Power Reduction in Pipelines

Sri Parameswaran

Dept. of Computer Science & Electrical Engineering
The University of Queensland
Brisbane, Qld 4072
e-mail: sridevan@elec.uq.edu.au

Hui Guo

Dept. of Computer Science & Electrical Engineering
The University of Queensland
Brisbane, Qld 4072
e-mail: guo@elec.uq.edu.au

Abstract— The reduction of power consumption for a system level pipeline is addressed in this paper. The pipeline is composed of several stages. Each stage has several behaviours. Different behaviours have differing execution times. The speed of the pipeline is only affected by the behaviours on the critical path of the slowest stages. Other behaviours can be slowed down to decrease the power consumed in the system. We propose a multi–voltage supply scheme, in which differing behaviours are supplied with differing voltages. The formulas for computing the supply voltage of each behaviour and minimal power consumption are derived in this paper. The results of computer experiment show that up to 80% hardware power can be saved with this scheme.

I. Introduction

System level pipelines contain several behaviours. These behaviours can be *grouped* [7] into stages which make a pipeline. Each stages in the pipeline is expected to have same execution time to obtain high performance. The speed of the pipeline is determined by the stage with the greatest execution time. If a stage has an execution time of less than t_M (t_M = event arrival time), it will mean poor resource utilization. However, if we slow down these fast behaviours by reducing the supply voltage (which would slow the clocking frequency), the power consumption of the system can be minimized. Without showing the overall system, an example is given below.

Figure 1(a) is a pipeline stage of four behaviours. The general relationship of power versus $1/f$ (f = frequency of a behaviour) under different supply voltages is illustrated by the curve, called *power characteristics curve*, shown in Figure 1(b). That is, the increase of supply voltage will result in the increase of power consumption and the decrease of execution time. Figure 2 shows the histogram of power dissipation of the system under three situations. Each bin corresponds to a behaviour in a system. The height of the bin stands for the behaviour power consumption and the width for the execution time. When the supplying voltage is changed, both of the height and width will be changed. The power consumed by the system is the sum of those consumed by each behaviour and also the whole execution time is the sum of each behaviour execution time. The pipeline stage execution time is given by t_M.

Under the normal situation (i.e., the stage is supplied by 5 volts), the power consumed by the four behaviours are 7, 11, 6 and 4, respectively, as illustrated in Figure 2(a). The system power consumption is then

$$\begin{aligned} P &= P_1 + P_2 + P_3 + P_4 \\ &= 7 + 11 + 6 + 4 = 28, \end{aligned}$$

and the execution time of the system is

$$t = t_1 + t_2 + t_3 + t_4.$$

If t is now smaller than the pipeline stage execution time, t_M, then the stage is allowed to slow down by using lower supply voltage. Assuming the stage is supplied only by a single voltage, we could reduce the voltage and get

$$t = t_M;$$

and using this supply voltage, the power consumption of each behaviour implementation is reduced, and the power consumption of the stage is given by,

$$P = 6 + 9 + 5 + 3.5 = 23.5,$$

as shown in Figure 2(b).

If each behaviour implementation can be supplied by a separate voltage (which is optimal for the whole stage) with $t = t_M$ the power consumption can be reduced even further, as shown in the Figure 2(c), where power consumption is only

$$P = 4 + 8 + 3 + 4 = 19.$$

Work on low power consumption systems can be found in many publications [2] [3] [1] [15] [14] [4] [10]. Approaches to reduce power dissipation in the system can be found in paper [9]. However, effort on multiple voltage for power reduction is limited [13] [8] [5]. Some work is on the predetermined set of supply voltages [13], some is by using programming approach to choose supply voltage. Mark Johnson and Kaushik Roy[8], have proposed

