IJCSC Volume 5 • Number 1 March-Sep 2014 pp. 101-102 ISSN-0973-7391

Rapid and Compact generic binary-to-BCD conversion circuit for less powerful computational devices

(An Algorithm and Hardware Simulation)

Ujjwal Thaakar, Darshana Upadhyay
CSE Department, NIT, Nirma University, Ahmedabad, India
{10bit050, darshana.upadhyay}@nirmauni.ac.in

Abstract— Conversion of binary representation into BCD is a very frequently used operation in modern computers due to the exponential increase in the demand for decimal arithmetic. Decimal data processing applications have developed exponentially in recent years thus increasing the need to have hardware support for decimal arithmetic. Most conversion algorithms are fixed bit. we propose an algorithm that is generic and independent of the size of the binary word to be converted.A general purpose algorithm can greatly simplify and unify the process of building BCD conversion hardware implementations at a certain performance cost. This paper illustrates a generic algorithm for converting any n-bit number to its packed BCD equivalent. It is based on an insight into the relationship between a number and its BCD representation. This paper presents novel high speed low power architecture for n- bit binary to BCD conversion.

Keywords—Algorithm; BCD, Binary; Conversion; Generic

I. INTRODUCTION

Decimal Arithmetic receives significant attentionin commercial business and internet based applications. Providing hardware support in this direction in future is essential.

Decimal computer arithmetic is preferred in decimal data processing environments such as scientific, commercial, financial and Internetbased applications[1]. Ever growing needs for processing power,required by applications with decimal arithmetic, cannot be met by conventional slow software simulated decimal arithmetic units[1]. Recognizing the need for decimal arithmetic has led to specifications for decimal floating point arithmetic to be added in the draft revision of the IEEE-P754 standard [1*]. Decimal arithmetic operations are usually slow and complex, its hardware occupies more area. They are typically implemented using iterative approaches or lookup table based reduction schemes. This has led to the motivation behind improving BCD architectures, to enable faster and compact arithmetic. However, their hardware counterparts, as an integral part of recently commercialized general purpose processors[2] are having more importance. Binarycoded decimal encoding of decimal digits has conventionally dominated decimal arithmetic algorithms.There are a lot of algorithms for conversion of binary numbers into BCD but most are word size specific[3]. Here we propose an algorithm that is generic and independent of the size of the binary word to be converted.

The rest of this paper is organized as follows: Section 2 gives an overview on general BCD conversion and its need. Section 3 discusses the methodology of proposed work. Section 4 explains algorithm of proposed work, Section 5 discusses the hardware simulation of proposed architecture in detail and finally Section 6 concludes the paper.

II. BACKGROUND

Demand for decimal data processing applications has been growing exponentially in recent years which increases the need to have hardware support. Binary to BCD conversion forms the basic building block of these types of applications [4] .There are a couple of standard methods for BCD conversion. These methods generally make use ofexpensive operations like multiplication or division and modulo, all of which are rlatively complex and thus time consuming operations[6]. Here we propose an algorithm which makes use of much faster operations such as addition, shifting and comparison.

Some of the existing methods make use of the fact that

$$\beta_v\beta_{v-1} \quad \beta_2\beta_1\beta_0 = 2^v\beta_v + 2^{v-1}\beta_{v-1} \quad 22\beta2 + 2\beta1 + \beta0$$

oρ

$$\beta_v\beta_{v-1} \quad \beta_2\beta_1\beta_0 = \beta_0 + 2(\beta_1 + 2(\beta_2 + \quad +2(\beta_{v-1} + 2\beta_v))) \qquad [6]$$

Another method is what comes to our minds intuitively. It involves constantly dividing the binary number by10 and using the remainder to obtain the individual digits of the final BCD number.

III. METHODOLOGY OF PROPOSED WORK

Here we propose an algorithm that is generic and independent of the size of the binary word tobe converted.

1. Let there be any *n+1* digit number represented by thepolynomial[3]:
    1. $a_0*10^n + a_1*10^{n-1} + a_2*10^{n-2} + \ldots. + a_n*100$ where $0 \le a_i \le 9$ □ $0 \le i \le n$
2. A BCD equivalent of this number is:
    1. $a_0*16^n + a_1*16^{n-1} + a_2*16^{n-2} + \ldots. + a_n*160$

3. Therefore it is clear that we need to add:
    1. $a_0*(16^n - 10^n) + a_1*(16^{n-1} - 10^{n-1}) + a_2*(16^{n-2} - 10^{n-2}) + \ldots + a_n*(160 — 100)$
4. Let's look at the first term $a_0*(16^n — 10^n)$ which can be split into:
    1. $a_0*(16^{n-1}*(10 + 6) - 10^n)$
    2. $a^0*(16^{n-1} * 6 + 16^{n-1}*10 — 10^n)$
    3. $a^0*16^{n-1}*6 + a^0(16^{n-1}*10 — 10^n)$
5. $a_0(16^{n-1}*10 — 10^n)$can again be split in the same manner until we're left with:
    1. $a_0 * (16^{n-1} * 6) + (a_0 * 10) * (16^{n-2}*6) + (a_0 * 10^2) * (16^{n-3} * 6) + \ldots + (a_0 * 10^{n-1}) * 6$
6. Similarly any term a_j □ $0 \leq j \leq n$ of the polynomial can be expanded as following:
7. Adding all these expansions results in:
    1.

