration. The area covered by these interconnects as well as the sleep transistor area have been reported. The $P_{dynamic}$ (P_d) and $P_{leakage}$ (P_l) have been measured while also taking these interconnects into account. The capacitances associated with the metal interconnect lines as well as the parasitic cross-coupling capacitance were measured. Metal 1 and Metal 2 lines were used. The capacitance for Metal 1 is 0.2236 fF/μm, while the capacitance for Metal 2 is 0.1905 fF/μm.

From Table IV, keeping the 5% speed degradation as a comparison basis, it is clear that [10] employs a smaller sized single sleep transistor containing the whole circuit compared to [9]. Consequently, a slight reduction in dynamic power is observed due to the reduction of the drain capacitance linked to the sleep transistor. Results in [10] achieve an average of 50% reduction in leakage power compared to [9].

The highest leakage reduction occurs in the 27-bit CIC benchmark. This is due to the large reduction in sleep transistor area (3247 to 153). It should be emphasized that the CIC benchmark employs many gates that have mutually exclusive discharge patterns which enhances the efficiency of [10] (unlike the other 5 benchmarks). On the other hand, the BP technique produces large reductions in the sleep transistors total area. Although the number of sleep transistors is higher than [9] and [10], the size of every sleep transistor is much smaller achieving an overall reduction in sleep transistor area. Therefore, the BP technique offers significant dynamic power

savings compared to [9] and [10] as shown in Table IV. On average, the BP technique achieves 17% reduction over [9] and 14% dynamic power reduction over [10]. The main saving however is associated with the leakage power, due to the reduction of the sleep transistor size, which is directly proportional to the leakage power dissipation. On average the BP technique achieves 95% and 85% leakage power reduction compared to [9] and [10].

The SP technique is then compared to the BP technique, [9] and [10], while still keeping the 5% speed degradation as a comparison basis. The SP technique produces large reductions in the sleep transistors total area compared to [9] and [10], but higher than BP because an additional constraint to the objective function is added (i.e routing cost) and no preprocessing is incorporated as explained in Section IV-A. The SP technique reduces the dynamic power on average by 10% and 6% compared to [9] and [10] respectively. This is attributed to the reduction of capacitance due to the down-sizing of the sleep transistors.

Furthermore, the SP technique achieves 88% and 66% leakage reduction compared to [9] and [10]. The main advantage of the SP technique is taking into consideration the location of the blocks in order to reduce the overall interconnects, providing more optimization to the area. The advantages of the SP technique will be even more evident in the DSM regime when interconnects dominate circuit performance [12] and dynamic power. More over, equally sized sleep devices for a given benchmark, as for BP and SP, facilitates design for other circuits and provides more regular layouts. The area of the sleep transistor (ST) is equal to $W_{sleep} \times L_{sleep}$. Keeping the length of the sleep transistor (L_{sleep}) constant in the 4 techniques mentioned in Table IV, the sleep transistor width (W_{sleep}) can now be used as the sleep transistor area representative. However, the total ST_Area values shown in Table IV include area of the sleep transistor interconnects in addition to the sleep transistor itself. The reduction in dynamic power is dependant on the number and size of sleep transistors and how big the circuit is (ratio of sleep transistor capacitance to overall circuit capacitance), while leakage power is only dependant on the number and size of the sleep transistors. Therefore, it can be noticed from Table IV, that the saving in leakage power is approximately proportional to the reduction in total sleep transistor area. Finally, the proposed technique offers minimal area overhead, with no perturbation to the layout. This is attributed to the very narrow cavity (Figure 9) that holds the sleep transistors, which is located at a fixed location parallel to either the supply or ground rails. This further guarantees that the sleep transistor will not change the overall floorplan of the circuit. Another point that should be mentioned is that the discharge current at the output of a gate differs very little after the insertion of the ON sleep transistor. An interesting comment to highlight is that after applying the BP and SP techniques, some sleep transistors may still have the capacity to contain more gates (ie. not fully utilized). Thus, these sleep transistors can be sized down and hence would further reduce leakage power. This can be investigated in future work to produce even more accurate results. However, the optimization steps will not change.

In summary, our proposed gate clustering technique in this work is fundamentally better than that in [9] and [10] because: (1) Partially overlapping currents are taken into account. (2) More advanced heuristics are used. (3) The technique gives good results for general structures circuits, not only tree shaped architectures, and (4) Routing complexity has been taken into account, which is an important issue in the DSM regime.

In order to further improve the results achieved, hybrid heuristics that combine the characteristics associated with the BP and SP algorithms, are devised.

V. Hybrid Heuristic Techniques

In this section we introduce several hybrid heuristics to improve upon the performance of the existing Set Partitioning technique explained in the previous section. One of the major bottlenecks in the Set Partitioning technique is the limitation of the clustering heuristic (introduced in Figure 11) to produce all possible types of clusters that can be used by a BIP solver. In other words, the clustering heuristic appends gates that are closest to form a cluster. There is no consideration to overlapping current (i.e preprocessing of gate currents) which was introduced in Section IV-A. The hybrid heuristics make use of the knowledge gained from preprocessing of gate currents in addition to the closeness of gates to form an effective cluster.

