Clock-skew optimization for peak current reduction

P. Vuillod* L. Benini A. Bogliolo G. De Micheli

Computer Systems Laboratory
Stanford University, Stanford CA 94305-9030

Abstract

The presence of large current peaks on the power and ground lines is a serious concern for designers of synchronous digital circuits. Current peaks are caused by the simultaneous switching of highly loaded clock lines and by the signal propagation through the sequential logic elements. In this work we propose a methodology for reducing the amplitude of the current peaks. This result is obtained by clock skew optimization. We propose an algorithm that determines the clock arrival time at each flip-flop in order to minimize the current peaks while respecting timing constraint. Our results on benchmark circuits show that current peaks can be reduced by more than a factor of two without penalty on cycle time and average power dissipation. Our methodology is therefore well-suited for low-power systems with reduced supply voltage, where low noise margins are a primary concern.

1 Introduction

Clock skew is usually described as an undesirable phenomenon occurring in synchronous circuits. If clock skew is not properly controlled, unexpected timing violations and system failures are possible. Mainly for this reason, research and engineering effort has been devoted to tightly control the misalignment in the arrival times of the clock. Although clock-skew control is still an open issue for extremely large chip-level and board-level designs, recently proposed algorithms for skew minimization have reported satisfying results [4, 3, 5]. For a large class of systems skew control can therefore be achieved with sufficient confidence margin.

Conservative design styles (such as those adopted for FPGAs) explicitly discourage "tampering with the clock"[12]. Nevertheless, the arrival time of the clock is often purposely skewed to achieve high performance in more aggressive design styles. In the recent past, several algorithms for *cycle-time minimization* have been proposed [1, 2, 7, 9, 8]. The common purpose of these methods was to find an *optimum clock-skewing strategy* that allows the circuit to run globally faster.

*On leave from INPG - CSI, Grenoble, France.

In this work, we discuss the productive use of clock skew in a radically new context. We target the minimization of the peak power supply current. Peak current is a primary concern in the design of power distribution networks. In state-of-the-art VLSI systems, power and ground lines must be over-dimensioned in order to account for large current peaks. Such peaks determine the maximum voltage drop and the probability of failure due to electro-migration [13]. In synchronous systems, this problem is particularly serious. Since *all sequential elements* are clocked, huge current peaks are observed in correspondence of the clock edges. These peaks are caused not only by the large clock capacitance, but also by the switching activity in the sequential elements and by the propagation of the signals to the first levels of combinational logic.

In this paper, we focus on single-clock edge-triggered clocking style, because it represents the worst case condition for current peaks. We propose an algorithm that determines the clock arrival times at the flip-flops in order to minimize the maximum current on the power supply lines, while satisfying timing constraints for correct operation. In addition, we propose *a clustering* technique that groups the flip-flops with the same clock driver. Since the number of sequential elements is generally large, it would not be practically feasible to specify a skew value for each one of them. In our approach, the user can specify the maximum number of clock drivers, and the algorithm will satisfy this constraint while minimizing the current peak.

Our technique is particularly relevant for low-power systems with reduced supply voltage, where the noise margins on power and ground are extremely low. Our method not only reduces the current peaks, but it does not increase the average power consumption of the system. We tested our approach on several benchmark circuits. In average, current peak reduction of more than 30% has been observed. Average power dissipation is unchanged. Timing constraints are satisfied by construction.

2 Skew optimization

It is known that clock skew can be productively exploited for obtaining faster circuits. *Cycle borrowing* is an example of such practice: if the critical path delay between two consecutive pipeline stages is not balanced, it is possible to skew the clock in such a way that the slower logic has more time to complete its computation, at the expense of the time available for the faster logic. For large and unstructured sequential networks, finding the best cycle borrowing strategy is a complex task that requires the aid of automatic tools.

