DSP BASED DIGIT SERIAL ARCHITECTURE

Ms. P. J. Tayade*

Dr. Prof. A. A. Gurjar**

Abstract:

This paper presents a systematic ***unfolding transformation technique*** to transform bit-serial architectures into equivalent digit-serial ones. The novel feature of the unfolding technique lies in the generation of functionally correct control circuits in the digit-serial architectures. Bit-serial systems process one bit of a word **or** sample in a clock cycle. For some applications bit-serial architectures may be too slow, and bit-parallel architectures may be faster than necessary and may *require* **too** much hardware. The desired sample rate in these applications can be achieved using the digit-serial approach, where multiple bits of a sample are processed in a single clock cycle. The number of bits processed in one clock cycle in the digit-serial systems is referred to **as** the ***digit-sire;*** the digit size can be any arbitrary integer (the digit size was restricted to be a divisor of word-length in the past ***adhoc*** designs). We present digit-serial implementation of two's complement adders and multipliers. Unfolding of multiple-rate operations (such **as** interpolators and decimators) is also presented.

Keywords:-

1] Altera / Xilink 2] Modelsim / Active HDL Tools / software CMOS: - 1] Cad tool 2] tanner tool 3] Cadence tool 4] microwind.

* Electronics & telecommunication, Sipna's C.O.E.T., Amravati, India

** Electronics & telecommunication, Sipna's C.O.E.T., Amravati, India

INTRODUCTION:-

These Dedicated systems requiring different sample rates will need to be implemented in different implementation styles (such as bit-serial, bit-parallel. and digit-serial). Bit-serial implementations process one input bit at a time and are ideal for low-speed applications systems **require** less interconnections, less hardware, and less pin-out. Bit-parallel systems process all input bits of a word or sample in one clock cycle, and **require** the largest amount area, interconnection, and pin-out. These systems are ideal **for** highest-speed applications Digit-serial systems, on the other hand,

process more than one input bit in one cycle**.** These systems are ideal for moderate speed applications, **for** which bit-serial style is too slow, and bit-parallel style is faster than necessary. The number of bits processed per cycle is referred to as the ***digit-size.*** Fm digit-size of unity, the architecture reduces to a bit-serial system, and for digit-size equal to the word-length, the architecture reduces to a bit-parallel system. For small digit-sizes, the sample rate in digit-serial architectures increases linearly with increase in the digit-size .

Although bit-serial architecture design styles have **been** known for a long time, the digit-serial architectures were explored only recently. In this paper, we present a systematic approach to transform an arbitrary bit-serial architecture into a digit-serial architecture. Our approach can systematically generate all appropriate control circuits for functionally correct operation of the digit-serial architectures, for any arbitrary digit-size (and not just for digit-sizes which are divisors of word-length). We present a novel ***unfolding transformation algorithm*** to transform a bit-serial structure into a digit-serial structure by unfolding with an unfolding factor equal to the digit size, we presented the unfolding technique to unravel hidden concurrency in data-flow signal processing programs, and to construct more efficient multiprocessor schedules.

The switches, interpolators and decimators. In this paper, we present general unfolding algorithms, which can unfold switches (for arbitrary unfolding factors), interpolators, and decimators.

Review of work:

. **Yun-Nan Chang** has presented a design methodology for a new class of digit-serial multiplier architectures. These architectures can be pipelined at the bit-level, and as a result power can be reduced. For a specified wsf, the clock speed required with a bit serial design is much higher than digit-serial with digit size 4 or 8. As a result, the power consumed by a bit-serial design due to high-speed clock is much higher and this favors digit serial architectures with respect to low power consumption. It should also be noted that for large digit sizes.

Digit-Serial Architectures using Unfolding Transformation In this section, we motivate the need for digit-serial architectures, and present systematic design of digit-serial architectures using the unfolding Utansformation. Consider the (word-serial) bit-serial implementation of the simple add operation. There are two approaches we can think of. First, we can process two input samples simultaneously, and process each input in a bit-serial manner; this corresponds to a word parallel bit-serial system with block size of two. Alliteratively, we can process the inputs in a word serial manner, but process two bits of a word in parallel; this corresponds to a word-serial digit-serial implementation with digit-size.

Motivation:

In parallel system speed, power is high and in addition area constraint is also high .To reduce the power and area we prefer bit serial architecture but it may be slow in speed, While we gain the power and area constraint but it loses the speed . So there is need of such a architecture which balance the both things from designer side, that is speed , power and area constraints .This is motivation to design a digit serial architecture.