HYBRID H_A

```
1. For (each I_EQ Cluster formed)
     Create a New Cluster (add all existing gates)
     Set Total Current to Max Current of I_EQ Cluster
     For (each gate closest to current formed cluster)
       If (gate does not belong to current cluster)
         TotalCurrent = I_EQ.MaxCurrent + Gate.Current
         If (TotalCurrent ≤ BinCapacity)
           Append Gate to Newly formed cluster
         else
           Reject the gate
       End If
     Update Cost for Newly created Cluster
     End For
   End For
2. Create SP Formulation of the benchmark
3. Solve the SP using CPLEX
```

Fig. 15. A Simple Preprocessing/SP Hybrid Heuristic

Figure 15 introduces the first heuristic technique H_A. For each I_{EQ} cluster formed (i.e set of gates with overlapping currents as in Section IV-A) a new cluster is created. All gates that are close to the I_{EQ} cluster will be appended to the current cluster as long as the capacity of the sleep transistor is not exceeded. For all clusters formed, a new Set Partitioning is formulated and then solved by CPLEX as a BIP problem. The main advantage of this heuristic is that the formed clusters can have much more current capacity than the simple clustering heuristic introduced in Section IV-C. Solving a Set Partitioning problem based on this hybrid heuristic will achieve results that utilize less sleep transistors than the pure Set Partitioning formulation based on the simple clustering

technique. To further improve the performance of the Set Partitioning problem we introduced a second hybrid technique H_B. This hybrid not only merges existing close gates to the I_{EQ} clusters formed by preprocessing but also merges I_{EQ} clusters together.

HYBRID H_B

1. **For** (each I_{EQ} Cluster formed)
   Find all possible I_{EQ} Candidates that can be added
   For (all possible new clusters that need to be created)
   Copy I_{EQ} Cluster to New Cluster (add other gates)
   Set Total Current to Max Current of I_{EQ} Cluster
   For (every other I_{EQ} Cluster in the pool)
   TotalCurrent = I_{EQ}.Max + OtherI_{EQ}.Max
   If (TotalCurrent $\leq$ BinCapacity)
   Append All Gates of New I_{EQ} to New Cluster
   else
   Reject the I_{EQ}
   End If
   Update Cost for Newly created Cluster
   End For
   End For
2. Create SP Formulation of the benchmark
3. Solve the SP using CPLEX

Fig. 16. An Effective I_{EQ}+I_{EQ} SP Clustering Hybrid Heuristic

As seen in Figure 16, the heuristic merges every I_{EQ} cluster with all other possible I_{EQ} clusters. It is important to notice that the efficiency of the heuristic will be less effective than the technique shown in Figure 11. It is expected that the solution quality of the two hybrid techniques lay between those obtained by the Bin Packing formulation and those based on solving the Set Partitioning problem (using a simple clustering heuristic).

The third hybrid heuristic H_C (shown in Figure 17) is similar to H_B except that all close gates to the newly formed cluster (i.e combination of I_{EQ} clusters) are appended as long as their current does not exceed the upper limit of the sleep transistor.

Finally, the last hybrid heuristic technique H_D creates clusters by utilizing all the previous hybrid heuristic approaches explained in addition to the regular clustering heuristic introduced in Section IV-C.

Table V compares the results obtained by the four Hybrid Heuristics. It is clear from the table that the quality of solutions obtained by all hybrid heuristics is better than the simple clustering technique initially proposed for solving the Set Partitioning problem (refer to Table VI). It is also evident from Table V that combining I_{EQ} clusters together reduces the total number of sleep transistors (especially for sleep transistors with large capacity) but at the expence of routing complexity. It is interesting to notice that hybrid heuristic technique H_D achieves similar results to those obtained using hybrid H_B in terms of the number of sleep transistors utilized. An important fact that might be overlooked in this case is that heuristic H_D accounts for the routing complexity of the gates to the sleep transistor. In addition, since hybrid H_D creates clusters by utilizing hybrids H_A, H_B and H_C, the Set Partitioning problem will be less constrained (i.e more clusters formed) and therefore can be solved in less time.

HYBRID H_C

1. **For** (each I_{EQ} Cluster formed)
   Find all possible I_{EQ} Candidates that can be added
   For (all possible new clusters that need to be created)
   Copy I_{EQ} Cluster to New Cluster (add other gates)
   Set Total Current to Max Current of I_{EQ} Cluster
   For (every other I_{EQ} Cluster in the pool)
   TotalCurrent = I_{EQ}.Max + OtherI_{EQ}.Max
   If (TotalCurrent $\leq$ BinCapacity)
   Append All Gates of New I_{EQ} to New Cluster
   else
   Reject the I_{EQ}
   End If
   End For
   For (all other close gates to newly formed cluster)
   TotalCurrent = CurrClus.Max + Gate.Current
   If (TotalCurrent $\leq$ BinCapacity)
   Append the Gate to Newly formed cluster
   End For
   Update Cost for Newly created Cluster
   End For
   End For
2. Create SP Formulation of the benchmark
3. Solve the SP using CPLEX

Fig. 17. An Effective I_{EQ}+I_{EQ}+GATE Preprocessing/SP Clustering Heuristic

Hybrid H_D

1. Initialize
2. Create 1 gate clusters
3. Use Preprocessing Heuristic
4. Use Regular Clustering Heuristic
5. Use Hybrid H_A to create clusters
6. Use Hybrid H_B to create clusters
7. Use Hybrid H_C to create clusters
8. Remove redundant clusters
9. Create SP Formulation of the benchmark
10. Solve the SP using CPLEX

Fig. 18. Combined Hybrid Heuristics

Table VI compares the results obtained by: Bin Packing, Set Partitioning based on simple clustering and Set Partitioning based on the newly introduced hybrid heuristic H_D. It is clear from the table that the Bin Packing model produces solutions with the least number of sleep transistors. The Set Partitioning model based on the simple clustering technique (introduced in Figure 11) gives the maximum number of sleep transistors but with less complex routing. Finally the results obtained based on the Set Partitioning model (using the hybrid heuristic technique) offers solutions that are balanced between the total number of sleep transistors to be used and routing complexity. Since H_D has been the heuristic technique of choice, H_D will be denoted by the "Hybrid Problem" (HP).