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00©1996

2.1 Background

We will briefly review the basic concepts needed for the formal definition of the skew optimization problem. The interested reader can refer to [2, 9] for further information. Clock-skew optimization is achieved by assigning an arrival time to the local clock signals of each sequential element in the circuit. We consider raising-edge-triggered flip-flops and single clock. The clock period is T_{clk}. For the generic flip-flop i ($i = 1, 2, ..., N$, where N is the number of flip-flops in the network) we define its *arrival time* T_i, $0 \leq T_i < T_{clk}$. The arrival time represents the amount of skew between the reference clock and the local clock signal of flip-flop i. A *clock schedule* is obtained by specifying all arrival times T_i. Obviously not all clock schedules are valid. The combinational logic between the flip-flops has finite delay. The presence of delays imposes constraints on the relative position of the arrival times.

The classical clock-skew optimization problem can be stated as follow: *find the optimal clock schedule* $\mathbf{T} = [T_1, T_2, ..., T_N]$ *such that no timing constraint is violated and the cycle time* T_{clk} *is minimized.* This problem has been analyzed in detail and many solutions have been proposed. Here we summarize the work presented in [2] where edge-triggered flip-flops are considered.

We assume for simplicity that all flip-flops have the same setup and hold times, respectively called T_{SU} and T_{HO}. If there is at least one combinational path from the output of flip-flop i to the input of flip-flop j, we call the maximum delay on these paths $\delta_{i,j}^{max}$. The minimum delay $\delta_{i,j}^{min}$ is similarly defined. If no combinational path exists between the two flip-flops, $\delta_{i,j}^{max} = -\infty$ and $\delta_{i,j}^{min} = +\infty$. For each pair of flip-flops i and j, two constraints must be satisfied.

First, if a signal propagating from the output of i reaches the input of j before the clock signal for j is arrived, the data will propagate through two consecutive sequential elements in the same clock cycle. This problem is called *double clocking* and causes failure. The first kind of constraints prevents double clocking:

$$T_i + \delta_{i,j}^{min} \geq T_j + T_{HO} \tag{1}$$

On the other hand, if a signal propagating from i to j arrives with a delay larger than the time difference between the next clock edge on j and the current clock edge on i, the circuit will fail as well. This phenomenon is called *zero clocking*. Zero clocking avoidance is enforced by the following constraint:

$$T_i + T_{SU} + \delta_{i,j}^{max} \leq T_j + T_{clk} \tag{2}$$

Input and output impose constraints as well. Input constraints have the same format as regular constraints, where the constant value of the input arrival time T_{in} replaces the variable T_i. For output constraints the variable T_j is replaced by the constant output required time T_{out}.

The total number of constraint inequalities constructed by this method is $O(N^2 + I + O)$, where, I and O are the number of inputs and outputs respectively. In practice, this number can be greatly reduced. Techniques for the reduction of the number of constraints are described in [7, 1] and are not discussed here for space reasons.

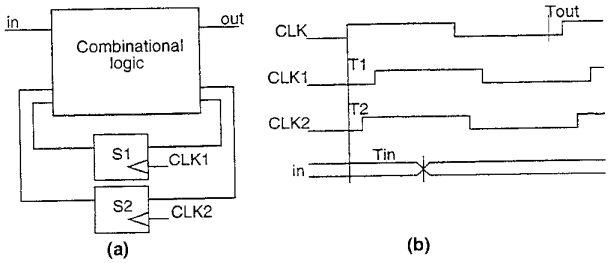

Figure 1: (a) Example circuit, with two flip-flops. (b) Timing waveform representing the skewed clocks.

Example 1 *We obtain the constraint equations for the circuit in Figure 1. There are two variables* T_1 *and* T_2*, representing the skew of the clocks CLK1 and CLK2. The clock period is* T_{clk}*. We assume that* $T_{SU} = T_{HO} = 0$*. The constraints for variable* T_1 *are the following:*

$$\begin{array}{lcl} T_1 + \delta_{1,2}^{max} & \leq & T_2 + T_{clk} \\ T_1 + \delta_{1,2}^{min} & \geq & T_2 \\ T_1 + \delta_{1,out}^{max} & \leq & T_{out} + T_{clk} \\ T_{in} + \delta_{in,1}^{min} & \geq & T_1 \end{array}$$