0-7803-4425-1/98/$10.00 1998 IEEE

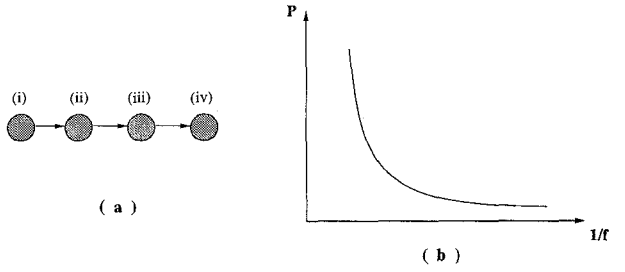

Fig. 1. A Stage With Four Blocks

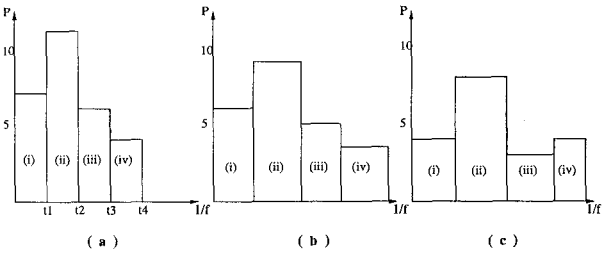

Fig. 2. Power-Delay Histogram Under (a) normal (b) reduced uni-supply-voltage (c) reduced multi-supply-voltage Conditions

an algorithm to minimize data-path energy dissipation through use of multiple supply voltages.

In this paper, an optimal method to determine supply voltage of each behaviour for a minimal power consumption (given a t_M) is given.

The rest of the paper is arranged as: definitions and concepts are given in the next section; formulas for minimal power consumption are derived in in section 3; an algorithm for power reduction in a pipeline is proposed in section four; the results for several benchmarks are given in section 5 and conclusions are drawn at the end of this paper.

II. Concepts

To make the rest of the paper easy to read, some definitions and concepts are given below.

Power consumed by a circuit is given by

$$P = C_L V_{dd}^2 f;$$

where V_{dd} is the supply voltage. The voltage is inversely proportional to the delay (the critical path delay T_d) of the circuit. At limits of operation the clock frequency of the circuit will be $1/T_d$. If the frequency is to be some percentage of $1/T_d$, then $frequency \propto 1/T_d$.

The power then can be said to be proportional to the voltage cubed or inversely proportional to the critical path delay cubed. Therefore at the limit of operation, power can be evaluated by

$$P = k/T_d^3,$$

and the delay can be referred to as *execution time* of the circuit.

Pipeline Stage Execution Time t_M: The maximal execution time of the stages in a pipeline, denoted by t_M.

Stage Graph, G_s: A digraph represents a stage, where all nodes stand for behaviours in the stage and edges for data dependency between behaviours.

Path: A set of nodes which are connected in series.

Independent Path: A path on which no nodes are adjacent to or from other nodes on other paths. That is, both indegree and outdegree of the nodes on the path are smaller (for initial node and end node) or equal to one.

Adjacent Path: A path which is adjacent to other paths in the graph as compared to the independent paths.

Tight Path: A path on which nodes are connected tightly in terms of behaviour duty time. That is, on the path, data from one behaviour will immediately go to the next behaviour without any delay. An independent path is always a tight path.

Loose Path: A path on which nodes are loosely connected in terms of the duty time. Some fast behaviours need to wait until the next behaviours are un-occupied.

Path Length: Sum of the execution times of the behaviours on the path.

III. Power Reduction for a Path of Blocks

A. Power Estimation for a Path of Blocks

Supposing a path has m behaviours. Each behaviour is implemented by a separate circuit with its own voltage supply and is working at the limits of operation, then the power consumed by the path is

$$\begin{aligned} P &= P_1 + P_2 + \ldots + P_m \\ &= k_1/{T_{d1}}^3 + k_2/{T_{d2}}^3 + \ldots + k_m/{T_{dm}}^3 \\ &= \sum_{i=1}^{m} \frac{k_i}{T_{di}^3}, \end{aligned} \tag{1}$$

where T_{dj} is the execution time of j-th behaviour.

The total execution time needed is

$$t_d = T_{d1} + T_{d2} + \ldots + T_{dm}. \tag{2}$$

B. Minimal Power

Assuming the path execution time is t_d and it is limited by t_M, then the difference

$$D = t_M - t_d, \tag{3}$$

called *slack time* can be used to reduce the power consumption by lowering the supply voltages. To optimally reduce the power with the given *slack time* is a constraint minima problem.

