Digital Filtering in FPGAs

Les Mintzer
Momentum Data Systems
Costa Mesa, California

Abstract

While the general purpose logic elements of the Field Programmable Gate Array (FPGA) appear to be unlikely candidates for implementing the multiply-intensive operations of digital filtering, the application of Distributed Arithmetic (DA) techniques turns the FPGA into a worthy contender. Indeed, in some important filter applications the Xilinx 4000 family of FPGAs offers superior performance over the fastest fixed point DSP microprocessors. A brief description of DA processing is presented to provide some background for the filter design examples that are presented. A simple FIR filter will serve to establish the design concepts, and a two dimensional Sobel edge detector will serve to illustrate the performance capabilities of this approach.

Introduction

The DSP engineer currently has two design choices - the DSP microprocessor or the dedicated function-specific device. The first can be programmed to cover a wide range of applications often not an optimum solution but, certainly satisfactory judging from the popularity of both fixed and floating point DSP chips. The second choice includes algorithm-specific devices such as FIR filters with video bandwidth sample rates and high performance application specific devices such as image data compression chips. The look-up table SRAM based FPGA offers a third choice - an alternative to both the programmed and dedicated hardware solutions. The paper is organized in four sections as follows:

1. A brief description of the Distributed Arithmetic (DA) techniques for implementing the weighted sum of products computations that underlie the filter and other important digital signal processes.
2. An introduction to the Xilinx family of FPGAs whose gate architecture provides a good platform for DA computations - indeed superior to all other FPGA types.
3. A simple, symmetrical FIR filter design is presented. This is a very direct application of the sum of products. A simple "trick" produces an efficient match to the FPGA circuits.
4. A second example of a two dimensional spatial filter demonstrates the efficiency and flexibility of the DA approach.

Distributed Arithmetic

Distributed Arithmetic [1] is a computational algorithm that affords efficient implementation of the weighted sum of products, or dot product, that defines important signal processing operators such as filters, frequency transformers, and artificial neural networks. The dot product is a multiply intensive computation whose speed is limited by the multiplier circuit. The parallel array multiplier found in all DSP chips and fourth generation microprocessors consumes many gates and is inappropriate for even the largest FPGAs. However, for linear, time-invariant systems (which apply to the operators listed above) where one factor of each product term is a constant, the multiplier may be replaced by more economic scaling and adding circuits. This is reminiscent of the programming of multiplication as a sequence of shifts and adds - a serial approach that requires fewer gates but operates at lower speeds. DA is a bit serial computation process, however, it offers speeds approaching those of the full array multiplier. The DA implementation of the dot product in FPGAs is detailed in reference [1] and some points to note are:

– Each product term consists of a variable (signal) and a constant (coefficient) both in fixed point binary format but not necessarily of the same word length.

– Rather than compute the product on a term by term basis, the partial products of all terms are computed simultaneously, and in the time it would take to compute a single partial product. This is achieved by precomputing all possible cumulative partial product outcomes and storing them in a look up table (LUT) which is addressed by the multiplier (variable) bits. All input variables are sequenced simultaneously, bit serial first to address the LUT. Each cumulative partial product is scaled up binarily and added to the accumulated partial products. When the most significant sign bits address the LUT, its outcome is subtracted from the accumulated partial products.

The complete dot product computation takes B clocks where B is the number of input variable bits, and is independent of the number of input variables.

The functional blocks of the DA data path are shown in figure 1; relatively few cover many applications. These blocks are comprised of standard logic circuits which may be scaled to meet particular dynamic range and accuracy requirements. In lieu of a full array multiplier and double precision accumulator, and a means of accessing pairs of operands, there are the following simple circuits:

1. A set of serial shift registers that can be loaded bit parallel.
2. A look up table (RAM or PROM) which is addressed by the serial outputs of the set of shift registers.
3. A single precision adder/subtractor with accumulator register. The register contents are scaled down by 1/2 prior to adding to the LUT output. As the process repeats with successively higher order partial products, the discarded bit in the scaling process is passed on to an auxiliary shift register thereby retaining the double precision result. Subtraction occurs on the final accessing of the LUT by the multiplier sign bits. All these functions are gathered into a scaling- accumulator block.