Of course the problem with binary numbers is that the terms a_i are not known beforehand. This can be solved since we know that adding 6 a certain number of times produces one of the digits of the number. This can be used effectively to add 6 and compare a certain portion of the word with the number of times we've added. This concept is explained in the algorithm given below.

IV. Algorithm of Proposed Work

Here is a C++ implementation of the proposed algorithm which has been implemented recursively. The

stack level can be predetermined by the size of the input in terms of the number of nibbles for a hardware implementation.

```
//n is the no. & b the no.of bits
int bcd(int n, int b) {
if (b == 0)
return n<=9?n:bcd(n+6, 4);
int count = 0;
while (count < (n >> 4)) {
n += 6;
count++;
}
if ((n&0xf) > 9) {
n += 6;
count++;
}
 return bcd(count, b-4)<<4|bcd(n&0xf, 0);
}
```

V. Simulation of Proposed Algorithm

Procedure

Let us illustrate the procedure using a numerical example. We will convert a 8-bit number 129 into its equivalent 12-bit BCD form.

1. Keep adding 6 to the number (129) until the number of times you have added 6 is equal to the number shifted 4 bits to the right.
2. We find this happens 12 times after which the number has become 201. 201 shifted 4 places to the right gives us 12. Thus we stop. Now the number in the least 4 bits is 9. This gives us our first digit 9.
3. Now we repeat the same procedure on 12. We find that adding 6 once gives us 18 which shifted 4 places to the right (which is equivalent to dividing by 16) gives us 1. At this point we have our next digit in the least significant 4 bits i.e. 2.
4. Now that we only have a single digit left i.e. 1 - we're done. Our final result is 1-2-9 or 297 which is the BCD of Hardware Simulation 129.

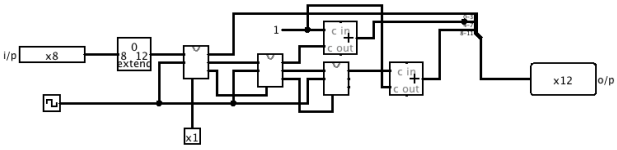

Fig 1. Main

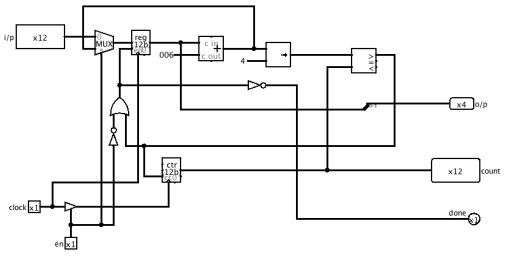

Fig 2. While loop

The blocks used in the main circuit are the while loops illustrated below.

VI. Conclusion

This paper is attempted to provide generic and independent solution which convert binary word to BCD. To get generic & efficient conversion,design to be proposed is repeatedly adding 6 until the number shifted 4 bits to the right is equal to the number of additions produces the BCD of an n bit word where n = 4*k, k □ □. This algorithm completely eliminates the need formultiplication and division operations, thus providing an alternative for less powerful computational devices which can use a hardwired version to perform blazing fast conversions. This paper presents novel high speed low power architecture for n- bit binary to BCD conversion.

REFERENCES

1. Erle, M.A.; Schulte, M.J., "Decimal multiplication via carry- save addition," Proceedings. IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2003, 24-26 June, 2003 Page(s):348 - 358.
2. Binary-coded decimal digit multipliers Jaberipur, G.; Kaivani, A.Computers and Digital Techniques, IET Volume 1, Issue 4, July 2007 Page(s):377 - 381.
3. A; Antelo, E M"i Vazquez A; Antelo, E; Montuschi P, "A New Family of High-Performance Parallel Decimal Multipliers" in the 18thIEEE Symposium on Computer Arithmetic 25-27 June 2007.
4. Sreehari Veeramachaneni, M. Keerthi Krishna , L. Avinesh, P Sreekanth Reddy, M.B. Srinivas, “Novel High-Speed 16-Digit BCD Adders Conforming to IEEE 754r Format”, IEEE Computer Society Annual Symposium on VLSI (ISVLSI 07), pages 343-350, Mar 2007.
5. James, R.K.; Shahana, T.K.; Jacob, K.P.; Sasi, S.”Decimal multiplicationusing compact BCD multiplier”,Electronic Design, 2008. ICED 2008. International Conference on 1-3 Dec. 2008 Page(s):1 – 6.
6. fsRhyne, V.T.: „Serial binary-to-decimal and decimal-to-binary conversion , IEEE Trans. Comput., 1970, 19, (9), pages. 808–812.
7. L. Eisen et al., “IBM POWER6 accelerators: VMX and DFU,” IBM Journal Research and Development, vol. 51, no. 6, pp. 663–684, Nov.2007.
8. E. Schwarz, J. Kapernick, and M. Cowlishaw, “Decimal floating-pointsupport on the IBM System z10 processor,” IBM Journal Research and Development, vol. 51, no. 1,pp. 4:1–4:10, Jan./Feb. 2009.
9. Erle, M.A.; Schulte, M.J.,...M. P. Vestias and H. C. Neto, “Parallel decimal multipliers using binary multipliers,” in VI Southern Programmable Logic Conference, (SPL 2010),Ipojuca, Brazil, Mar. 2010, pp. 73–78.