Rewrite Formula 1 as

$$P = \sum_{i=1}^{m} \frac{k_i}{(t_i + x_i)^3} \tag{4}$$

$$\delta = \sum_{i=1}^{m} x_i - D = 0 \tag{5}$$

where t_i is original execution time, and x_i gives the amount of execution time to be changed.

To obtain the minimal value of P under condition (5), the following group equations must be satisfied [12].

$$\frac{\partial P}{\partial x_i} + \lambda \frac{\partial \delta}{\partial x_i} = 0 \qquad i = 1, 2, \ldots, m. \tag{6}$$

From the formula, we obtain

$$\frac{1}{t_i + x_i} = \frac{P^{1/4}}{{k_i}^{1/4}(\sum_{j=1}^{m} t_j + D)^{1/4}} \qquad i = 1, 2, \ldots, m; \tag{7}$$

In equation (4), P is given as

$$P = \sum_{i=1}^{m} k_i \frac{P^{3/4}}{{k_i}^{3/4}(\sum_{j=1}^{m} t_j + D)^{1/4})^3}. \tag{8}$$

The minima of P is

$$P_{min} = \frac{(\sum_{j=1}^{m} k_j^{1/4})^4}{{t_M}^3}, \tag{9}$$

and the supply voltages [6] for each behaviour are

$$V_{ddi} = V_t + \frac{k_{ti}}{t_i + x_i} \qquad i = 1, 2, \ldots, m \tag{10}$$

$$= V_t + k_t \frac{k_i^{3/4}(\sum_{j=1}^{m} k_j^{1/4})}{t_M^{3/4}(\sum_{j=1}^{m} t_j + D)^{1/4}}, \tag{11}$$

where V_t is the threshold voltage of the circuit and k_{ti} a constant.

IV. Algorithm for Power Reduction in a Pipeline

Given a pipeline, power reduction is done on the stages. The stage power reduction is, in turn, fulfilled by minimizing the power consumption on the paths.

As defined in section 2, there are several kinds of paths in a stage graph. Stage power reduction begins from the path which is either an independent path or a tight path. Loose paths will be segmented into a number of tight paths. Reduction is always applied to the tight and longer paths since the loose path and shorter paths will bring about time conflict during power reduction. The explanation is given in the following example.

Figure 3 is a stage graph. Nodes 1, 4, 7 are *initial nodes*, nodes 3, 6, 9 are *end nodes*. Path 1, 1 → 2 → 3, is an independent path. Path 2, 4 → 5 → 6, and path 3, 7 → 5 → 8 → 9, are adjacent paths. They are connected though node 5. These 3 paths cover all nodes in the graph. The set of paths can be used in power reduction by applying the formula given in section 2 to the paths. The execution time for each behaviour is shown below each behaviour. It can be seen that of the two adjacent paths, path 2 is short and loose path, the task on this path is finished on time slot 17, and path 3 is long and tight path, with a finishing time slot of 25. Assume the stage execution time, t_M, is 28 and path 2 is *power-reduced* first. This could lead to an optimal power reduction in path 2, but the execution time of path 3 might be greater than t_M.

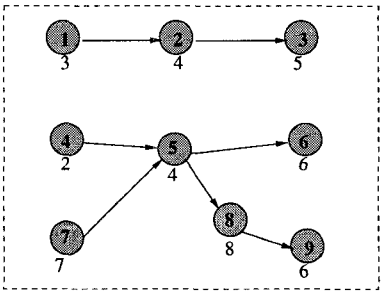

Fig. 3. *A Stage Graph*

For a *intersecting behaviour*, the power reduction is done from the longest adjacent path. After power reduction on the longest path, the rest of adjacent paths are broken up at the intersecting behaviours.