The basic DA path prevails over a wide range of applications. Thus with a fixed set of blocks the frequency response of a digital filter can be changed by simply changing the contents of the LUT. Similarly, the controls for the data path are very simple and remain fixed for many applications.

The Xilinx FPGA

Among the several FPGA manufacturers Xilinx was the first to use look up tables for constructing user logic. These logic truth tables are embedded in configurable logic blocks (CLBs) that also include a pair of D flip flops and clock enable and control circuits. Literally underlying the CLB is an SRAM (static random access memory) which stores the bit patterns that define the CLB logic functions and the paths linking them. The SRAM bit patterns are loaded in a variety of configuration modes, and, as with any RAM, can be reloaded or written into an unlimited number of times. Thus a design may be corrected or modified through programmatic configurations with no change in hardware. Configuration data may be downloaded from a host computer or may be transferred from local PROM automatically on power on.

The Xilinx XC4000 family [2] features a two dimensional matrix of CLB elements interconnected by a hierarchy of routing resources - all enclosed within a perimeter of programmable Input/ Output Blocks (IOBs). The IOB has a tri-state port which can be dynamically configured to function as either a source or sink with internal registers to capture input data, or deliver stable, clocked output data. The smallest family member (XC4002) has an 8x8 CLB matrix and 64 IOBs while the largest device (XC4025) has a 32x32 CLB matrix and 256 IOBs.

The XC4000 family is supported/by a wide range of software design tools developed by Xilinx and third-party providers such as Synopsys. Design entry may be via the schematic capture of a logic diagram using one of the industry standard CAE tools such as Viewdraw. After schematic or equation-based entry the design is automatically converted to the Xilinx Netlist Format. Next, the Xilinx XACT software partitions the design into logic blocks, then finds a near-optimal placement for each block, and, finally, selects the interconnect routing. All these partitioning, placement, and routing routines run automatically, but the designer may intervene by setting specific constraints or by editing critical portions of the design. The completed design is documented in a configuration data base file.

As these FPGA products mature, a library of proven, optimally designed macros have been developed to facilitate the designer's task. Thus one finds counters, shift registers, parallel adders and accumulators, RAMs, FIFOs, etc, which can be compiled or scaled to the desired dimensions to satisfy design requirements The DA circuits can be configured largely from blocks or macros already in the design library. These blocks are now being assembled into higher order signal processing macros such as the 8 tap FIR filter which will be described in the next section.

The Distributed Arithmetic FIR Filter

The FIR filter is a simple application of the DA technique. In the N tap filter of figure 2a the input variables are the outputs of the individual taps i.e.; delayed input samples. Thus once the input sample is converted from parallel to serial form a chain of serial shift registers provides the tap delays and the bit serial addressing of the LUT. The resulting DA circuit is shown in figure 2b. During each input sample period there are at least B shift clock periods.

Memory rapidly becomes a limiting factor in FIR DA designs. A 20-tap filter for example requires 2exp(20) words. Fortunately very significant memory reduction can be achieved for symmetric FIR filters. The number of addresses can be halved by first adding bit serial the outputs of symmetrical tap pairs. The resulting memory size of 2exp (10) represents a 1000 to 1 reduction. This is achieved at a cost of 10 serial full adder circuits and an additional shift clock to process the overflow of the serial adder.

An 8-tap FIR filler design example is first offered since it maps readily into the 16 word logic LUT of the Xilinx CLB. The data path design utilizing CLB components is shown in figure 3. Design details are offered in [2]. The entire FIR filter including input and output pads was placed and routed in a Xilinx 3042 FPGA. The original estimate of 54 CLBs was increased by four to minimize