Figure 19 shows the CPU time involved in solving the benchmarks for BP, SP (based on the simple clustering technique) and SP (based on the Hybrid H_D) respectively. It is evident from the figure that solving the SP problem involves more CPU cycles than solving the BP problem. This is due to the fact that the number of variables and constraints in the SP problem are much larger than that of the BP problem.

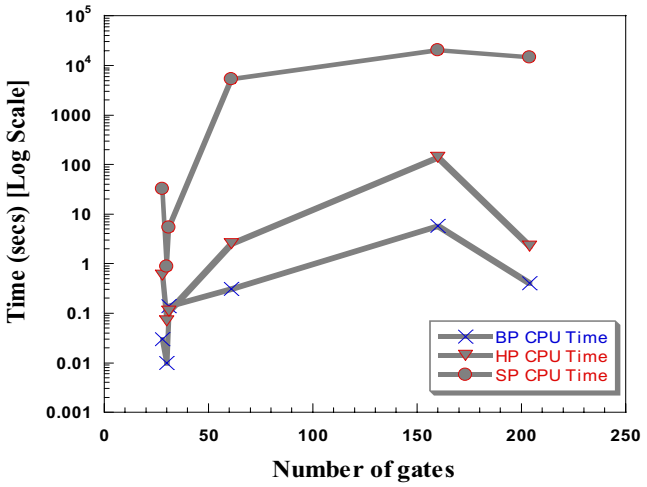


Fig. 19. Computation Time for BP, SP and HP

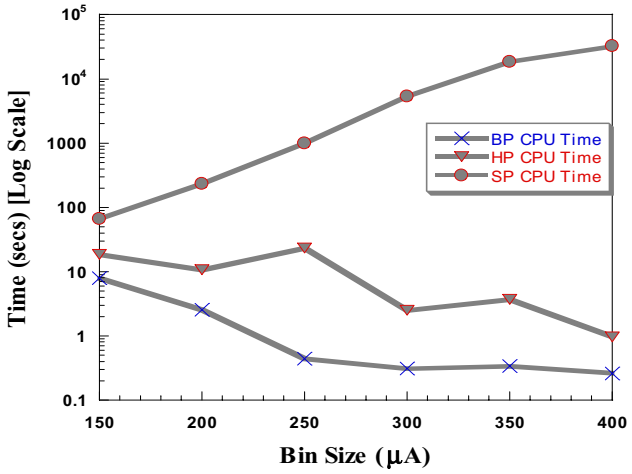


Fig. 20. Computation Time for BP, SP and HP (ALU Benchmark)

The CPU time of the SP technique improves drastically as we invoke the hybrid heuristic versus the simple clustering technique. This is due to the fact that as we increase the number of clusters generated for the SP technique, the smaller the computation time involved (as evident from the graph with respect to the Hybrid approach). Figure 20 shows the effect of the size of the bin (I_{sleep}) with respect to the computation time for the ALU benchmark. The BP preprocessing algorithm has a worst case complexity of $O(n^2)$, where *n* is the number of gates in the circuit. On the other hand, the SP algorithm complexity is $O(nk)$ where *n* is the number of gates in the circuit and *k* is the maximum gates to be appended in a cluster. For large circuits it is recommended that heuristic search techniques such as Genetic Algorithms and Tabu Search would be used instead of the CPLEX solver. The hybrid heuristic technique H_D is chosen to be verified by the six benchmarks. This is because the H_D technique creates clusters using all the other hybrid heuristic approaches, and thus employs less sleep transistors in the design (Table V), leading to dynamic and leakage power savings.

VI. Virtual Ground Bounce

So far, the criteria for sizing the sleep transistor, have been performance (5% speed degradation is set), as well as the minimization of dynamic and leakage power. However, an equally important design criterion, is sizing the sleep transistor for noise. In MTCMOS circuits, virtual ground rails have a higher impedance than the true ground rails, and will thus unavoidably bounce. This will cause a serious reduction in gate speed as the effective supply voltage decreases, as well as a degradation to the noise margins. The problem with ground bounce is that many logic gates share a centralized sleep transistor, hence the same virtual ground.

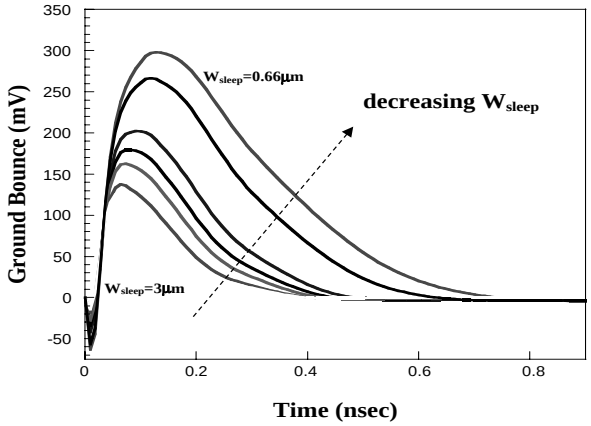


Fig. 21. Transient Response: Ground Bounce

Figure 21 shows the virtual ground bounce transient. It was generated by simulating the transient response of the virtual ground rail for different sleep transistor sizes for the ALU benchmark. The virtual ground rail attached to the sleep transistor that holds the highest number of gates is monitored. As the sleep transistor width decreases, the longer the time duration of the ground bounce bump. This is attributed to the RC time constant associated with the very high resistance case produced from the small sized sleep transistors. Consequently, this would cause the waveform at the gate output supported by that sleep transistor to slow down. Figure 22 shows the variation of ground bounce with the sleep transistor size. The smaller the sleep transistor, the higher the ground bounce. Therefore, the sleep transistor should not only be sized for speed, dynamic and leakage power, but for noise as well. Previous work [9], [10] have not included ground bounce in their analysis which is a critical issue that should have been taken into consideration. Some physical issues related to the ground bounce will be first illustrated, followed by the design methodology that takes ground bounce into account.

