LOW POWER TRI-STATE REGISTER FILES DESIGN FOR MODERN OUT-OF-ORDER PROCESSORS

Na Gong[1], Geng Tang[1], Jinhui Wang[2] and Ramalingam Sridhar[1]
[1]University at Buffalo, State University of New York, Buffalo, NY, USA
[2]VLSI and System Lab, Beijing University of Technology, Beijing, China
{nagong,rsridhar}@buffalo.edu

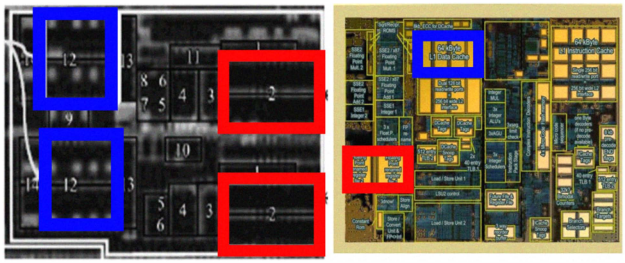
ABSTRACT

In this paper, we propose a novel integrated circuit and architectural level technique to reduce power consumption of register files in high-performance microprocessors. Our simulation results on 32-nm process show 12.7-14.1% power reduction for ROB (Reorder Buffer)-based microprocessors and 12.4-17.9% power reduction for checkpoint-based microprocessors, respectively, with less than 5% impact on excess time.

I. INTRODUCTION

As a fundamental part in modern microprocessors, register files attempt to shrink the big gap between microprocessor and memory systems to achieve high performance [1-3]. Fig. 1, the chip photographs of Intel Pentium 4 processor and AMD Deerhound processor, shows that register files have comparable area to that of level-1 data cache [1]. However, they consume much more power than level-1 data cache due to the highly frequent accesses: every instruction requires three accesses (two reads and one write) to the register files on average, but level-1 data cache is only accessed by each load and store instruction once. This situation is further exacerbated in Simultaneous Multithreading (SMT) processors, where the access frequency is increased by multiple threads. Therefore, register files become one of the most power hungry parts of modern microprocessors and they consume up to 25-37% of the total processor power [4]. Consequently, low power register files design is a critical issue in modern microprocessors.

Researchers have reduced the power consumption of register files in two major ways. The first set of solutions focus on allocation and deallocation mechanisms of registers to enhance the efficiency in register usage and therefore smaller register files can be used to reduce the power consumption [5-6]. The second set of techniques aim at designing novel register files to suppress the power consumption [2-3]. These techniques are close in spirit to our scheme. In this paper, a novel low power tri-state register file design is proposed and its detailed circuit-level implementations and architectural mechanisms are provided. Compared to existing work, our scheme is different in a couple ways: (1) it has greater potential for power reduction (owning to short-pulsed dead signal, register early release technique, and effective low-leakage drowsy and minimum-leakage dead states); (2) it can be applied for both ROB-based and checkpoint-based microprocessors.



(a) Intel Pentium 4 (b) AMD Deerhound

Figure 1: Register files (RED) and level-1 data cache (BLUE) in microprocessors.

II. MOTIVATION

Modern out-of-order processor adopts register rename technique to increase instruction level parallelism. To recover the processor state on an incorrect branch speculation, two well-known recovery mechanisms are applied in industry: ROB and checkpoints. For example, MIPS R1000 [7] uses the ROB approach and Power4 [8] adopts checkpoint-based structure.

This work was primarily motivated by the fact that, registers in both ROB- and checkpoint- based microarchitectures experience different states. In ROB-based microarchitecture, during decoding stages, instructions are inserted in program order in a recorder buffer (ROB). And then they are committed in program order to maintain a precise processor state. According to the usage of its content, a register have four states: empty, ready, idle

and free [5]. Consider the physical register P1 in the code example in Fig. 2. When instruction A is renamed, P1 is mapped to r1 and accordingly P1 changes from free state to the empty state; as A executes, P1 becomes the ready state to contain valid data; after its last consumer instruction B read its value, P1 is in idle state for recovery until the instruction L redefining its logical register (Redefiner) commits; and then P1 returns to the free state. Different from ROB-based microarchitecture, checkpoint-based microarchitecture creates checkpoints (such as CP1 and CP2 in Fig.2) to restore correct processor state on rollback [3]. Accordingly, a register has three states: active registers are used to store useful information; checkpointed registers keep data only for the purpose of recovery; and free registers will not be accessed until they are mapped to another architecture register.