The whole algorithm of power reduction for a pipeline is given as follows.

n_s: number of stages in the pipeline;
G_s: stage graph;
t_M: stage execution time;
$P(n1, n2)$: tight path from node n1 to node n2;
t_m: the maximal execution time allowed for a path;
t_1: start time of a behaviour;
t_2: end time of a behaviour;

for *all stage, i*
/* get stage graph for stage i from the system graph */
G_s == stage_extract(i);
while $G_s \neq \phi$
/* find the longest path in G_s */
P(n1, n2)=longest_path(G_s);
/* determine the maximal execution allowed for the path */
if *n2 is ending node*
tm = t_M-*t1(n1)*;
else
tm = *t2(n2)-t1(n1)*;
end if
/* apply power reduction formula to the path the length of the path will be increased to *tm* */
PowRed(P, tm);
/* delete path P from G_s. Some adjacent paths will break up at intersect node and the start time and end time of all unprocessed nodes adjacent to the intersection node will be changed */

delete(P);
end while
end for

V. Results

The power reduction algorithm was programmed in C language and was run on several benchmarks. Some of the benchmarks were borrowed from the early paper[7] and pipelined by the method given in[11].

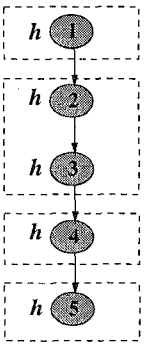

Fig. 4. Benchmark 1

TABLE I
Data for Benchmark 1

behaviour	delay1	V_{dd1}	delay2	V_{dd2}
1	89.0	5.0	89.0	5.0
2	65.0	5.0	52.6	6.0
3	15.0	5.0	36.4	2.5
4	68.0	5.0	89.0	4.0
5	36.0	5.0	89.0	2.5

Figure 4 is a simple system with 5 behaviours connected in series. All behaviours in the system were partitioned into hardware and pipelined into four stages as illustrated by the dashed lines. The system was uniquely supplied by 5 volts, and the execution time for each behaviour is listed under *delay1* column in Table I. The columns V_{dd2} and *delay2* show the voltage and execution time of each behaviour computed for minimal power. With these values, the power consumption can be reduced from 0.685939 units to 0.124112 units.

Figure 5 is a system with 13 behaviours. The system was partitioned and pipelined into 8 stages with the stage execution time being 385. It is noted that some behaviours were allocated into software. These behaviours with software implementation were left alone without any action by the algorithm. Only behaviours with hardware implementations were considered power reducible. Originally, the power consumption was 0.051644 units and each behaviour was supplied by the same voltage. After applying the power reduction method, each hardware

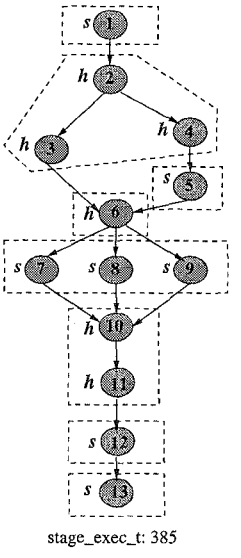

Fig. 5. Benchmark 2

TABLE II
Data for Benchmark 2

behaviour	delay1	V_{dd1}	delay2	V_{dd2}
1	333.0	5.0	333.0	5.0
2	68.0	5.0	172.3	2.4
3	158.0	5.0	212.7	4.0
4	121.0	5.0	212.7	3.1
5	385.0	5.0	385.0	5.0
6	333.0	5.0	385.0	4.4
7	147.0	5.0	147.0	5.0
8	175.0	5.0	175.0	5.0
9	321.0	5.0	321.0	5.0
10	122.0	5.0	182.6	3.6
11	184.0	5.0	202.4	4.6
12	271.0	5.0	271.0	5.0
13	152.0	5.0	152.0	5.0

implemented behaviour was supplied by the voltage as indicated by column V_{dd2} in Table II. And the power dissipation was reduced to 0.010642 units.

In benchmark 3 shown in Figure 6, with the new supply voltages shown in Table III, the system power dissipation can be reduced from 0.587834 units to 0.360099 units.

For the system shown in Figure 7, there is only one behaviour allocated into hardware. However, by applying 3.7 volts to that behaviour implementation, the power consumption was reduced from 0.142687 units to 0.046971 units.

For the system shown in Figure 8, if the supply voltages are changed from V_{dd1} to V_{dd2} shown in Table V, the system power can be reduced from 0.179883 units to 0.056497 units.

