Fine-grained hardware switching scheme for power reduction in multiplication

Y. Huang✉, C. Li, M. Li, L. Van der Perre and W. Dehaene

This Letter presents a fine-grained hardware switching scheme to choose from the proper hardware for low power computing. It exploits the word-length optimisation opportunities for multiplication unit. With the proposed technique, the gate-level simulation result on OpenRISC shows 23.7% power reduction for the multiplication unit, which accounts for 9.5% power reduction for its execution unit.

Introduction: Energy consumption is one of the most critical metrics for embedded signal processing systems. Traditionally, designers optimise the fixed-point word-length that provides just-necessary precision to minimise the power consumption. On the other hand, driven by the increasing demand for computing reprogrammability, general purpose computing devices, e.g. DSPs, ASIPs, GPUs, and application processors, are becoming more favourable. In these systems, designers are constrained to perform the arbitrary word-length optimisation, since processors are typically sacrificed in hardware cost to cater to the most complex computing cases.

In general purpose processors, single instruction, multiple data (SIMD) exploits the over-reserved word-length by applying parallelism in data-path processing. This reduces the number of operations, and hence decreases energy consumption. However, it requires dedicated hardware as well as software tuning, to enable those SIMD intrinsic functions. Soft-SIMD [1], on the other hand, relies purely on software to exploit the sub-word parallelism. Nevertheless, in this scheme, guard bits is required to be inserted, which is non-trivial for software developers, especially for multiplication operations [2].

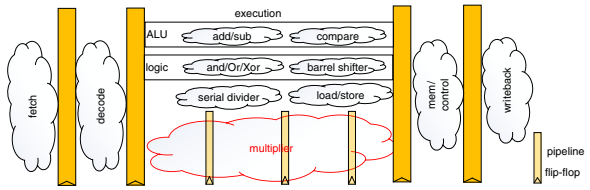


Fig. 1 *Customised Cappuccino mor1kx microarchitecture*

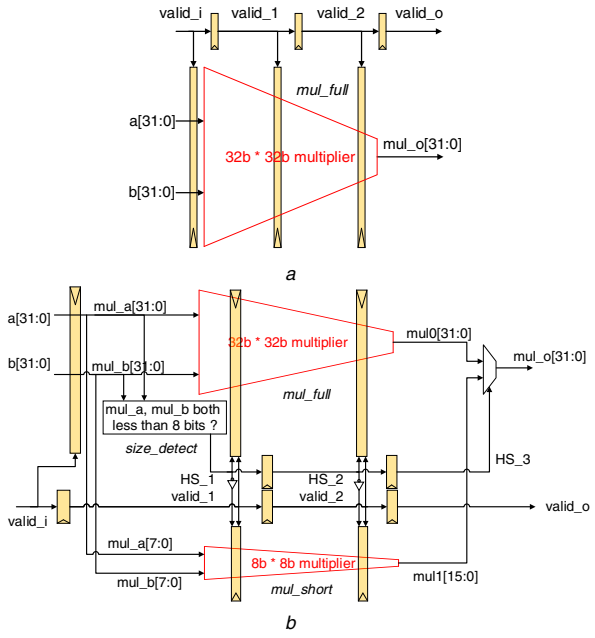


Fig. 2 *Schematics of multiplier implementations*

a Original three-stage multiplier (original)
b HS between 32 bits and 8 bits multiplier

In this Letter, we introduce an alternative low precision computation unit besides the traditional full precision unit. A hardware word-length detector is used to switch the hardware units, in a fine-grained manner, to reduce the computational cost when full-precision computation is not necessary. The low word-length computations can be accomplished by the reduced precision unit without performance degradation. Therefore, the activation chance of the full precision unit is radically reduced and hence the dynamic power consumption decreases significantly accordingly. As the detection and execution units are both constructed in hardware, this technique requires no modification on compiler nor on software.

In the remaining of this Letter, we demonstrate this technique with an alternative low-precision multiplier. As its power consumption is $O(n^2)$ regarding the word-length n, using low-precision multiplication results in significant power saving.

Power in multiplication: Without losing generality, we adopt a simple 32-bit OpenRISC processor, called mor1kx (Cappuccino implementation) [3]. The schematic of the processor is shown in Fig. 1. The clock frequency is set to 1 GHz. The processor is implemented in TSMC 28 nm hpm technology. The execution stage is comprised of a ALU, a Logic computation unit, a Load/Store unit, a serial divider and a four-cycle multiplier. We adopt clock-gating in the multiplier unit to avoid the signal toggling in the multiplier when performing irrelevant instructions (see Fig. 2*a*).

The area utilisation and power consumption of the multiplier unit for each instruction are profiled in Table 1. For the 32-bit multiplier, even if the word-length of multiplicands is much shorter than 32-bit, the power consumption is comparable with full-precision multiplication. This is due to the fact that multiplicands are represented in 2's complement form. In this form, the most significant bits (MSBs) are filled with '1's or '0's when the number is short, which results in high toggling rate during positive-to-negative or negative-to-positive transition. Nevertheless, if proper multipliers are used, e.g. 8-bit multipliers for 8-bit multiplicand, the power consumption can obviously be much reduced.