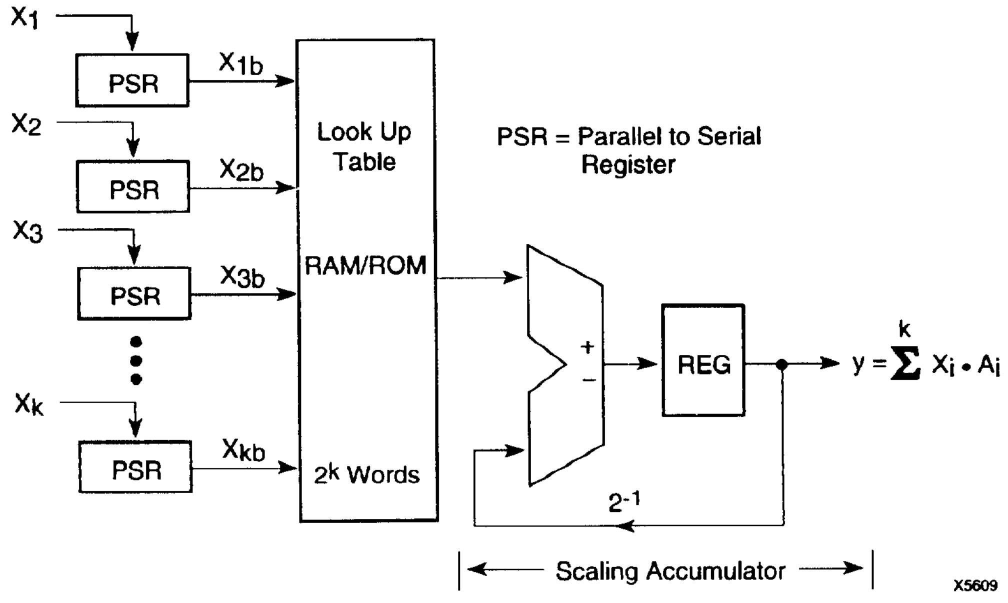

Figure 1. Distributed Arithmetic Processor

path delays. Initial delay analysis indicated a reliable system clock of 12.5 MHz. The long pole is the carry propagate path of the adder/subtractor. Further layout refinements could probably extend the clock to 15 MHz. A recent redesign with the XC4000 family yielded a lower CLB count by mechanizing the shift registers with random access memory rather that flip flops. Fast look ahead carry embedded in the adder circuits boosted performance to 25 MHz.

Image Edge Detectors

Image edge detectors such as the Sobel and Prewitt operators are two dimensional extensions of the basic FIR filter. The coefficients of these spatial filters are single integers with symmetries that can be exploited as in the example above. Furthermore the use of hardware shift registers (or RAM for the 4000 family) and the interconnect flexibility of the FPGA facilitates the acquisition and alignment of data for the filter processing. Both detector operators feature a 3x3 pixel template whose coefficient map as follows:

$$\begin{matrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{matrix}$$

and for the Sobel templates:

$$\text{vertical edge} \quad \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \text{horizontal edge} \quad \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

These templates are applied to a two dimensional image to determine if the central pixel of a 3x3 array is on a vertical or horizontal edge of the image. The pixel array has its incremental row/column coordinates as shown:

$$\begin{matrix} X11 & X12 & X13 \\ X21 & X22 & X23 \\ X31 & X32 & X33 \end{matrix}$$

The central pixel will be set to indicate a detected edge. The dot product computations are basically that of a pair of 9 tap FIR filter which are defined by the following equations:

$$Yv = X11(-1) + X12(0) + X13(1) + X21(-2) + X22(0) + X23(2) + X31(-1) + X32(0) + X33(1)$$

$$Yh = X11(1) + X12(2) + X13(1) + X21(0) + X22(0) + X23(0) + X31(-1) + X32(-2) + X33(-1)$$

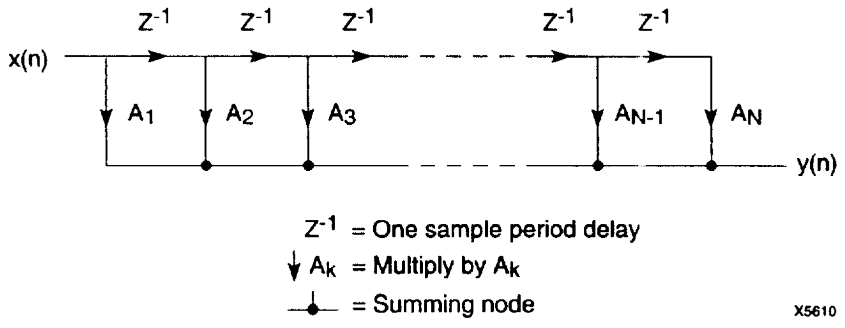

Figure 2a. Flow Diagram of FIR Filter

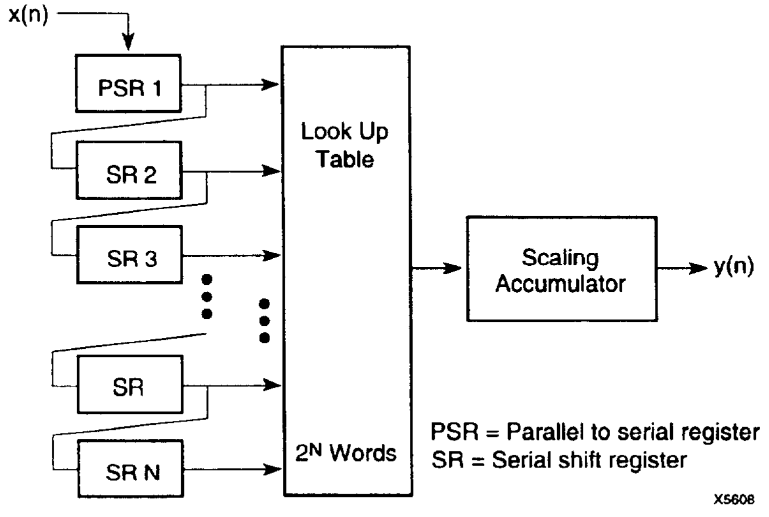

Figure 2b. N Tap FIR Filter With Distributed Arithmetic

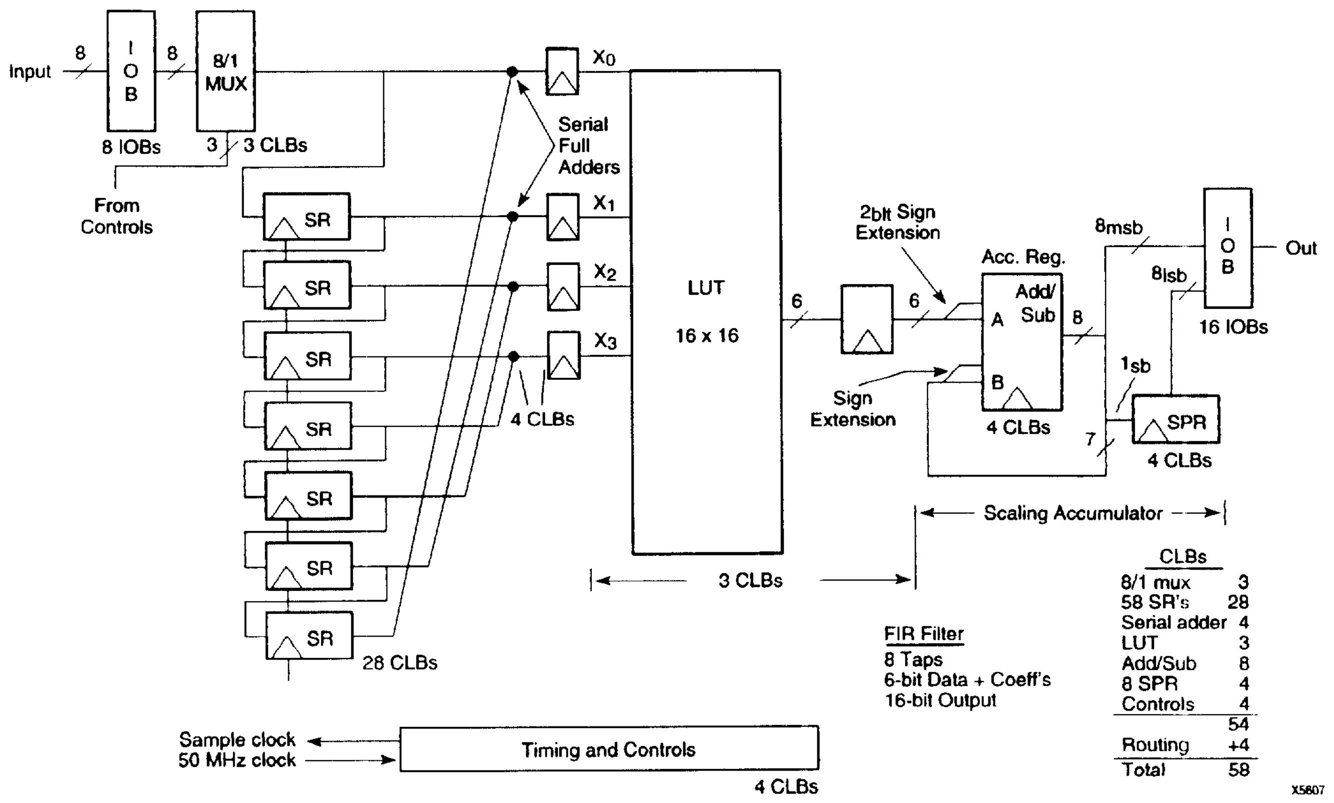

Figure 3. Distributed Arithmetic FIR Filter

The presence of an edge is determined by the magnitude of the resultant:

Y = SQRT[Yv**2 + Yh**2]

And, finally, a threshold is applied to Y to establish the presence of an edge pixel. For video image processing these computations are performed at HxVx 30 pixels/sec, where H and V denote the raster dimensions. A TV standard 30 frame/sec rate is assumed. Pixel processing rate grows with the fineness of image granularity. For a moderately fine grain image (1024x1024) the edge detection computations indicated above are made at a 31.46 mega-pixels/sec. This load exceeds the capability of the fastest DSP microprocessors, and cannot be partitioned gracefully even among several of them. Accordingly several chip manufacturers have designed dedicated 3x3 image filters to meet this need. One example is the Harris HSP48901 which can handle 8-bit data and coefficient values and can compute only one of the dot products at a maximum rate of 30 megapixels/sec.

Again exploiting symmetry (impossible in the standard DSP microprocessor) a significant hardware reduction is realized. The terms of the sum-of product equations above can be reduced by grouping pixel variables with the same coefficient values as sum or difference terms in the following way:

Yv = (X13 -X11)(1) + (X23 -X21)(2) + (X33 -X31)(1)

Yh = (X11 - X31)(1) + (X12 - X32)(2) + (X13 - X33)(1)

Further consolidation of terms with the factor "1" is also possible but would not yield further circuit savings. With 3 serial subtractors the LUT address lines are similarly reduced to 3. For 8-bit pixel data 9 serial shift clocks are required to include the final borrow bit. Thus the DA circuit block diagram for the Sobel filter data path can be developed in similar fashion as shown in figure 4 An initial hardware sizing indicates that 37 CLBs are required to compute Yv and Yh. The circuit for computing the magnitude of the resultant is not shown; however, a simple approximating algorithm can be implemented with 13 CLBs. Each color requires 50 CLBs and 150 CLBs for the three primary colors. Control circuits estimated to be less than 20 CLBs - are shared by the three color channels. With a modest 40 MHz system clock, the pixel processing rate is 40/9 or 4.444... megapixels/sec which can support a coarse (256x256) grained raster. By increasing the system clock to 60 MHz and by replicating the filter circuits to compute 5 contiguous edge pixels simultaneously, it now becomes possible to provide fine grained, three color edge detection in a single XC4025 FPGA.

References

[1] While, S.A. "Applications of Distributed Arithmetic to Digital Signal Processing. A Tutorial Review."
IEEE ASSP Magazine July 1989

[2] Mintzer L. "Fir Filters with the Xilinx FPGA"
First International ACM/SIGDA Workshop on FPGAs
February 1992 pp129-134.