The power reduction algorithm on the five systems are concluded in Table VI.

where P_{before} stands for the power consumed before the reduction and P_{after} stands for that consumed after the reduction. The values in the column *saving* represent the

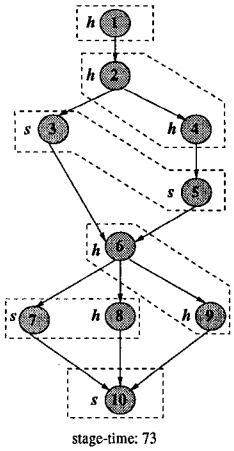

Fig. 6. Benchmark 3

TABLE III
DATA FOR BENCHMARK 3

behaviour	delay1	V_{dd1}	delay2	V_{dd2}
1	56.0	5.0	73.0	4.0
2	37.0	5.0	37.6	4.9
3	60.0	5.0	60.0	5.0
4	29.0	5.0	35.4	4.2
5	69.0	5.0	69.0	5.0
6	23.0	5.0	33.3	3.7
7	73.0	5.0	73.0	5.0
8	58.0	5.0	73.0	4.1
9	46.0	5.0	40.0	5.7
10	57.0	5.0	57.0	5.0

percentage of the reduced power over the original consumptions. It can be seen that by optimally selecting a set of supply voltages for each behaviour, up to 60% of the power can be saved.

VI. CONCLUSION

The method for reducing power dissipation in a system level pipeline is proposed in this paper. In a system level pipeline, some fast parts or parts not on the critical path are allowed reduced supply voltages, without overall degradation of performance of the pipeline. This reduction is possible as the speed of a pipeline is only determined by the critical components. With the reduction of supply voltage the power dissipation can be reduced.

The power reduction is done assuming that each behaviour is implemented by a separate circuit and each circuit is supplied by a separate voltage. The formulas of computing minimal power and related supply voltages are given in this paper. The power reduction in a pipeline can be performed by applying these formulas to the paths, which only covers all behaviours in a stage. An algorithm for reducing power is also given in this paper. The results by the algorithm run on several benchmarks show that with this method the power can be reduced by up to 80% of power.

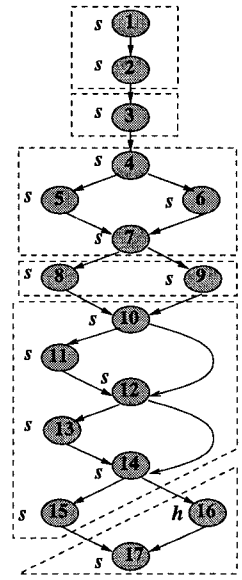

Fig. 7. Benchmark 4

REFERENCES

[1] L. Benini and G. De Micheli. State assignment for low power dissipation. *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, pages 136–9, 1994.

[2] Anantha P. Chandrakasan, Miodrag Potkonjak, Renu Mehra, Jan Rabaey, and Robert W. Brodersen. Optimizing power using transformations. *IEEE Transactions on Computer-Aided Design on Integrated Circuits and Systems*, Vol. 14, No. 1:12–31, Jan, 1995.

[3] Anantha P. Chandrakasan, Miodrag Potkonjak, Jan Rabaey, and Robert W. Brodersen. Hyper-lp: A system for power minimization using architectural transformations. In *1992 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 300–9, 1992.

[4] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen. Low-power cmos digital design. *IEEE Journal Of Solid-State Circuits.*, Vol. 27(No. 4):473–484, April, 1992.

[5] J. Chang and M. Pedram. Energy minimization using multiple supply voltages. In *Proceedings of International Symposium on Low Power Electronics and Design*, pages 157–162, 1996.

[6] Hui Guo and Sri Parameswaran. An estimation model for low-power–department research report.

TABLE IV
DATA FOR BENCHMARK 4

behaviour	delay1	V_{dd1}	delay2	V_{dd2}
1	11.0	5.0	11.0	5.0
2	31.0	5.0	31.0	5.0
3	48.0	5.0	48.0	5.0
4	35.0	5.0	35.0	5.0
5	13.0	5.0	13.0	5.0
6	13.0	5.0	13.0	5.0
7	5.0	5.0	5.00	5.0
8	50.0	5.0	50.0	5.0
9	62.0	5.0	62.0	5.0
10	5.0	5.0	5.0	5.0
11	9.0	5.0	9.0	5.0
12	5.0	5.0	5.0	5.0
13	15.0	5.0	15.0	5.0
14	5.0	5.0	5.0	5.0
15	12.0	5.0	12.0	5.0
16	29.0	5.0	42.0	3.7
17	20.0	5.0	20.0	5.0