Moreover, $0 \leq T_1 \leq T_{clk}$*. Similar constraints hold for* T_2*. We have eliminated one input constraint and one output constraint because we assume that skews are positive and that the circuit with no skews was originally satisfying all input and output constraints. Notice that all constraints are linear. The feasibility of a set of linear constraints can be checked in polynomial time by the Bellman-Ford algorithm [16].*

Cycle time minimization is an optimization problem targeting the minimization of a linear cost function (i.e., $F(T_1, T_2, ..., T_N, T_{clk}) = T_{clk}$) of linearly constrained variables. It is therefore an instance of the well-known *linear programming* (LP) problem. Several efficient algorithms for the solution of LP has been proposed in the past [14]. Our problem is radically different and substantially harder. It can be stated as follows: *find a clock schedule such that the peak current of the circuit is minimum.* The cost function that we want to minimize is *not* linear in the variables T_i. In the following subsection, we discuss this issue in greater detail.

2.2 Cost function

Ideally, we would like to minimize the maximum current peak that the circuit can produce. This is however a formidable task, because such peak can be found by exhaustively simulating the system for all possible input sequences (and a circuit level simulation would be required, because traditional gate-level simulators do not give information on current waveforms). To simplify the problem, we make two important assumptions. First, we only minimize the current peak directly caused by clock edges (i.e., caused by the switching of clock lines and sequential elements' internal nodes and outputs). This approximation is justified by experimental evidence. In all circuits we have tested, the largest current peaks are observed in proximity of the clock edges. The current profile produced by the propagation of signals through the combinational logic is usually spread out and its maximum value is sensibly smaller.

Notice that we are *not* neglecting the combinational logic, but we consider its current as a phenomenon on

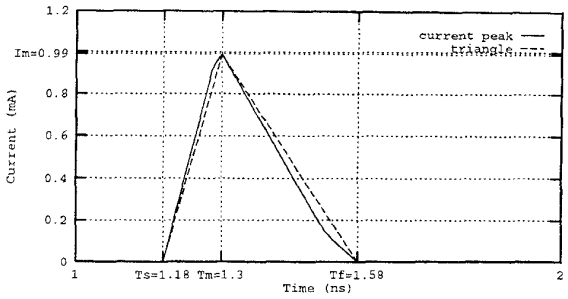

Figure 2: The four parameters characterizing the triangular approximation of the average current profile. t_s and t_e are the times at which the current reaches 1% of its maximum value.

which we have no control. Again, this choice is motivated by experimental evidence: our tests show that in most cases, the current profile of the combinational logic is not very sensitive to the clock schedule. For some circuits, the combinational logic may be dominant and strongly influenced by the clock schedule. We will discuss this case in a later section.

The second approximation regards the shape of the current waveform. Each sequential element produces two peaks, one related to the rising edge of the clock, and the other to the falling edge. We approximate the current peaks produced by each sequential element, (or group of sequential elements) with two triangular shapes, that are fully characterized by four parameters: starting time t_s, maximum time t_m, maximum value current I_m and final time t_f.

To compute these parameters we run several simulations with PPP [19] (see Section 4) and we obtain current waveform envelopes $I_{av}(t)$ ($I_{av}(t)$ is obtained by averaging the current at t on different input patterns). For each peak of the curve I_{av}, we define the four parameters as shown in Figure 2: t_s is the time at which the current first reaches 1% of the maximum value, t_f is the time at which the current decreases below 1% of the maximum value, I_m and t_m are respectively the maximum current value and the time when it is reached. Experimentally we observed that the triangular approximation is satisfying for the current profiles of the sequential elements. For combinational logic, this approximation is reasonable but less accurate.