A. Impact of Virtual Ground Capacitance

The wire and junction capacitance associated with the virtual ground line should actually help reduce the ground

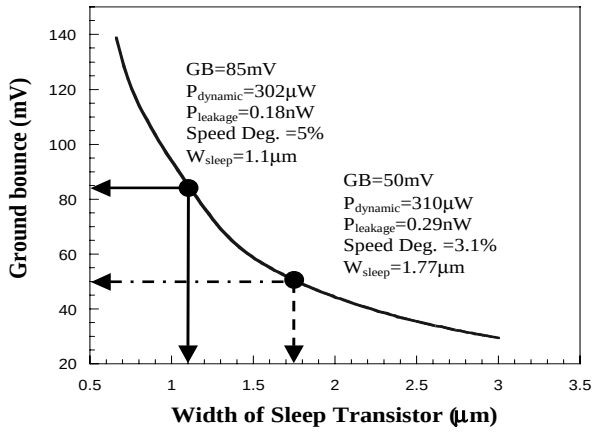


Fig. 22. Ground Bounce vs. W_{sleep}

bounce by serving as a local charge sink or reservoir for current [9] as shown in Figure 23. However, this capacitance would have to be extremely large in order to offset the effects of a poorly sized sleep transistor. The RC network serves as a lowpass filter, where the RC time constant would have to be large enough such that the virtual ground voltage can only rise to a fraction of its peak DC value.

If the time constant is very large, then it will also take longer for the virtual ground node to discharge back to ground after a transition (as seen in Figure 21). Rather than rely on large capacitances to ensure MTCMOS performance, it is much easier to lower the effective resistance with proper sleep transistor sizing instead.

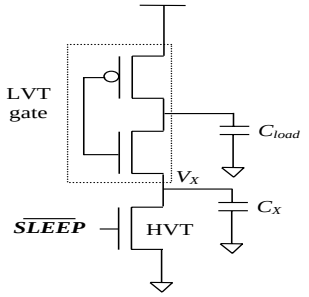


Fig. 23. Capacitance associated with virtual ground rail

B. Reverse Conduction Paths through Virtual Ground

MTCMOS logic blocks can also suffer from reverse conduction as shown in Figure 24, where current flows from the virtual ground though the LVT pull-down devices and charges up the output load capacitance [11]. The virtual ground rises above 0V so that another gate (which is supposed to be low) can experience reverse conduction as the output voltage rises from 0 to V_X. This charging current comes from the discharging current of other gates transitioning from high to low. As a result, the MTCMOS circuit is slightly faster because the V_X voltage drop is not quite as large as one would expect if all current flowed through the sleep transistor to ground. Another effect of the reverse conduction is that gate charging from low to high would be faster since it is already precharged to V_X. The drawback is that the noise margins in the circuits are reduced, and in the worst case the circuit can fail logically. Therefore, the sleep transistor should be again properly sized to attain adequate noise margins.

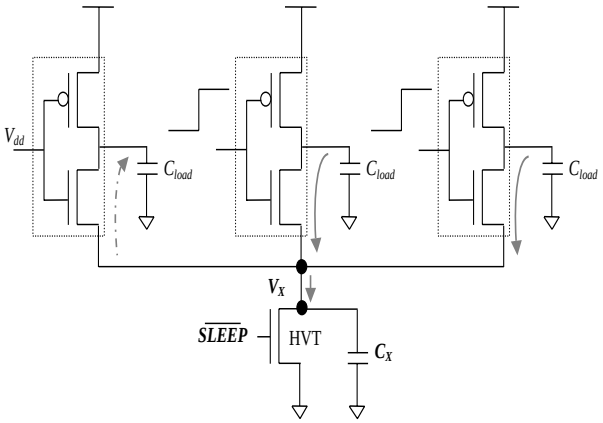


Fig. 24. Reverse Conduction

C. Design Methodology

In order to include ground bounce as a design criterion, dynamic and leakage power are reduced under two constraints that must be achieved simultaneously. Firstly, the speed degradation is set to never exceed 5% and secondly, ground bounce is also set to never exceed 50mV. Based on these constraints, the circuit guarantees to achieve sufficient speed and noise margins.

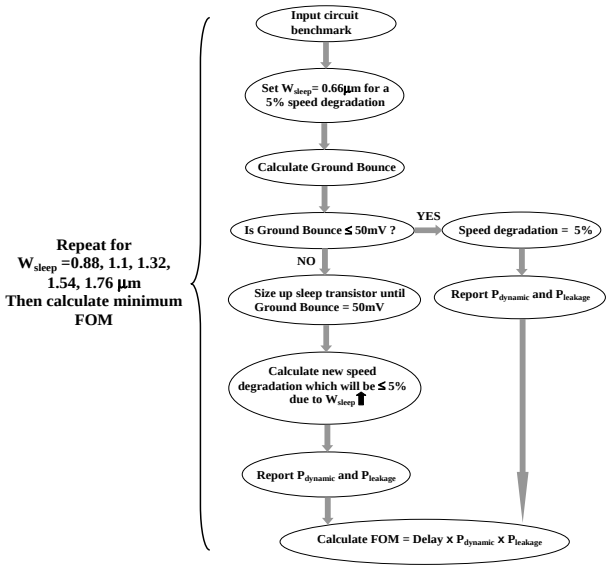


Fig. 25. Design Methodology

In Section IV-B, Table III showed the values for W_{sleep} for a 5% speed degradation. For W_{sleep} = 1.1μm, the speed degradation due to the sleep transistor is 5% when I_{sleep}=250μA.