Proposed work:

Digit-Serial Architectures using Unfolding Transformation

The digit-serial architecture requires less latches in the add computation portion, and in the **data** format conversion portions, and requires simpler control circuits **(as** compared **to** the word-parallel bit-serial design). The reduction in the number of latches favors the digit serial

architectures for VLSI implementation. and this motivated the study **of** systematic design of these architectures. Bit-serial circuits, as we will see, require explicit specification of control operations . Most of these operations are described by switches. The switches connect to a certain input in specified time instances. Thus we need to consider unfolding of circuits with and without switches. Now we present a systematic unfolding algorithm, which unfolds an arbitrary bit-serial flow-graph (containing function operators and switches) by an arbitrary unfolding factor.

Systematic Unfolding:

Thus we need to consider unfolding of circuits with and without switches. Now we present a systematic unfolding algorithm, which unfolds an arbitrary bit-serial flow-graph (containing function operators and switches) by an arbitrary unfolding factor. The architecture is assumed to be represented by a data-flow graph (DFG) . The DFG consists of nodes (which describe computations or tasks), and arcs (which represent communication). Some nodes could be dummy and arcs carry non-negative number of registers **or** delays. For example, a node *U+V* with ***i*** registers implies that the result of iteration n of *U* is used for the (n+i)-th execution of node V. If a data-flow graph is unfolded by a factor of *J* , then the number of nodes and arcs in the unfolded data-flow graph are ***J*** times those in the original graph. Each node ***I/*** in the original graph is replaced by *J* nodes (lo. **U1,** ..., and The node *U,* executes the iterations *q,* ***q + J , q+U,*** etc.

Unfolding Algorithm:

Step **1:** For each node ***U*** in the bit-serial system, draw ***J*** nodes in the unfolded system, and label them ***Uo. U1,*** .... U]-1.

Step ***2*** For an arc *U-+V* in the original graph with no delay, draw arcs *U,-tV,* with no delay in the unfolded graph [for *4* = 0 to (J-l)].

Step **3:** Consider the arc *U+V* with i delays. If i d , t hen perform Step **3a.** If **za,**th en perform

Step **3a:** Draw **arcs *U,-iw+Vq*** with one delay [for *q* =O to (i-l)]. Draw arcs *U,,+V,* with no delay [for ***4*** = i to (J-l)].

In this case, the execution of *4-th* iteration of V consumes the output of (q-i)-th iteration of ***U,*** which is executed by rhe node ***U,,*** is *4* >i, or **by *U* , , ,** before one **cycle** if *q* **<*i.***

Step 3b: Draw arcs *U,*+,,-,*-+*V,* with ry 1 delays [for *q* = 0 to (J-l)].

For this cme. the (q-i)-rh iteration of ***U*** is executed by the node ***U r y v l w*** &fore *(7*1 cycles

Step **4:** Let ***U*+*S*** be an input to switch **S** from a node ***U*** (and let the switching instance to this input

be (WI + **U),** or equiv a***I***le**"**n.tly + ***U* .** Let the output of the switch S be connected to ***V* .** For each

such operator, draw ***J*** switches **So** through ***S J -] .*** The output of switch **S,** is connected to node *V,* [for

q = 0 to (J-I)]. Connect switch ***S,*** to the input from node ***U, mod J*** at time instant *(W'I* + 1).

Note that the switching instance is expressed as

Jw'l **+ *U*** *=I(W'[* + ["J ***) + (U mod J) ,***

which implies that in the unfolded data-flow graph, U *mod J* should **be** connected to S *mod J* at time instance [*W'I* + L" J 3.

The notation Lx J represents the floor function of ***x*** .The notation Lx J represents the floor function of ***x* ,** i.e. the largest integer equal to **or** less than **x.** The notation ***[x*** 1 represents the ceiling function of **x,** i.e. the smallest integer greater than or equal to ***x* .** The notation ***a*** *mod b* represents the remainder of ***alb* .**

Bit-Serial and Digit-Serial Multipliers:

In this section, we apply the unfolding transformation on known bit-serial multipliers to obtain architectures for digit-serial multipliers. We present digit-serial multiplier implementations with programmable coefficients, which are time-invariant. Consider the two's complement multiplication of a 8-bit signal ***x*** with 4-bit coefficient . The signal ***x*** and the coefficient a are represented **as**

x = ***x7*** . ***x*** ~ ***s x y (3 x 2 x 1 x***a~ **,**= a3 . *a 2 a~a o* .

The product **s** is given by

s = - x x a , + x x a ~ +' X X U , ~ -+~ x ~ 0 2 - 3

= -xxa3 + [x x a z + [x u l + xxa,,~'] 2-' I 2-l

Digit-Serial Serial-Parallel Multiplier:

Fig3. shows the bit-serial architecture for a 8x4-bit multiplier. This architecture is unfolded by four to obtain a digit-serial implementation with digit-size **4;** in this architecture **4** bits of the word or sample are input in one cycle, and the entire 8-bit sample is input in **two** consecutive clock cycles.