Therefore, the basic idea of our scheme is to design a novel register cell with three states (work, drowsy and dead) and then relate the state of register cells to the state of this register. Specifically, when a register is in ready or active states, its cells are in work state to keep valid data effectively; as the register enters idle state or checkpointed state, its cells are in low-leakage data-retention state for recovery; once the register is released, its cells become minimum-leakage dead state without keeping valid information. The circuit-level implementation and architectural mechanisms of our scheme will be discussed in Section III and IV.

PC	Original code	Rename code	ROB	Check points
A	ADD -> r1	ADD ->P1	P1	
B	OR <- r1	OR<- P8		
Many instructions do not need r1				
L	SUB ->r1	SUB ->r1	P2	
M	BRANCH		-	
N	ADD-> r3	ADD P9	P9	CP1
O	BRANCH			
--	----------	------		CP2

Figure 2: Example Code Sequence

III. CIRCUIT IMPLEMENTATION OF NOVEL TRI-STATE REGISTERS

Fig. 3 shows the schematic of the novel tri-state register. Here, a discharging NMOS transistor ($N_{discharge}$) is inserted to each traditional register cell; and a data-retention NMOS transistor ($N_{data\text{-}retention}$) is connected to each register. The operation of cells in a register is determined by two signals: dead and drowsy. The generation logic of these two signals will be discussed in Section IV.

A. Work state

When the novel tri-state register is in work state (dead=0 and drowsy=0), $N_{discharge}$ of each cell is maintained cutoff and $N_{data\text{-}retention}$ is turned on, so the working function of each cell is similar to a conventional one.

B. Drowsy state

To achieve a low-leakage data-retention drowsy state (dead=0 and drowsy=1), the gated-ground technique [9] is utilized in our design. While $N_{data\text{-}retention}$ is turned off, due to the charging leakage current, the node storing '0' goes up and gets saturated quickly. This saturated voltage depends on the size and threshold voltage of $N_{data\text{-}retention}$. Based on our sizing criteria and technology detailed in Section V, this saturated voltage is about 0.35V, so it can not turn on the NMOS transistor which connects the node storing '1' to virtual ground. Therefore, the node storing '1' is firmly strapped to V_{dd} and the data is retained successfully. Due to the stack effect, the leakage current of a register is suppressed effectively. Note that, the drowsy state is more susceptible to noise sources and process variations as compared to work state.

As compared to the CP register in [3] based on power-gating technique to reduce the leakage current, the advantage of our low-leakage data-retention implementation is that it incurs less area penalty and also it achieves larger leakage current savings which will be discussed in Section V.

C. Dead state

In addition to offering a low-leakage data-retention drowsy state, the proposed tri-state technique provides a minimum-leakage dead state (dead=1 and drowsy=0). In a register cell, the leakage current generated by the local bit line of all read ports, which depends on the stored data of the cell, dominates the total leakage current of the cell. Due to the stack effect, the leakage current of storing '0' is much smaller than that of storing '1' [2]. Therefore, when a register can be released safely, the useless data in it cells can be discharged to '0's, achieving the minimum leakage current.

There are two major differences between our implementation in dead state and the design in [2]. First, the design in [2] adopts a DC dead signal to turn on discharging transistor in the entire dead state, but our design applies a short-pulse dead signal: when a register is released, the dead signal to its cells becomes high to discharge $N_{discharge}$; when the discharging process finishes, the dead signal changes to low immediately to turn off $N_{discharge}$. Our experiment in Section IV shows that this discharging process is very short (~18ps) as compared to the clock frequency (125ps). Such short-pulse dead signal results in two advantages: (1) it achieves gate leakage current savings, because the reverse gate leakage current in OFF $N_{discharge}$ is much less than the forward gate leakage current in ON $N_{discharge}$; (2) the dead pulse turns off $N_{discharge}$ very early, and therefore there is no need to set this signal when the cells becomes work state, which is beneficial to achieve a fast and robust transition. The hardware implementation overhead of the short-pulse dead signal is very small, as shown in Fig. 4. Another difference lies in the fact that the dead signal in [2] is generated based on the conventional register release mechanism while our design combines the early register release technique to achieve maximum power savings, which will be discussed in Section IV.