The total current is the sum of the current contributions represented as triangular shapes. Every flip-flop i has two associated contributions $\Delta_i^r(t, T_i)$ and $\Delta_i^f(t, T_i)$, representing respectively the current drawn on the raising and falling edge of the clock. Notice that such contributions are functions of time t and of the clock arrival time T_i. In fact, the curve translates rigidly with T_i. The current drawn by the combinational logic is approximated with a triangle $\Delta_C(t)$. $\Delta_C(t)$ is *not* a function of the arrival time of any clock. The total current is the sum of the contributions due to flip-flops and combinational logic:

$$I_{tot}(t, \mathbf{T}) = \Delta_C(t) + \sum_{i=1}^{N} \Delta_i^r(t, T_i) + \sum_{i=1}^{N} \Delta_i^f(t, T_i) \quad (3)$$

The cost function F that approximates the peak current is:

$$F(\mathbf{T}) = \max_{t \in [0, T_{clk}]} \{I_{tot}(t, \mathbf{T})\} \quad (4)$$

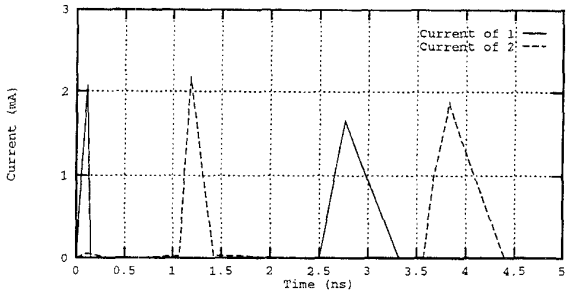

Figure 3: Current profiles for the two flip-flops 1 and 2 from PPP simulation of our example circuit.

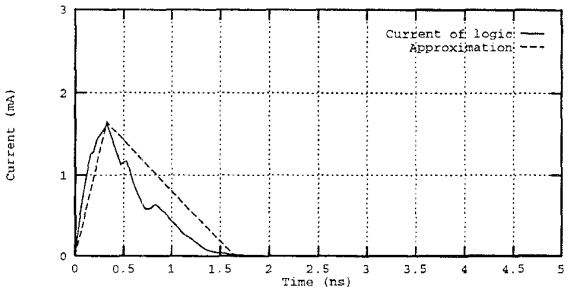

Figure 4: Current profile corresponding to the combinational logic from PPP simulator of our example circuit. The dashed line is its triangular approximation

Our target is to find the optimum clock schedule $\mathbf{T_{opt}}$ which minimizes the cost function F. We clarify these definition through an example.

Example 2 *The current profiles for the flip-flops of the circuit in Figure 1 are shown in Figure 3 for one assignment of T_1 and T_2. The current profile of the combinational logic for this example is shown in figure 4 with its approximation triangle.*

The contribution of a flip-flop is approximated by two triangular shapes. The first corresponds to the rising edge of the clock, the second to the falling edge. Here we have $T_1 = 0ns$ and $T_2 = 1.07ns$. Notice that the current profile of flip-flop 2 is shifted to the right. The profiles for the two flip-flop do not have exactly the same shape because they are differently loaded.

Notice that when $T_1 = T_2$ the two current profiles of the flip-flops are perfectly overlapped. When $T_1 \neq T_2$, the two contributions are skewed. The value of the cost function $F(T_1, T_2)$ is the maximum value of the sum of the five triangles. Here $F(0, 1.07) = 2.7$, whereas initially, $F(0, 0) = 4.2$.

3 Peak current minimization

We now describe our approach to the minimization of the cost function described in the previous section. The first key result of this section is summarized in the following proposition stated without proof.

Proposition 1 *The cost function F of equation 4 can be evaluated in polynomial time (in the number of triangular contributions).*

We developed a $O(N_\Delta^2)$ algorithm (N_Δ is the number of triangular current contributions) for the evaluation of

```
/* Let T[i] (i..N) be the variable vector */
/* Delta_orig[i] (i..2N+1) are the 2N+1 contributions when T[i]=0 */
float evaluate (T)
  /* computes the contributions for the vector T */
  Delta = translate_triangles (Delta_orig, T);
  max = 0;
  foreach (c1 in [0..2N])
    val = max(Delta([c1]));
    foreach (c2 in [0..2N])
      if (c2 != c1) then
        if (overlap (Delta[c1], Delta[c2])) then
          /* we look if the 2 triangles overlap and add the value */
          /* of c2 at the maximum point of c1 */
          max += get_value (Delta[c2], time_max (Delta[c1]));
        endif;
      endif;
    endfor;
    if (val ≥ max) then max = val;
  endfor;
  return (max);
end evaluate;
```

Figure 5: $O(N^2)$ algorithm for the computation of the cost function F.

the cost function. The algorithm is based on the observation that the maximum of the cost function can be attained in a finite number of points, namely the points of maximum of the triangles that compose it. In order to evaluate the value of F in one of such points, we must check if the corresponding triangle is overlapping with any of the other contributions. The quadratic complexity stems from this check. The pseudo-code of the algorithm is shown in Figure 5.