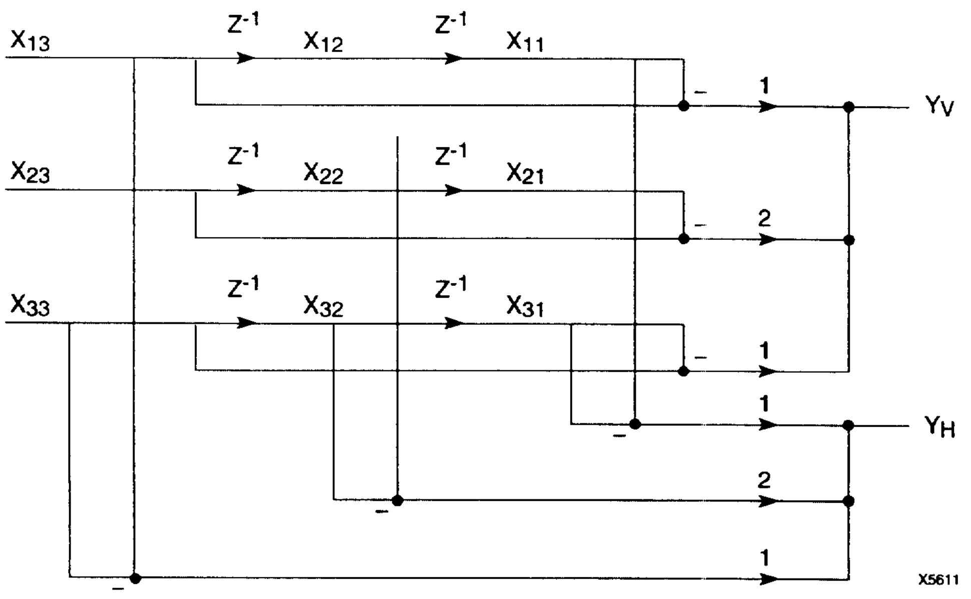

Figure 4a. Signal Data Flow Diagram of Sobel 2D Filter

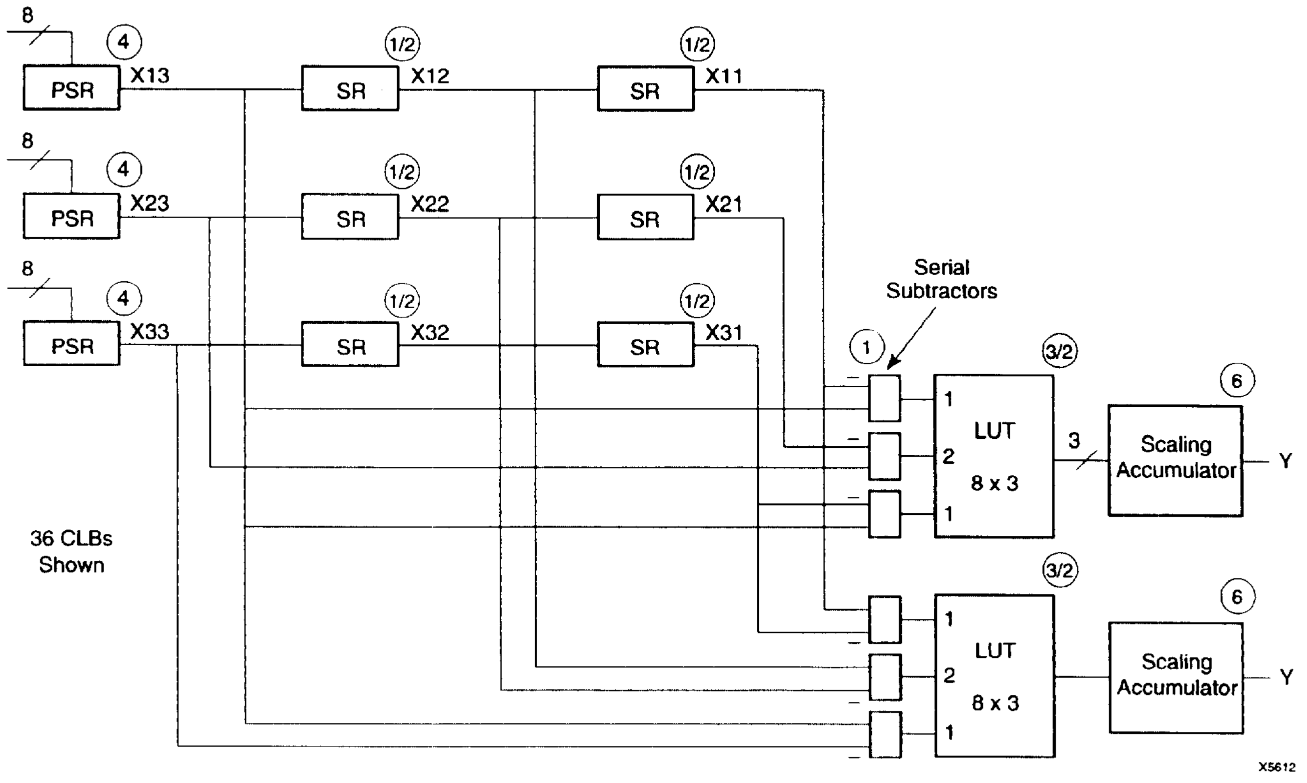

Figure 4b. DA Circuit Blocks of Sobel Filter