Note that, since typically the scheduler selects an instruction more than a cycle before the operands are accessed, we expect that the transition cycles of register cells from dead state to work state have little or no impact on the clock cycle. If, however, this becomes a concern, then we can get a warning several cycles before free registers enter the ready state to maintain the performance.

IV. ARCHITECTURAL MECHANISMS OF NOVEL TRI-STATE REGISTER TECHNIQUE

In this section, the architectural mechanisms of the proposed tri-state register are explored in both ROB-based and checkpoint-based processors.

A. Implement in ROB-based microarchitecture

As discussed in Section II, in ROB-based microarchitecture, the conventional register releasing mechanism is designed to support the worst-case scenarios: a register keeps its data in idle state just for branch mispredication recovery and can not be freed until its Redefiner reclaims.

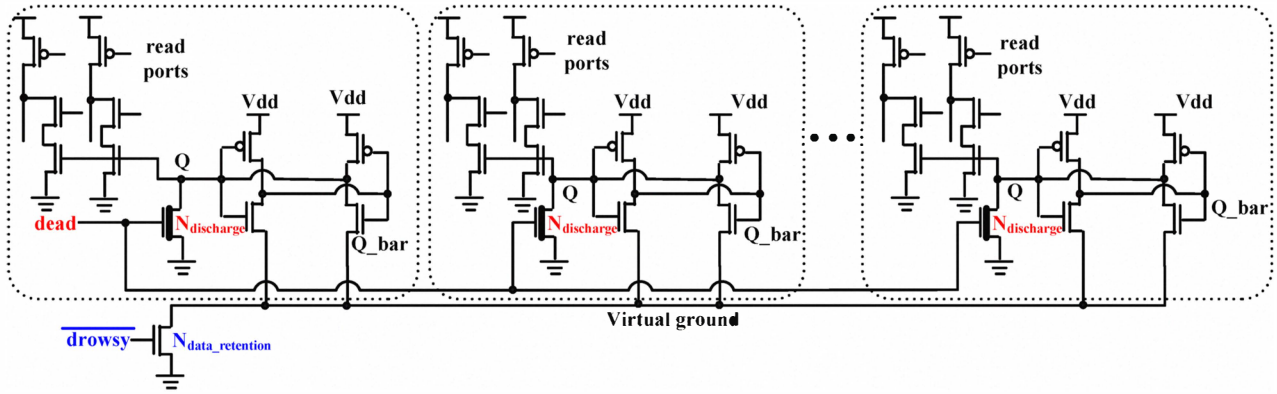
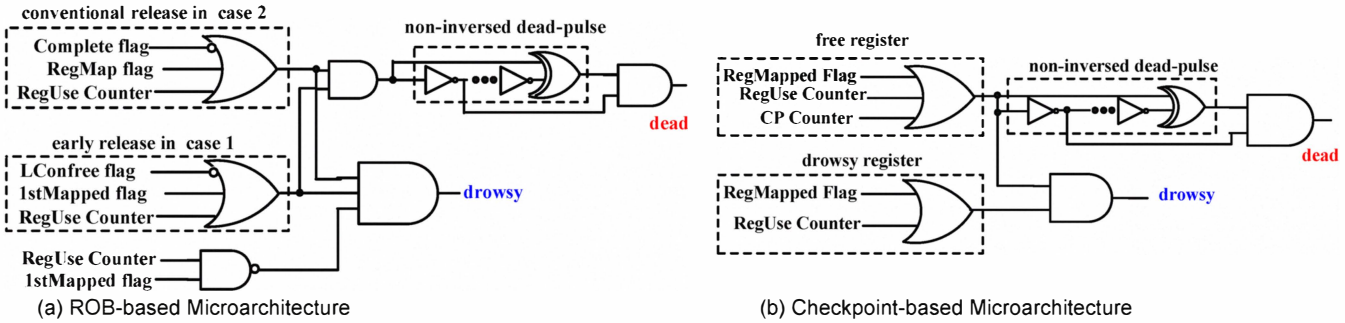


Figure 3: Schematic design of novel register files.