Table 1: Multiplier area and power consumption on each instructions w.r.t. multiplicands size

Multiplier size	Cell area [μmm^2]	Power during instructions [μW]				
		NOP	4-bit MUL	8-bit MUL	16-bit MUL	32-bit MUL
4-bit	109	12.152	48.022	N/A	N/A	N/A
8-bit	289	16.497	100.417	111.18	N/A	N/A
16-bit	1030	49.511	234.2	278.934	349.657	N/A
32-bit	1744	50.737	391.350	451.808	531.32	567.924

This provides great opportunity for power optimisation in processors, as the multiplicands are not always defined at the full size of 32-bit long integer. Moreover, even if the multiplicands are defined as 32-bit long integer, the actual value can be very small, e.g. between −128 and 127 (which can be represented by a 8-bit number).

Considering the above fact, we propose to introduce an alternative lower-cost multiplier to perform the computation when the word-length of multiplicands is short enough (see Fig. 2*b*).

A simple size detecting unit (*size_detect*) is deployed to detect if both multiplicands are small, by checking if the MSBs (in this example from 8-bit on) is the same (all '1's or '0's). If both multiplicands are short, *mul_short* will execute the operation while the *mul_full* is clock-gated, and vice versa. This HS scheme ensures that, the signal toggling only happens in the proper multiplier unit, and the signal toggling in the other multiplier unit is minimised. The multipliers are divided into three stages by two sets of pipeline registers. Signals in the first stage always toggle even if the unit is not enabled, since the logic input of the first stage are not clock-gated by the *size_detect*. To minimise the power cost of the first stage, we retime the multiplier using the synthesise tool (Cadence RTL compiler), to locate the minimum necessary portion of the multiplier into the first pipeline stage.

The cell area of the HS multiplier is 2053 μmm^2, which is 18% higher than the original multiplier. This is due to the introduction of the short multiplier and the corresponding MUX circuit. The power consumption of the original and HS multiplier is compared in Fig. 3. It is broken down into three parts: *mul_full*, *mul_short*, and *mul_rest* (rest of the parts in the multilier). During no operations (NOPs), both multipliers consume less than 40 μW, which mainly attributes to clock gating cells. For the HS multiplier, if the multiplicand is shorter than 8 bits,

the *mul_full* unit is clock-gated and the processing is switched to the low-power *mul_short*. Therefore, the overall power consumption is significantly lower than the original multiplier. This advantage diminishes when all the multiplicands are greater than 8 bits. In that situation, the HS multiplier suffers from the power penalty of the *size_detect* and the MUX unit.

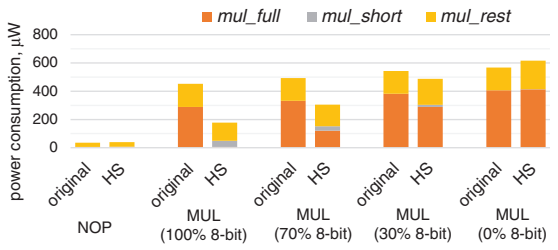


Fig. 3 *Power consumption of Ori and HS multiplier during NOP and MUL. The multiplicands are randomly generated to be whether 8-bit or 32-bit, with accordingly possibility*

Algorithm profiling: To measure the power consumption benefits of the HS multiplier, it is important to track the utilisation frequency of the multiplication operation (#multiplication/#instructions), and the statistical chances that both multiplicands are short. These statistics depend heavily on applied algorithms and the input data. To get a fair result, coremark 1.0 [4] and 10 other common algorithms are benchmarked. *Cormark* focuses on benchmarking CPU cores of embedded systems. The selected algorithms cover a wide range of typical embedded processing applications, e.g. fft, filtering, jpeg decoding, cryptography, and error correction. The input data is set to best represent the typical usage scenario.

Fig. 4 shows the utilisation frequency of the multiplier. On average, 1% of the instructions is a multiplication. Since each multiplication takes four cycles, the processor will take around 4% of the cycles for multiplications.

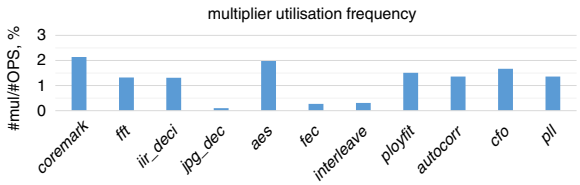


Fig. 4 *Utilisation frequency of multiplier*

The word-length distribution of the multiplicands is illustrated in Fig. 5. The data is obtained by the cycle-accurate OpenRISC simulator. The multiplicands are recorded for each multiplication. The figure shows that if the criterion for short-input is stronger, i.e. # of bits is larger, the activation chance of the *mul_short* unit will increase, which hence leads to lower power. However, the *mul_short* unit itself consumes more power with increased short-input word-length due to the higher area cost. Therefore, designers are suggested to profile the multiplication size coverage for the typical applications and the power consumption w.r.t. multiplication size.