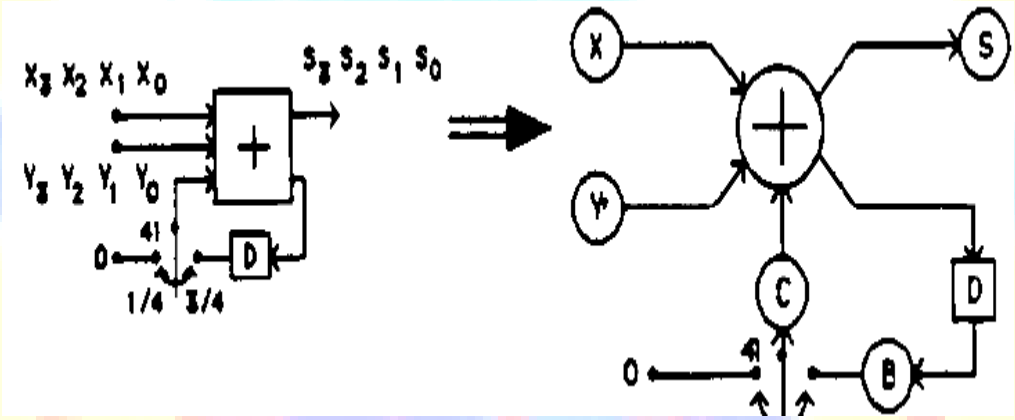

Figure 1- BIT SERIAL ADDER

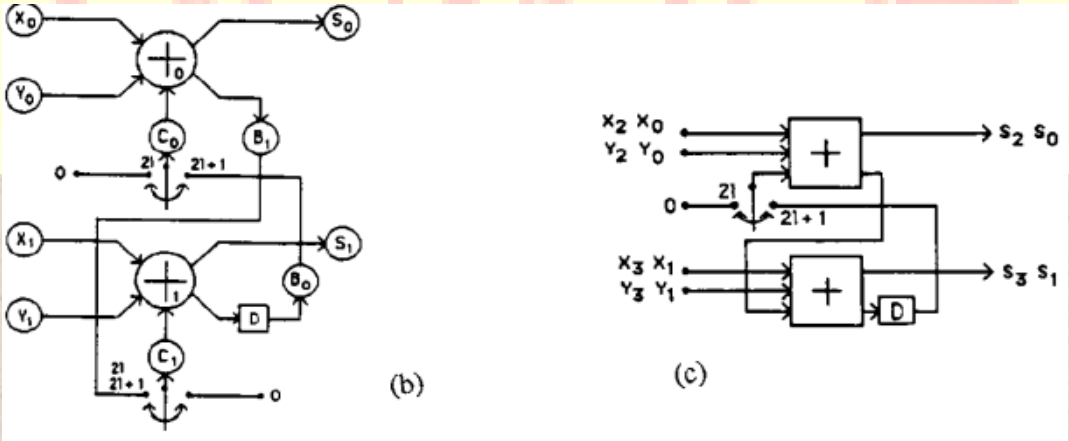

Figure 2-DIGIT SERIAL ADDER

Unfolding *of* Interpolators:

Interpolation operations sample an input signal with period T and output an input signal with period - where ***M*** is the interpolation ratio. This system upsamples the input signal and leads to an increase in the sample rate by a factor of ***M*** **.** Typically this is achieved by inserting **(M-1)** zero **sam**ples to every input sample. In some applications, ***M*** times series may also need to be interpolated to upsample the sample rate by a factor of ***M*** .

Let *U 4* be an **arc** between a node U and an interpolator 1 and let the switching instance be Ml+m (where M is the interpolation ratio). The interpolation operation is described by

T M

i(Ml+m) = *u (l)*

and the unfolded interpolation operation is given by

u(Jl+j) = i(MJl+Mj+m) , ***j*** = ***0*** J-1

after L+ J cycles, since

The input ***u(Jl+j)*** represents the output of ***U*** in the current cycle. The output ***i(MJI*** + ***Mj*** + ***m) is*** the output of node ~[w,+,,,) ***M '+m' mod***

MJl + M j + m = J [M i + 1-1 I + ***[Mj+ml mod J*** .

Thus we connect node ***U,*** to interpolator ***I c m + , ~mo)d*** with a switching time of ***MI+*** l J l.