(a) ROB-based Microarchitecture

(b) Checkpoint-based Microarchitecture

Figure 4: Per register management logic in microprocessors

However, due to the instructions between a register's last consumer and its Redefiner (see Fig. 2), a register may stay in the idle state for many cycles and consumes significant leakage power.

Therefore, to achieve high potential power reduction, we combine the register early release technique [5-6] while implementing our proposed scheme on architectural level. Depending on if there are unresolved branch instructions between a register's last consumer and its Redefiner, two distinct cases arise:

Case 1. Based on the compiler assisted register early release scheme, the processor such as an oracle knows the instructions that are the last consumer and Redefiner of each register. If there is no pending branch instruction between the last consumer and Redefiner of a register, once the last consumer reads it, this register can be freed immediately and its cells becomes dead state.

Case 2. The last consumer of a register can be identified, but there are still unsolved branch instructions between its last consumer and its Redefiner. In this case, the register must hold its data in idle state for recovery from brunch misprediction. Since the mispredication happens rarely in practice, its cells can be in the low-leakage data-retention state for recovery and then changes to dead state when this register can be released safely.

Fig. 4 (a) shows the register management logic. RegMap flag, Complete flag, and RegUse counter have been contained in the renaming logic of conventional microprocessors. RegMap flag indicates whether a register is mapped to an architecture register. RegUse counter is used to record the number of consumers that have not read the information of a register. Complete flag denotes if a register has been redefined. In addition, another two bits need to added to achieve register early release mechanism: LConFree flag is set to '1' by the complier in Case 1 and '0' in Case 2; 1stMapped flag is set to '1' when the processor identifies the first instruction which will write valid data into a register, thereby turning the register's cells from dead state to work state. Therefore, as shown in Fig. 4 (a), a dead pulse will be generated when a register is freed either in conventional release condition (RegMap=0, Complete=1 and RegUse=0) or in early release condition (RegUse=0, LConFree=1, and 1stMapped=0). Otherwise, if RegUse=0 and 1stMapped=0, a high drowsy signal is produced to set the cells to low-leakage data-retention drowsy state.

B. Implement in Checkpoint-based processors

The register management logic in checkpointed microarchitecture is shown in Fig. 4 (b), which uses two counters and one status bit per physical register. In addition to RegUser Counter and RegMapped flag, a CP counter is incremented (or decremented) when a reading instruction with the same checkpoint tag renames or executes [3]. Therefore, a register can be freed if and only if RegUse counter = RegMapped = CP counter = 0. In this case, a short-pulse dead signal is generated to discharge all of its cells to dead state. If RegUse counter = RegMapped = 0 but CP counter = 1, then this register is checkpointed and must hold its information for recovery. So its cells can be turned into low leakage data-retention drowsy state by generating a low drowsy signal.

V. EVALUATIONS AND DISCUSSIONS

This section evaluates our proposed tri-state register scheme. We use the following model to estimate the power consumption of registers:

$$\begin{cases} P_{base}=1 \\ P_{non\text{-}base}=\%RF_{'1'}\times NormP_{'1'}+\%RF_{'0'}\times NormP_{'0'} \\ NormP_{'1'}= \\ NormP_{work'1'}\times\%RF_{work}+NormP_{drowsy'1'}\times\%RF_{drowsy} \\ \quad +NormP_{dead'1'}\times\%RF_{dead} \\ NormP_{'0'}= \\ NormP_{work'0'}\times\%RF_{work}+NormP_{drowsy'0'}\times\%RF_{drowsy} \\ \quad +NormP_{dead'0'}\times\%RF_{dead} \end{cases}$$

The variables in the previous equations are defined as follows.

- P_{base} – the power consumption of a conventional register and it equals to 1;
- $P_{non\text{-}base}$ – the power consumption of registers with different techniques, which is normalized to that of a conventional register;
- $NormP_{'1'}$ ($NormP_{'0'}$) – normalized power of registers storing '1'('0');
- $NormP_{work'1'}$ ($NormP_{work'0'}$) – normalized active power of registers in work state while containing '1' ('0');
- $NormP_{drowsy'1'}$ ($NormP_{drowsy'0'}$) – normalized leakage power in drowsy state while containing '1' ('0');
- $NormP_{dead'1'}$ ($NormP_{dead'0'}$)– normalized leakage power in dead state while containing '1' ('0');

$\%RF_{work}$ – the fraction of registers in work state;
$\%RF_{drowsy}$ – the fraction of registers in drowsy state;
$\%RF_{dead}$ – the fraction of registers in dead state;
$\%RF_{'1'}$ ($\%RF_{'0'}$) – the fraction of registers storing '1' ('0').