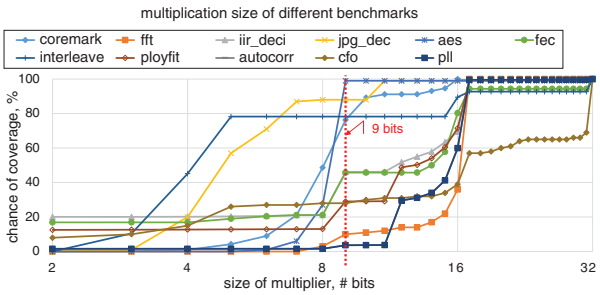
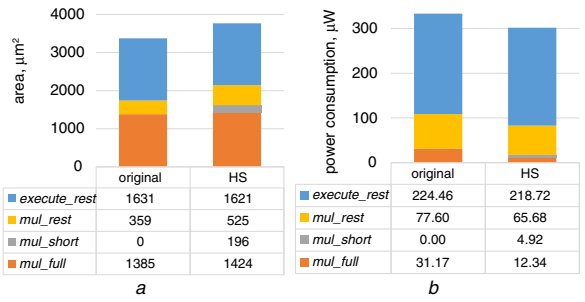


Fig. 5 *Chance of both multiplicands are short for each benchmarks*

On the basis of the benchmarks in Fig. 5, we use a 9-bit multiplier for the *mul_short*. With this scheme, the *mul_short* performs more than 80% of the multiplication for *coremark*, *jpg_dec*, *aes*, and *interleave*; around 40% multiplication for *iir_deci*, *fec*, *polyfit*, and *cfo*; around 5% for *fft* and *pll*. This result shows the HS scheme best fits algorithms that heavily use *short_integer* data-types for multiplications. In this scenario, the *size_detect* takes the role of the compiler to choose the suitable multiplication hardware. Moreover, for algorithms that use only full-width *integer* data-type, e.g. *iir_deci* and *polyfit*, the HS scheme still performs around 40% of the multiplications. This is due to the fact that the varying input data has a very high tendency of falling into the short-size range, even though they are defined to be very wide to avoid the overflow in the worst case.

Verification: The mor1kx is synthesised at 1 GHz in TSMC 28 nm, and is simulated at gate-level with extracted parasitic parameters. The area and power metrics with original or HS (with 9-bit *mul_short*) schemes are shown in Fig. 6. For the processor with HS scheme, The extra *mul_short* and *size_detect* results in 23.0% area overhead for the multiplier unit, which is equivalent to 11.5% area overhead for the whole execution unit. The power consumption of the *mul_full* is reduced from 31.167 to 12.344 µW, since its execution ratio is much reduced. This leads to a total of 23.7% power saving for the multiplier unit and 9.5% power saving for the execution unit.



	original	HS
execute_rest	1631	1621
mul_rest	359	525
mul_short	0	196
mul_full	1385	1424

a

	original	HS
execute_rest	224.46	218.72
mul_rest	77.60	65.68
mul_short	0.00	4.92
mul_full	31.17	12.34

b

Fig. 6 *Metrics for the execution unit with original and HS scheme*
a Cell area breakdown
b Power breakdown when running cormark

Conclusion: The HS scheme proposed in this Letter exploits the word-length opportunities to reduce the dynamic power consumption. An alternative short multiplier is utilised when the circuit detects the input is short enough. As a result, power consumption is reduced.

The proposed scheme does not affect the software nor the compiler, since the detection and switching are implemented at hardware level. It best fits processors which frequently perform short multiplications. In such processors, the multiplier unit power can be much reduced.

Submitted: *30 October 2015* E-first: *14 July 2016*
doi: 10.1049/el.2015.3828
One or more of the Figures in this Letter are available in colour online.

Y. Huang, C. Li and M. Li (*IMEC, Leuven, Belgium*)

✉ E-mail: yanxiang.huang@imec.be

L. Van der Perre and W. Dehaene (*ESAT Department, KU Leuven, Belgium*)

Y. Huang and C. Li: Also with ESAT Department, KU Leuven, Belgium

References

1 Kraemer, S., Leupers, R., Ascheid, G., and Meyr, H.: 'SoftSIMD – exploiting subword parallelism using source code transformations'. Design, Automation Test in Europe Conf. Exhibition (DATE'07), 2007, pp. 1–6
2 Novo, D., Kritikakou, A., Raghavan, P., der Perre, L., Huisken, J., and Catthoor, F.: 'Ultra low energy domain specific instruction-set processor for on-line surveillance'. IEEE 8th Symp. on Application Specific Processors (SASP), 2010, pp. 30–35
3 'mor1kx – an OpenRISC Processor IP Core'. Available at https://github.com/openrisc/mor1kx
4 'CoreMark, an EEMBC Benchmark. CoreMark scores for embedded and desktop CPUs'. Available at http://www.eembc.org/coremark/index.php