Due to the small size of the sleep transistor both dynamic and leakage power are reduced during the active and idle modes respectively. However, the ground bounce on the virtual ground rail is high and equal to 85mV which is over the acceptable noise limit (50mV) (this is shown in Figure 22). The sleep transistor is therefore sized up until ground bounce is reduced to 50mV. This is achieved at W_{sleep}=1.77μm. The sizing up of the sleep transistor actually enhances the circuit speed, and now causes only a 3.1% degradation in speed. However, dynamic and leakage power will rise due to the sizing up of W_{sleep}. A figure of merit (FOM) is therefore established which takes into account delay, dynamic and leakage power dissipation while attaining a ground bounce $\leq$ 50mV.

$$Figure{-}of{-}Merit = P_{dynamic} \times P_{leakage} \times Speed_{Degradation} \quad (25)$$

This FOM is calculated for different sleep transistor sizes while adhering to the constraints mentioned, and the W_{sleep} achieving minimum FOM is recorded. It should be noted that the area associated with the sleep transistors is implicitly in Eq. 25 of the $P_{leakage}$ term.

The design methodology is presented by the flow diagram in Figure 25. This methodology is applied to each of the six benchmarks, each at six different sleep transistor sizes; W_{sleep} = 0.66, 0.88, 1.1, 1.32, 1.54, 1.76μm. The design methodology starts by assuming that speed degradation is equal to 5%. Ground bounce (GB) is then calculated. If GB $\leqslant$ 50mV, then speed degradation will be taken as the pre-assumed 5%, and $P_{dynamic}$, $P_{leakage}$ are reported, followed by calculating the FOM. On the other hand, if GB is higher than 50mV, the sleep transistor is sized up until GB=50mV. The new value for speed degradation is reported, as well as $P_{dynamic}$,$P_{leakage}$. The related FOM is then calculated. For every benchmark, FOM is calculated for the different sleep transistor sizes, and the size achieving minimum FOM is recorded. At this sleep transistor size, the speed, $P_{dynamic}$ and $P_{leakage}$ powers are thus recorded. Since a single benchmark would employ several sleep transistors, ground bounce was always monitored on the virtual ground rail attached to the sleep transistor that holds the highest number of gates.

VII. Results: Taking Ground Bounce into Account

Table VII summarizes the results for the six benchmarks where the Bin Packing technique (while taking ground bounce into account) BP_{GB} is compared to the same technique BP without taking ground bounce into account (Table IV). Similarly the Set Partitioning technique while taking ground bounce in account SP_{GB} is compared to the Set Partitioning technique without taking ground bounce into account SP. Furthermore, the devised hybrid heuristic while taking ground bounce into account HP_{GB} is also compared to the BP, BP_{GB}, SP and SP_{GB}.

The BP_{GB} technique achieves on average 10% and 6% dynamic reduction compared to [9] and [10]respectively. Similar to BP, the main saving for BP_{GB} is associated with leakage power, where 80% and 65% savings are achieved compared to [9] and [10] respectively. From Table VII, the BP_{GB} technique achieves lower dynamic and leakage savings compared to the BP approach. This is expected, since taking noise immunity into consideration, causes the sleep transistor to size up, dissipating more dynamic and leakage power. However, due to this sizing up of the sleep transistor, a reduction in delay is associated with BP_{GB} compared to BP. By constructing the FOM (Eq. 25), it could be seen from Figure 26 that BP_{GB} has slightly higher FOM values compared to the BP technique.

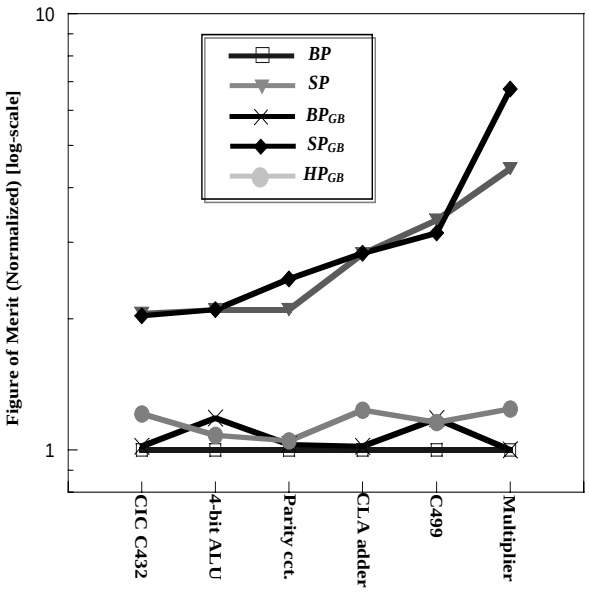


Fig. 26. Comparison: Figure of Merit for benchmarks

On the other hand, the SP_{GB} technique achieves on average 9.5% and 6.3% dynamic reduction compared to [9] and [10] respectively. Furthermore, leakage power savings of 82% and 61% are achieved compared to [9] and [10]. Similar to BP_{GB}, the SP_{GB} technique achieves lower dynamic and leakage savings compared to the SP approach. This is again expected, due to the same cause highlighted above for the BP_{GB} case. Moreover, Figure 26 shows that both the SP and SP_{GB} achieve high FOM. This is attributed to the large number of sleep transistors employed in the circuit due to set-partioning technique, leading to large power values.