The second key result is summarized by the following proposition:

Proposition 2 *The peak current minimization problem is an instance of the* **constrained DC optimization problem***[15]*[1].

An important consequence of this theorem is the NP-completeness of the current minimization problem (since DC optimization is NP-complete). Our solution strategy is heuristic and it is based on the genetic algorithm (GA) [17]. Notice that the GA approach is attractive in our case because we have an efficient way to compute the cost function.

GA-based functional optimization requires a very large number of function evaluations (proportional to the number of generations multiplied by the size of the population). Since F can be efficiently evaluated, large instances of the problem can be (heuristically) solved.

Notice two important facts. First, our algorithm heavily relies on the triangular approximation. If we relax this assumption, the evaluation of F becomes an extremely complex problem (finding the maximum of a multi-modal function), and the GA approach would not be practical. Second, we consider the contribution of the combinational logic as function of time only (independent from the T_i). As a consequence, if the maximum current is produced by the combinational logic, $F(T_1, ...T_N)$ is a constant, and no optimization is achievable.

3.1 Clustering

Up to now, we have assumed that the arrival time T_i of each individual flip-flop can be independently controlled. This is an unrealistic assumption. In an actual VLSI circuit the clock is distributed using regular

[1] Problems where the cost function can be expressed as the difference of two concave functions. See [15] for a detailed treatment.

```
/* Let F[i] (i..N) be the instances of the flip-flops */
/* Let T[i] (i..N) be the values given by the GA for instance i */
/* Let N_p be the number of clusters to obtain */
F_sort[i] = sort_by_skew (F[i], T[i]);
size_cluster = N / N_p;
num_cluster = 0;
foreach (i in F_sort[i])
  if (size (Cluster [num_cluster] == size_cluster)) then
    num_cluster++;
  endif;
  add_in_cluster (Cluster [num_cluster], F_sort[i]);
endfor;
return (Cluster);
```

Figure 6: Clustering algorithm.

structures such as *clock trees* [10]. Usually, sub-units of a complex system have local clocks, connected with buffers (drivers) to the main clock tree. The buffers are the ideal insertion points for the delays needed for skew optimization (for practical implementations of such delays refer for example to [4, 3]). In general it would not be feasible to provide each flip-flop with its own buffer and delay element, for obvious reasons of layout complexity, routability and power dissipation.

Since clock-skew optimization is practical only if applied at a coarser level of granularity, we have developed a strategy that allows the user to specify the number of clusters (i.e., the number of available clock buffers with adjustable delay), and heuristically finds flip-flops that can be clustered without large penalty on the cost function. Here we assume that no constraints on the grouping of flip-flops have been previously specified. This is often the case for circuits generated by automatic synthesis. Structured circuits (data-path, pipelined systems) with pre-existing clustering constraints are discussed later.

Our clustering algorithm can be summarized as follows. The user specifies the number of clusters N_P. First, we solve the peak current minimization problem without any clustering (every flip-flop may have a different arrival time). We then insert the flip-flops in a list ordered by clock arrival times. The list is partitioned in N_P equal blocks. New constraint equations and new current profiles are obtained for the blocks of the partition. A new peak current minimization is solved where the variables are the arrival times T_j^P, $j = 1, 2, ..., N_P$, one for each cluster. We also recompute the delays from cluster i to cluster j. The number of equations reduces to $O(N_P^2 + I + O)$. The pseudo-code of the clustering algorithm is shown in Figure 6.

Using clustering, we can control the granularity of the clock distribution. The first step of our partition strategy is based on the optimal clock schedule found without constraint on the number of partitions. Clustering implies loss in optimality, because some degrees of freedom in the assignment of the arrival times are lost. Our clustering strategy reduces the loss by trying to enforce a natural partitioning. The second iteration of current peak optimization guarantees correctness and further reduces the optimality loss.