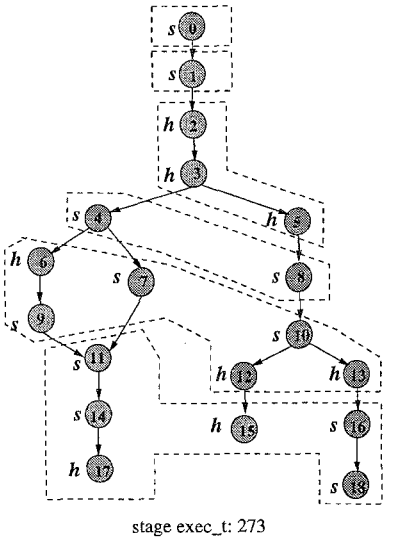

Fig. 8. Benchmark 5

TABLE V
DATA FOR BENCHMARK 5

behaviour	delay1	V_{dd1}	delay2	V_{dd2}
1	169.0	5.0	169.0	5.0
2	255.0	5.0	255.0	5.0
3	76.0	5.0	91.3	4.3
4	93.0	5.0	96.0	4.9
5	273.0	5.0	273.0	5.0
6	59.0	5.0	85.7	3.7
7	68.0	5.0	129.0	3.0
8	246.0	5.0	246.0	5.0
9	175.0	5.0	175.0	5.0
10	85.0	5.0	85.0	5.0
11	161.0	5.0	161.0	5.0
12	59.0	5.0	59.0	5.0
13	83.0	5.0	112.0	3.9
14	81.0	5.0	112.0	3.8
15	123.0	5.0	123.0	5.0
16	72.0	5.0	273.0	1.8
17	122.0	5.0	122.0	5.0
18	68.0	5.0	150.0	2.6
19	127.0	5.0	127.0	5.0

TABLE VI
POWER SAVINGS

system	P_{before}	P_{after}	saving (%)
benchmark1	0.685939	0.124112	81.90
benchmark2	0.587834	0.360099	38.74
benchmark3	0.051644	0.010642	79.39
benchmark4	0.142687	0.046971	67.08
benchmark5	0.179883	0.056497	68.59

Technical report, Department of Electrical and Computer Engineering, University of Queensland, Aug. 1995.

[7] Hui Guo and Sri Parameswaren. System level pipelining. In *1996 APCHDL Conference Proceedings*, pages 28–33, 1996.

[8] Mark C. Johnson and Kaushik Roy. Datapath scheduling with multiple supply voltages and level converters. In *Proceedings of Design Automation Conference*, 1997.

[9] Farid N. Najm. A survey of power estimation techniques in vlsi circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.*, Vol.2, No.4:446–455, DEC. 1994.

[10] Lars S. Nielsen, Cees Niessen, Jens Sparso, and Kees van Berkel. Low-power operation using self-timed circuits and adaptive scaling of supply voltage. *IEEE Transactions On Very Large Scale Integration (VLSI) Systems.*, Vol. 2(No. 4):391–397, Dec. 1994.

[11] Sri Parameswaren and Hui Guo. Partitioning of system level pipelines. In *14th Australia Microelectronics Conference(MICRO'97) Conference proceedings*, 1997.

[12] Louis A. Pipes. *Applied Mathematics for Engineers and Physicists.* McGRAW-HILL BOOK COMPANY, INC., 1958.

[13] S. Raje and M. Sarrafzadeh. Variable voltage scheduling. In *Proceedings of International Symposium on Low Power Design*, 1997.

[14] Vivek Tiwari, Pranav Ashar, and Sharad Malik. Technology mapping for low power. In *Proceedings of Design Automation Conference*, pages 74–79, June, 1993.

[15] Chi-Ying Tsui, Massoud Pedram, and Alvin M. Despain. Technology decomposition and mapping targeting low power dissipation. In *Proceedings of Design Automation Conference*, pages 68–73, June, 1993.