The HP_{GB} hybrid heuristic while taking ground bounce into account, achieves low FOM values. HP_{GB} achieves on average 10% and 6.6% dynamic reduction compared to [9] and [10], while 82% and 60% savings in leakage power are achieved compared to [9] and [10]. Figure 26 shows that HP_{GB} achieves comparable FOM values compared to BP and BP_{GB}, but has the advantage that interconnect complexity is taken into consideration. Furthermore, HP_{GB} takes noise associated with ground bounce into consideration, unlike BP. HP_{GB} could therefore be considered the best technique that takes sleep transistor capacity constraints and interconnect complexity into consideration while achieving low leakage and dynamic dissipation values, as well as sufficient performance.

VIII. Conclusions and Future Work

This paper presented several heuristic techniques for efficient gate clustering in MTCMOS circuits by modeling the problem via Bin Packing (BP) and Set Partitioning (SP) techniques. The SP technique takes the circuit routing complexity into consideration which is critical for Deep Sub-Micron (DSM) implementations. By applying the technique to six benchmarks to verify functionality, results obtained indicate that our proposed techniques can achieve on average 84% and 12% savings for leakage power and dynamic power respectively. Furthermore, four hybrid clustering techniques that combine the BP and SP techniques to produce a more efficient solution are also devised. Moreover, the noise associated with ground bounce was also taken as a design parameter in the optimization problem. While accounting for noise, the proposed hybrid solution achieves on average 9% savings for dynamic power and 72% savings for leakage power dissipation. This is achieved at sufficient speeds and adequate noise margins. For future work: (1) we would like to utilize a more accurate technique that estimates the maximum current envelope drawn by a circuit [20] to further enhance our current results, (2) if the ratio between the active/standby periods is relatively small, then it could be taken into account to find the optimal size for the sleep transistor, (3) to further enhance the current results, a more optimized preprocessing heuristic should be devised, and (4) utilize advanced heuristic search techniques to efficiently solve both the Bin Packing and Set Partitioning problems instead of the pure binary programming models.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments to improve the quality of this paper. The authors would also like to thank Mohamed Mahmoud of Texas Instruments for his preliminary work.

References

[1] T.Kuroda, "A 0.9v 150mhz 10mw $4mm^2$ 2-d discrete cosine transform core processor with variable threshold voltage scheme," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1770–1779, Nov. 1996.

[2] M.Stan, "Low-Threshold CMOS Circuits with Low Standby Current," in *Proceedings of the International Symposium on Low-Power Electronics and Design*. Monterey, CA: IEEE/ACM, 1998, pp. 97–99.

[3] J.Hatler and F.Najm, "A Gate-Level Leakage Power Reduction Method for Ultra Low-Power CMOS Circuits," in *Proceedings of the IEEE Custom Integrated Circuits Conference*. Santa Clara, CA: IEEE, 1997, pp. 475–478.

[4] Z.Chen, L.Wei, and K.Roy, "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks," in *Proceedings of the International Symposium on Low-Power Electronics and Design*. Monterey, CA: IEEE/ACM, 1998, pp. 239–244.

[5] Y.Ye, S.Borkar, and V.De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," in *Proceedings of the 1998 Symposium on VLSI Circuits*. IEEE, 1998, pp. 40–41.

[6] M.Anis and M.Elmasry, *Multi-Threshold CMOS Digital Circuits - Managing Leakage Power*. Kluwer Academics Publications, 2003.

[7] L.Wei, Z.Chen, K.Roy, M.Johnson, and V.De, "Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications," *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 16–24, March 1999.

[8] S.Sirichotiyakul, T.Edwards, O.Chanhee, Z.Jingyan, A. harchoudhury, R.Panada, and D.Blaauw, "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing," in *Proceedings of the 36th Design Automation Conference*. New Orleans, LA: IEEE/ACM, 1999, pp. 436–441.

[9] S.Mutah, T.Douseki, Y.Matsuya, T.Aoki, S.Shigematsu, and J.Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multi-Threshold Voltage CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 847–853, 1995.

[10] J.Kao, S.Narendra, and A.Chandrakasan, "MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns," in *Proceedings of the 35th Design Automation Conference*. Las Vegas, Nevada: IEEE/ACM, 1998, pp. 495–500.

[11] J.Kao, A.Chandrakasan, and D.Antoniadis, "Transistor Sizing Issues And Tools For Multi-threshold CMOS Technology," in *Proceedings of the 34th Design Automation Conference*. Las Vegas, Nevada: IEEE/ACM, 1997, pp. 409–414.

[12] M.Bohr and Y.Elmansy, "Technology for Advanced High-Performance Microprocessors," *IEEE Transactions on Electron Devices*, vol. 45, no. 3, pp. 620–625, 1998.

[13] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design Circuits and Systems*. Kluwer Academics Publications, 1995.

[14] J.M.Rabaey, *Digital Integrated Circuits*. Prentice Hall, 1996.

[15] Y.Taur and T.Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 1999.

[16] D.Sylvester and C.Hu, "Analytical Modeling and Characterization of Deep-Submicron Interconnect," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 634–664, 2001.

[17] M.Anis, S.Areibi, M.Mahmoud, and M.Elmasry, "Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering," in *Proceedings of the 39th Design Automation Conference*. New Orleans: IEEE/ACM, 2002, pp. 480–485.

[18] R.Rardin, *Optimization in Operations Research*. Boston: Prentice Hall, 1998.