Example 3 *Consider the small benchmark* s208. *It consists of 84 combinational gates and 8 flip-flops. The cycle time is* 10*ns, the clock has 50% duty cycle. The current profile for the circuit is shown in Figure 7 with the dashed line. Observe the two current peaks synchronized with the raising and falling edge of the clock. The irregular shape that follows the first peak shows the current drawn by the combinational logic.*

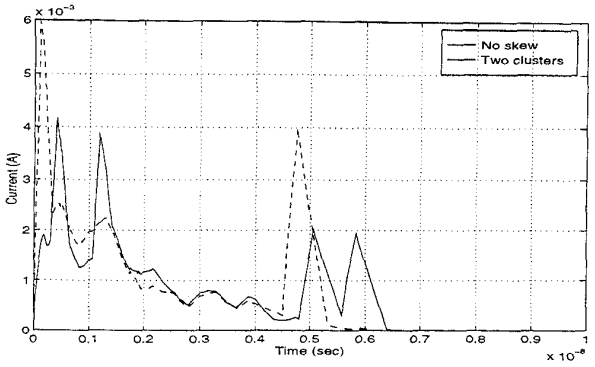

Figure 7: Current profile for benchmark s208 before and after skew optimization with two clusters. The current profiles are obtained by accurate current simulation

The skew is then optimized with the constraint of 2 partition blocks (i.e., two separate clock drivers allowed). The current profile after skew optimization is shown in Figure 7 with continuous line. The beneficial effect of our transformation is evident. Notice also that the current profile of the combinational logic is more spread out, but the overall contour is almost unchanged.

3.2 Dealing with combinational logic

In the previous sections, we have solved the current peak optimization problem assuming that we cannot control the current profile of the combinational logic. For many practical circuits this is an overly pessimistic assumption, because the data path of large synchronous systems is often *staged*. In a staged structure, a set of flip-flops A feeds the inputs of a combinational logic block. The output of the block are connected to the input of a second set of flip-flops B. The sets A and B are disjoint. The flip-flops in A and the block of combinational logic are called a *stage*. Pipelined circuits are staged, and most data paths have this structure, that makes the design easier and the layout much more compact.

If the circuit has a staged structure, the behavior of the combinational logic is much more predictable. If we cluster the flip-flops at the input of each stage, by imposing the same arrival time (i.e., assigning the same clock driver) to their clock signal, we can guarantee that all inputs of the combinational logic of the stage are synchronized. As a consequence, the current profile of the combinational logic translates rigidly with the arrival time of the clock of the flip-flops at its inputs.

For staged circuits our algorithm is more effective, because the clock schedule controls the current profile of the combinational logic as well. The current peak can therefore be reduced even if it is entirely dependent on the combinational logic [21].

4 Implementation and results

The implementation of a program for peak current minimization depends on the availability of a tool that provides accurate current waveforms for circuits of sufficiently large size. Electrical simulators such as SPICE are simply too slow to provide the needed information. In our tool, current waveforms are accurately estimated by an enhanced version of PPP [19], a multi-level simulator specifically designed for power and current estimation [20] of digital CMOS circuits. PPP has performance similar to logic level simulators, it is fully compatible with Verilog XL and provides power and current data with accuracy comparable to electrical simulators. Input signal and transition probabilities for all the simulations are set to 50%.

The starting point for our tool is a mapped sequential network (we accept Verilog, SLIF and BLIF netlists). First, the sequential elements are isolated and current profiles are obtained. Alternatively, pre-characterized current models of all flip-flops in the library can be provided. The combinational logic between flip-flops is then simulated and its average current profile is obtained. The first simulation step assumes no skews.

Timing information is extracted from the network. Maximum and minimum delays are estimated with safe approximations (i.e., topological paths). Input arrival times and output required times are provided by the user. The constraint inequalities are then generated. In this step several optimizations, such as those described in [7, 1], are applied to reduce the number of constraint inequalities. Data needed for the evaluation of the cost function are produced: the triangular approximations are extracted from the current profiles and passed to the GA solver [18].