In the following subsections, first, HSPICE simulation on circuit level is performed to obtain the normalized power consumption of different registers and then combine the architectural level parameters to obtain their overall power consumption.

A. Power consumption on circuit-level

32 nm PTM High k/Metal-gate technology [10] ($V_{tnlow} = |V_{tplow}| = 0.49V$; $V_{tnhigh} = 0.65V$; $V_{DD} = 1V$) is used in this paper for the characterization of register files. Four 32-bit registers with two read ports are designed: standard register, dead register in [2], CP register in [3], and our proposed register.

The sizing of transistors in different registers is based on three important criteria: transistors in the basic register are sized to achieve 8 GHz operation in the application of 128-entry ×32b register files; in the dead register [2] and our design, $N_{discharge}$ is sized minimum (W=L=32 nm) to lower the area overhead; for the transistors used for power-gating technique in CP register and $N_{data\text{-}retention}$ in our proposed register, they are sized to achieve similar ($\leq$ 5%) access time to that of basic register. Based on the MOSIS deep sub-micrometer design rules, we implemented the layout design of different registers and our proposed register cell design uses 16.7% more area than the standard register cell design. Given the power savings discussed as followed, this area increase is tolerable.

Table 1 lists the power consumption of four 32-bit registers. As shown, our proposed tri-state register technique shows good power efficient characteristics as compared to other techniques: in dead state, it achieves as much leakage power reduction as the dead register in [2]; in drowsy state, it can suppress the leakage power effectively; in active state, it shows negligible active power overhead. As also observed, the leakage power of different registers depends on the clock signal of local bitlines (LBL) strongly: registers consume much more leakage power when clock=0. This is because, the low clock signal precharges LBL and produces significant leakage current, which flows from LBL to register cells.

B. Evaluation of overall power consumption

Table 2 reports our evaluation results of the overall power consumption of different registers. Note that, the architectural parameters are derived from the Simplescalar simulations based on SPEC CPU2000 integer and float-pointing benchmarks in [2] and [3]. An eighty-entry ROB and an eight-deep in-order checkpoint buffer are implemented in two microarchitectures. As shown in Table 2, significant power consumption reduction is achieved by our novel tri-state technique. In ROB-based microprocessors, this reduction ranges from as much as 10.1%–11.7% in integer benchmarks and 10%-11.3% in float-pointing benchmarks; in checkpoint-based microprocessors, this reduction is 11% in integer and 11.8% in float-pointing benchmarks.

Also, in our scheme, since all free registers contain '0's by default, writing a '0' to a register cell can be eliminated, thereby saving power consumption of cells, decoder, wordline drivers. This can be implemented by extracting an internal signal from the existing early zero detection logic within the functional units without timing overhead. Simulation result in [2] shows that eliminating writing '0' operation can achieve 6% savings in active power consumption. Accordingly, we re-evaluate our proposed technique in Table 2. As shown, it achieves 12.7-14.1% savings for ROB-based and 12.4-17.9% savings for checkpoint-based processors.

Finally, it should be note that, even if our evaluation does not include the power of management logic, but the hardware implementation overhead of management logic is very limited as compared to two existing microarchitectures. Also, it is based on 32-bit registers with two read ports and. Since our proposed technique reduces the leakage current in LBL, so it can achieve much more power consumption savings for register files with more reading ports.

VI. CONCLUSION

We propose a low power register files technique and provides its circuit implementation and architectural mechanism for both ROB-based and checkpoint-based microprocessors. Experiments have shown significant savings (12.7-14.1% for ROB-based microprocessors and 12.4-17.9% for checkpoint-based microprocessors) and great potential in register files with multiple read ports.