[19] S.Mutoh, S.Shigematsu, Y.Matsuya, H.Fukuda, T.Kaneko, and J.Yamada, "A 1-V multithreshold-voltage CMOS digital signal processor for mobile phone applications," *IEEE Journal of Solid-State Circuits*, pp. 1795–1802, 1996.

[20] S. Bobba and I. Hajj, "Estimation of Maximum Current Envelope For Power Bus Analysis and Design," in *Proceedings of International Symposium of Physical Design*. Moneteray: IEEE/ACM, 1998, pp. 141–146.

TABLE I

RESULTS: CURRENT EQUIVALENCE

I_{EQ_1} $I_{overlap}(\mu A)$=80	I_{EQ_2} 80	I_{EQ_3} 110	I_{EQ_4} 90	I_{EQ_5} 50	I_{EQ_6} 30	I_{EQ_7} 16	I_{EQ_8} 50	I_{EQ_9} 30	$I_{EQ_{10}}$ 16	$I_{EQ_{11}}$ 50	$I_{EQ_{12}}$ 30	$I_{EQ_{13}}$ 50	$I_{EQ_{14}}$ 37
I_1, I_2, I_{12}, I_{19}, I_{20}, I_{21}, I_{22}, I_{24}, I_{25}, I_{26}	I_3, I_4, I_{27}, I_{28}	I_5, I_6	I_7, I_8	I_9	I_{10}	I_{11}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{23}

TABLE II

RESULTS: CURRENT ASSIGNMENTS

Sleep Transistor (Cluster)	1	2	3
Equivalent Currents	I_{EQ_5}, I_{EQ_7}, I_{EQ_8}, I_{EQ_9}, $I_{EQ_{10}}$, $I_{EQ_{11}}$, $I_{EQ_{12}}$	I_{EQ_1}, I_{EQ_6}, $I_{EQ_{14}}$	I_{EQ_3}, I_{EQ_4}, $I_{EQ_{13}}$
Assigned Gates	G_9, G_{11}, G_{13}, G_{14}, G_{15}, G_{16}, G_{17}	G_1, G_2, G_3, G_4, G_{10}, G_{12}, G_{19}, G_{20}, G_{21}, G_{22}, G_{23}, G_{24}, G_{25}, G_{26}, G_{27}, G_{28}	G_5, G_6, G_7, G_8, G_{18}
$\sum$ Currents(μA)	242	227	250

TABLE III

VALUES FOR I_{sleep}

I_{sleep} (μA)	150	200	250	300	350	400
$(W/L)_{sleep}$	3.67	4.89	6	7.5	8.56	9.78
W_{sleep} (μm)	0.66	0.88	1.1	1.32	1.54	1.76

TABLE IV
ALGORITHM COMPARISON

REF	Benchmark	4-bit CLA Adder	32-bit Parity Checker	6-bit Multiplier	4-bit 74181 ALU	32-bit Single Error Correcting C499	27-channel interrupt controller circuit C432
	No. of gates	28	31	30	61	202	160
[9]	Delay (Norm.)	1	1	1	1	1	1
	P_d (Norm.)	1	1	1	1	1	1
	P_l (Norm.)	1	1	1	1	1	1
	# Sleep Trans	1	1	1	1	1	1
	Total ST_Area [$W_{sleep}(\mu$m)]	50	42	65	97	176	3247
[10]	Delay (Norm.)	1	1	1	1	1	1
	P_d (Norm.)	0.98	0.97	0.89	0.97	0.99	0.98
	P_l (Norm.)	0.58	0.51	0.23	0.41	0.46	0.05
	# Sleep Trans	11→1	16→1	5→1	37→1	32→1	52→1
	Total ST_Area [$W_{sleep}(\mu$m)]	29.3	21.6	15	39.5	81	153
BP	Delay (Norm.)	1	1	1	1	1	1
	P_d (Norm.)	0.86	0.81	0.67	0.81	0.80	0.98
	P_l (Norm.)	0.066	0.062	0.041	0.072	0.052	0.0062
	P_d savings to [9]	14%	18.4%	31.4%	17%	20%	2%
	P_d savings to [10]	12.2%	15.9%	23%	14.4%	19.2%	0%
	P_l savings to [9]	93.4%	92.3%	89.0%	92.8%	94.8%	99.4%
	P_l savings to [10]	88.6%	84.8%	71.0%	82.4%	88.7%	87.6%
	# Sleep Trans	3	4	3	7	5	11
	W_{sleep} (μm)	1.1	0.66	1.32	0.88	1.54	1.54
	Total ST_Area [$W_{sleep}(\mu$m)]	3.6	2.82	4.36	7.1	9.3	20.5
SP	Delay (Norm.)	1	1	1	1	1	1
	P_d (Norm.)	0.91	0.88	0.81	0.85	0.91	0.98
	P_l (Norm.)	0.117	0.12	0.15	0.126	0.13	0.0107
	P_d savings to [9]	7%	9%	19%	11%	9%	2%
	P_d savings to [10]	5.1%	6.2%	9%	8.2%	8.1%	0%
	P_l savings to [9]	87%	85%	85%	86%	87%	98.9%
	P_l savings to [10]	77.7%	70.4%	34.8%	65.6%	71.1%	76.6%
	# Sleep Trans	9	9	9	18	22	40
	W_{sleep} (μm)	0.66	0.66	1.1	0.66	1.1	0.88
	Total ST_Area [$W_{sleep}(\mu$m)]	7.2	7.3	11.2	13.15	26.55	38.8