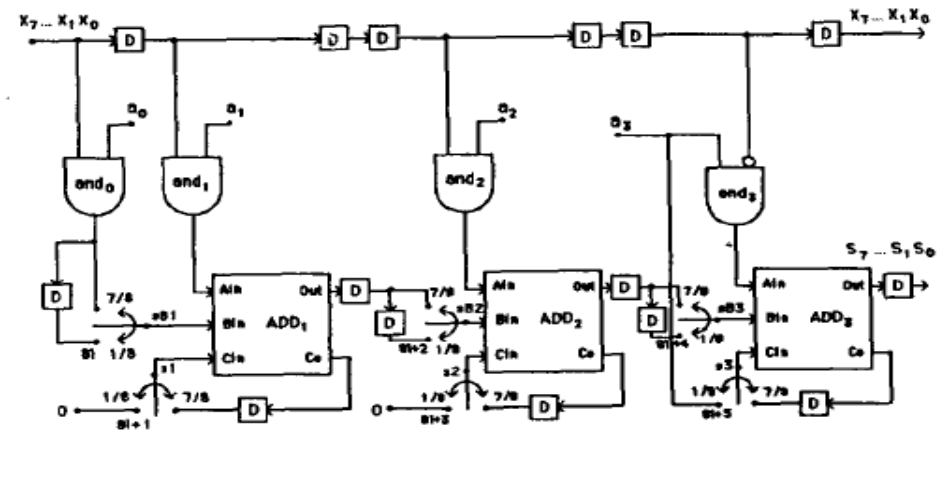

Figure 3:- A bit-serial 8x4-bit programmable-coefficient serial-parallel multiplier. The coefficient is time-invariant.

International Coherence on Application Specific Array Processors:

In this paper, we present general unfolding algorithms, which can unfold switches (for arbitrary unfolding factors), interpolators, and decimators. In our notation, multiple consecutive bits constitute a "digit". Further, if the digit-size is not a divisor of the word-length, then the digit processed in a clock cycle can contain bits of two consecutive words or samples. It is important to note that the unfolding algorithm makes use **of** the data-flow graph representation, and thus the technique is not limited to any particular number system. Unfolding also applies at the word-level as well as the bit-level (since word-level architectures reduce to bit-level architectures if the word-length is unity). For example, one can unfold a word-serial architecture to obtain a word-parallel architecture. Throughout this paper, we concentrate on two's complement fixed-point **data** representation, and least-significant-bit (Isb) first serial computation. We also assume a fraction representation of signals bounded between -1 and 1 (i.e. one sign bit is assumed to lie to the left of the decimal point). Furthermore. we assume the Isb is assigned a bit index.

The organization of the paper is as follows. Section 2 presents the unfolding algorithm and design of digit-serial architectures. illustrates unfolding of known bit- serial multipliers to obtain digit-serial multipliers. extensions of the unfolding algorithm to multiple sample rate operations, such as interpolators and decimators.

Conclusion:

We have proposed systematic unfolding algorithm to transform any bit-serial architecture into digit-serial architectures. Thus one can Erst synthesize a bit-serial architecture for a given algorithm or program, and then use the unfolding technique to synthesize a digit-serial architecture. We have also addressed the "folding technique" which is the reverse of the unfolding operation. critical path and power consumption of different digit-serial multipliers and their variation with respect to digit sizes have been explored. However, the comparison between the digit-serial and bit-parallel multipliers has not been addressed. This paper has presented a design methodology for a new class of digit-serial multiplier architectures. These architectures can be pipelined at the bit-level, and as a result power can be reduced.

References:

- H. Nguyen and A. Chatterjee, "Number-Splitting with Shift-and Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis," IEEE Trans. on VLSI, vol. 8, no. 4, pp. 419--424, 2000.
- R. Hartley, "Subexpression Sharing in Filters using Canonic Signed Digit Multipliers," IEEE TCAS II, vol. 43, no. 10, pp. 677-688, 1996.
- L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications," IEEE TCAD, vol. 27, no. 6, pp. 1013-1026,2008.
- A. Dempster and M. Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," IEEE TCAS II, vol. 42, no. 9, pp. 569-577, 1995.
- Y. Voronenko and M. Piischel, "Multiplierless Multiple Constant Multiplication," ACM Transactions on Algorithms, vol. 3, no. 2, 2007.
- R. Hartley and K. Parhi, Digit-Serial Computation. Kluwer Academic Publishers, 1995.
- K. Johansson, O. Gustafsson, and L. Wanhammar, "Multiple Constant Multiplication for Digit-Serial Implementation of Low Power FIR Filters," WSEAS Transactions on Circuits and Systems, vol. 5, no. 7, pp. 1001- 1008,2006.