The GA solver then is run to find the optimal schedule that minimizes the peak current. The initial population is generated by perturbing an initial feasible solution (zero skew). The GA execution terminates after an user-specified number of generations. The resulting optimal clock schedule is then applied in a last simulation pass, where the effect on current peaks and average power dissipation is evaluated.

If the maximum number of clock drivers has been specified, the tool first clusters the solution with the algorithm described in Figure 6, then it runs another simulation to obtain the new current profiles for the clusters (which are now regarded as atomic blocks). A second GA run is performed to re-optimize the clustered solution. Finally, simulation is repeated to check the quality of the result.

The results on a set of benchmark circuits (from the MCNC suite) are reported in Table 1. The first two columns represent the name of the circuit and the number of flip-flops. For each of the following columns, two rows are reported for each benchmark. The first row refers to the results obtained with no clustering (i.e., clusters of size 1), the second lists the results obtained with the number of partitions reported in column four. For the biggest benchmarks $s15850$ and $s13207$, we initially clustered the flip-flops to reduce the complexity of the first execution of the GA. Columns four, five and six describe the effect of clock-skew optimization on average power dissipation. The last three columns describe the effect on current peaks. Without clustering, we reduced on average the current peak by 39%. When we constraint the number of clock drivers, we reduce it by 27%.

We were concerned about a possible increase in power dissipation inside the combinational logic due to unequal arrival times of the clocks controlling flip-flops at its inputs (i.e., increased glitching). From the analysis of the results it appears that skew optimization does not have a sizable impact on average power dissipation. On the other hand, the effect on current peaks is always

positive, and often very remarkable. For some circuits, current peaks are reduced to less than a half the original values. The range in quality of the results is due to the relative importance of the current in the combinational logic. For circuit where the current peak produced by the combinational logic is close to that produced on the clock edges, only marginal improvements are possible. Notice however that some improvements have always been obtained even for small circuits with few flip-flops. This result may seem surprising and warrants further explanation. In the combinational logic, signals propagate through cascade connections of gates, therefore only a relatively small number of logic gates is switching at any given time. In contrast, on a clock transition (with zero skew) all flip-flips switch and all gates directly connected to them draw current approximatively at the same time.

The running time of the algorithm is dominated by the first skew optimization step and it ranges from a few minutes to one hour (on a DEC station 5000/240). In average, the simulation time is approximatively 40% of the total. A larger fraction (55% in average) is spent in the GA solver. The remaining 5% is spent in generating the constraints and parsing the files. When the clustered solution is simulated and optimized, the speedup is almost linear in the size of the clusters.

Bench	FF	P	AvgPower (microW)			Current peak (mA)		
			before	after	ratio	before	after	ratio
s15850	550	90	46732	46342	0.992	320	176	0.550
		20	46731	46717	1.000	320	219	0.680
s13207	490	80	52094	48476	0.931	267	165	0.619
		20	52094	48856	0.938	267	196	0.733
dsip	224	224	70081	70038	0.999	270	230	0.852
		20	70081	69720	0.995	270	240	0.889
s5378	163	163	71565	72813	1.017	99.3	60.6	0.610
		15	71587	72420	1.012	99.5	75.0	0.754
s9234	135	135	19364	20154	1.041	49.7	12.0	0.241
		20	19364	20619	1.065	49.7	18.0	0.362
mm30a	90	90	14141	14040	0.993	163	138	0.846
		9	14141	14239	1.007	163	131	0.807
s1423	74	74	8043	7276	0.905	50.0	22.2	0.444
		10	8043	7965	0.990	50.0	35.1	0.702
mult32b	61	61	71970	72462	1.007	40.7	24.5	0.602
		7	71970	71944	1.000	40.7	28.2	0.693
sbc	27	27	34285	34178	0.997	47.4	40.8	0.861
		4	34285	34619	1.010	47.4	43.2	0.911
s400	21	21	9773	9751	0.998	12.6	7.94	0.630
		3	9777	9854	1.008	12.6	10.6	0.844
s208	8	8	4207	4095	0.973	5.98	3.19	0.533
		2	4207	4077	0.969	5.98	3.86	0.645

Table 1: Results of our procedure applied to MCNC benchmarks.