TABLE V
COMPARISON OF SLEEP TRANSISTORS FOR HYBRID HEURISTICS

Circuit	I_{sleep}=150				I_{sleep}=250				I_{sleep}=350				I_{sleep}=400			
	H_A	H_B	H_C	H_D	H_A	H_B	H_C	H_D	H_A	H_B	H_C	H_D	H_A	H_B	H_C	H_D
CLAD	9	8	9	7	6	5	5	5	5	4	5	4	5	3	5	3
Mult	10	10	10	10	9	6	6	6	6	3	3	3	6	2	3	2
Parity	8	4	4	4	5	3	3	3	4	2	4	2	4	2	4	2
Alu	17	17	17	17	10	6	7	6	9	5	6	5	8	4	5	4
Error	34	14	14	14	28	9	13	9	28	6	8	6	18	6	8	6
AllCh	53	47	47	47	32	19	22	19	25	12	13	12	23	11	11	11

TABLE VI
BP/SP/HYBRID COMPARISON OF SLEEP TRANSISTORS

Circuit	I_{sleep}=150			I_{sleep}=250			I_{sleep}=300			I_{sleep}=350			I_{sleep}=400		
	BP	SP	H_D	BP	SP	H_D	BP	SP	H_D	BP	SP	H_D	BP	SP	H_D
CLAD	6	9	7	3	6	5	3	6	4	3	5	4	3	6	3
Mult	6	18	10	3	9	6	2	9	2	2	8	3	2	7	2
Parity	4	9	4	3	7	3	2	6	2	2	6	2	2	6	2
Alu	10	18	17	6	12	6	5	11	5	4	11	5	4	11	4
Error	12	33	14	8	22	9	6	20	6	5	21	6	5	21	6
AllCh	32	56	47	16	33	19	13	30	14	11	28	12	10	27	11

TABLE VII
GROUND BOUNCE ALGORITHM COMPARISON

REF	Benchmark	4-bit CLA Adder	32-bit Parity Checker	6-bit Multiplier	4-bit 74181 ALU	32-bit Single Error Correcting C499	27-channel interrupt controller circuit C432
	No. of gates	28	31	30	61	202	160
BP	Delay (Norm.)	1	1	1	1	1	1
	P_d savings to [9]	14%	18.4%	31.4%	17%	20%	2%
	P_d savings to [10]	12.2%	15.9%	23%	14.4%	19.2%	0%
	P_l savings to [9]	93.4%	92.3%	94.9%	92.8%	94.8%	99.4%
	P_l savings to [10]	88.6%	84.8%	77.8%	82.4%	88.7%	87.6%
	# Sleep Trans	3	4	3	8	6	13
	W_{sleep} (μm)	1.1	0.66	0.88	0.88	1.54	1.54
	Total ST_Area [W_{sleep}(μm)]	3.6	2.82	2.82	7.1	9.3	20.5
SP	Delay (Norm.)	1	1	1	1	1	1
	P_d savings to [9]	7%	9%	19%	11%	9%	2%
	P_d savings to [10]	5.1%	6.2%	9%	8.2%	8.1%	0%
	P_l savings to [9]	87%	85%	85%	86%	87%	98.9%
	P_l savings to [10]	77.7%	70.4%	34.8%	65.6%	71.1%	76.6%
	# Sleep Trans	9	9	9	18	22	40
	W_{sleep} (μm)	0.66	0.66	1.1	0.66	1.1	0.88
	Total ST_Area [W_{sleep}(μm)]	7.2	7.3	11.2	13.15	26.55	38.8
BP_{GB}	Delay (Norm.)	0.61	0.644	0.13	0.24	0.163	1
	P_d savings to [9]	11%	16.5%	14.3%	9.4%	6.4%	2%
	P_d savings to [10]	9.2%	13.9%	3.7%	6.6%	5.5%	0%
	P_l savings to [9]	90.5%	90.4%	69.2%	67.5%	99.3%	58%
	P_l savings to [10]	84.3%	81.2%	51%	25%	35.5%	98.2%
	# Sleep Trans	3	4	3	7	7	14
	W_{sleep} (μm)	1.77	1.005	6.675	4.485	7.97	1.32
	Total ST_Area [W_{sleep}(μm)]	5.84	4.42	22.03	34.5	61.4	20.3
SP_{GB}	Delay (Norm.)	0.97	1	0.552	0.568	0.526	0.88
	P_d savings to [9]	9%	12%	12%	13%	8.7%	2%
	P_d savings to [10]	7.1%	9.3%	1.1%	10.3%	7.8%	2%
	P_l savings to [9]	87.9%	85.9%	62%	78.8%	76.9%	98.8%
	P_l savings to [10]	79.1%	72.4%	38%	48.3%	49.8%	76%
	# Sleep Trans	9	9	9	18	33	40
	W_{sleep} (μm)	0.6713	0.66	2.74	1.142	1.23	0.984
	Total ST_Area [W_{sleep}(μm)]	6.65	6.55	26.8	22.86	44.95	43.25
HP_{GB}	Delay (Norm.)	0.46	0.55	0.13	0.312	0.126	0.688
	P_d savings to [9]	10%	15%	14%	11.6%	7.7%	2%
	P_d savings to [10]	8.2%	12.4%	3.4%	8.9%	6.8%	0%
	P_l savings to [9]	88.7%	88.8%	69.4%	80%	65%	99.1%
	P_l savings to [10]	80.5%	78%	45%	51.2%	24%	81.6%
	# Sleep Trans	5	4	3	8	13	20
	W_{sleep} (μm)	1.411	1.18	6.64	2.773	5.15	1.57
	Total ST_Area [W_{sleep}(μm)]	7.75	5.15	22.1	24.4	73.6	34.5