Table 1: Comparison of Power Consumption in different registers (nW)

Power (nW)	States	32 bit Registers	Stored '1'				Stored '0'			
			Clock		Ave.	$NormP_{'1'}$	Clock		Ave.	$NormP_{'0'}$
			0	1			0	1		
Leakage power	all states	basic	1419	433	926	1	377	368	373	1
	Drowsy State	Dead register [2][1]	1413	445	929	1.003	374	365	370	0.992
		register in [3]	1208	219	714	0.770	479	228	354	0.949
		our design	**1101**	**117**	**609**	**0.650**	**513**	**191**	**352**	**0.945**
	Dead State	dead register [2]	393	397	395	0.426	374	365	370	0.992
		register in [3]	1122	137	630	0.679	640	192	416	1.120
		our design	**401**	**400**	**401**	**0.432**	**365**	**356**	**361**	**0.968**
Average Active power	Work State	basic	1950			1	476			1
		dead in [2]	2114			1.084	492			1.034
		tri-state in [3]	1950			1.000	476			1.000
		our design	**2121**			**1.088**	**495**			**1.040**

[1]There is no drowsy state in [2] and this power is just in the work state

Table 2: Evaluation result of power consumption in different registers

	processors	ROB-based microprocessor				Checkpoint-based microprocessor	
Registers	Parameters	Integer		Floating Point		Integer	Floating Point
All registers	$\%RF_{'1'}$ ($\%RF_{'0'}$)	24% (76%)					
Dead-register [2]	$\%RF_{work}$	66%		69%		-	-
	$\%RF_{dead}$	34%		31%		-	-
checkpoint register [3]	$\%RF_{work}$	-		-		56%	13%
	$\%RF_{drowsy}$	-		-		9%	41%
	$\%RF_{dead}$	-		-		35%	46%
Our design	Two Cases	Case1	Case2	Case1	Case2		
	$\%RF_{work}$	21%	21%	23%	23%	56%	13%
	$\%RF_{drowsy}$	0	45%	0	46%	9%	41%
	$\%RF_{dead}$	79%	34%	77%	31%	35%	46%
Normalized overall power with writing '0's	basic	1		1		1	1
	dead register [2]	0.951		0.950		-	-
	register in [3]	-		-		0.886	0.968
	our Design	**0.883**	**0.899**	**0.887**	**0.900**	**0.890**	**0.882**
Normalized overall power without writing '0's	basic	1		1		1	1
	dead register [2]	0.933		0.931		-	-
	register in [3]	-		-		0.886	0.968
	our Design	**0.857**	**0.873**	**0.859**	**0.876**	**0.821**	**0.866**

REFERENCES

1. http://www.chip-architect. com/
2. Lingling Jin, Wei Wu, Jun Yang, Chuanjun Zhang, and Youtao Zhang, "Reduce Register Files Leakage Through Discharging Cells", IEEE International Conference on Computer Design (ICCD), pp. 40-46, Oct. 2006.
3. Amit Golander, Shlomo Weiss Tel-Aviv,"Checkpoint allocation and release", ACM Transactions on Architecture and Code Optimization (TACO), Volume 6 Issue 3, September 2009.
4. R. Shioya, K. Horio, M. Goshima, and S. Sakai, "Register Cache System Not for Latency Reduction Purpose", in Proc. MICRO, pp.301-312, 2010.
5. Monreal, T., Vinals, V., Gonzalez, A., Valero, M. "Hardware Schemes for Early Register Release", in Proc. of ICPP-02, 2002.
6. Amit Golander and Shlomo Weiss, "Exploring the Limits of Early Register Release: Exploiting Compiler Analysis", ACM Transactions on Architecture and Code Optimization, pp:10-26, 2009.
7. K. C. Yeager, "The MIPS R10000 Superscalar Microprocessor," IEEE Micro, vol. 16, no. 2, pp. 28–40, 1996
8. T. N. Buti, R. G. McDonald, et al, "Organization and implementation of the register-renaming mapper for out-of-order IBM POWER4 processors", IBM Journal of Research and Development, vol. 49, no. 1, pp. 167–188, 2005.
9. Amit Agarwal, Hai Li and Kaushik Roy, "A Single-Vt Low-Leakage Gated-Ground Cache for Deep Submicron", IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 38, NO. 2, pp. 319-328, FEBRUARY 2003
10. Predictive Technology Model (PTM). http://www.eas.asu.edu/~ptm