5 Conclusions and future work

We proposed a new approach for minimizing the peak current caused by the switching of the flip-flops in a sequential circuit using clock scheduling. We reduced on average 30% of the peak current of the circuit without any increase of power consumption. Moreover, the initial clock frequency of the circuit was preserved. We showed that linear programming approaches traditionally used for clock scheduling are not suitable, and we proposed a heuristic solution strategy based on the genetic algorithm. Clustering techniques have been introduced to account for constraints on the maximum number of available clock drivers.

Although we conservatively assumed that we have no control on the current profiles of the combinational logic, this assumption can be relaxed for staged circuits. In such circuits, the combinational logic can be clustered with the sequential elements. In this case the peak current of the combinational logic plays a role in the cost function of the peak reduction algorithm: the waveform of this combinational logic would be shifted if the clock schedule changes. Clock skewing in this case would also reduce the current peak caused by combinational logic, therefore allowing more effective minimization.

References

[1] T. G. Szymanski, "Computing Optimal Clock Schedules" *DAC, Proceedings*, pp. 399–404, 1992.

[2] J. P. Fishburn, "Clock Skew Optimization" *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 945–951, July 1990.

[3] Jun-Dong Cho, Majid Sarrafzadeh, "A Buffer Distribution Algorithm for High Performance Clock Net Optimization" *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, March 1995.

[4] Ren-Song Tsay, "An Exact Zero-Skew Clock Routing Algorithm" *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 12, no. 2, February 1993.

[5] N.-C. Chou et al., "On general zero-skew clock net construction," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, March 1995.

[6] T. G. Szymanski, N. Shenoy, "Verifying Clock Schedules" *Proceedings of ICCAD 1992*, pp. 124–131.

[7] N. Shenoy, R. Brayton, A. L. Sangiovanni-Vincentelli "Graph Algorithms for Clock Schedule Optimization" *Proceedings of ICCAD 1992*, pp. 132–136.

[8] T. M .Burks, K. A. Sakallah, "Min-Max Linear Programming and the Timing Analysis of Digital Circuits" *Proceedings of ICCAD 1993*, pp. 152–155.

[9] K. A. Sakallah, T. N. Mudge, O. A. Olukotun "Analysis and Design of Latch-Controlled Synchronous Digital Circuits" *IEEE Transactions on CAD*, vol. 11, no. 3, March 1992.

[10] M. Horowitz, "Clocking Strategies in High Performance Processors" *1992 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 50–53.

[11] A. Ohba, H. Sato et. al., "New Decoding Architecture to Reduce Peak Current and Its Implementation to 4M ECL SRAM" *1992 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 30–31.

[12] Actel, *FPGA databook and design guide*, 1994.

[13] S. Chowdury and J. Barkatullah, "Estimation of Maximum Currents in MOS IC Logic Circuits," *IEEE Transaction on CAD*, vol. 9, no. 6, pp. 642 – 654, 1990.

[14] K. G. Murty, *Linear Programming*, Wiley, 1983.

[15] R. Horst, P. M. Pardalos, ed. *Handbook of global optimization*, Kluwer, 1995.

[16] E. Lawler, *Combinatorial optimization: networks and matroids*, Holt, Rinehard and Winston, 1976.

[17] D. Goldberg, *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.

[18] J. J. Grefenstette, *A user's guide to GENESIS*, 1990.

[19] A. Bogliolo, L. Benini, and B. Riccò, "Power Estimation of Cell-Based CMOS Circuits," *DAC*, 1996.

[20] A. Bogliolo, L. Benini, G. D. Micheli, and B. Riccò, "Gate-level current waveform simulation," *ISLPD*, 1996.

[21] J. Yoo, G. Gopalakrishnan et al., "High speed Counterflow-Clocked pipelining illustrated on the design of HDTV subband vector quantizer chips," Advanced Research on VLSI, Chapel Hill, 1995.