# RUNTIME RECONFIGURABLE DSP UNIT USING ONE'S COMPLEMENT AND MINIMUM SIGNED DIGIT

SCHULICH School of Engineering, UNIVERSITY OF CALGARY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Travis Manderson and Laurence Turner, *University of Calgary, Alberta, Canada*

## Abstract

A runtime reconfigurable Digital Signal Processing (DSP) unit using one's complement data and a Minimum Signed Digit (MSD) multiplier is shown. The MSD multiplier changes the partial product shift-and-add operations to shift-and-add/subtract operations and reduces the number of partial product terms by half, decreasing the size and increasing the speed of the multiplier.

One's complement data allows the multiplication by negative one (used in the subtract operation) to be a bitwise complement operation. Using a carry-save architecture, the propagation delay of an adder to add the carry-out bits produced in one's complement addition is eliminated. Taking advantage of the deterministic nature of the desired DSP algorithm, the runtime reconfiguration of the multiplier can be pipelined to eliminate an added set-up state.

The described DSP unit compared to a DSP unit using a conventional multiplier is shown to be 20% faster and 30% smaller for a 32-bit coefficient and using Xilinx Field Programmable Gate Arrays (FPGAs) with 6-input Look-Up Tables (LUTs).

## Overview

### Minimum Signed Digit

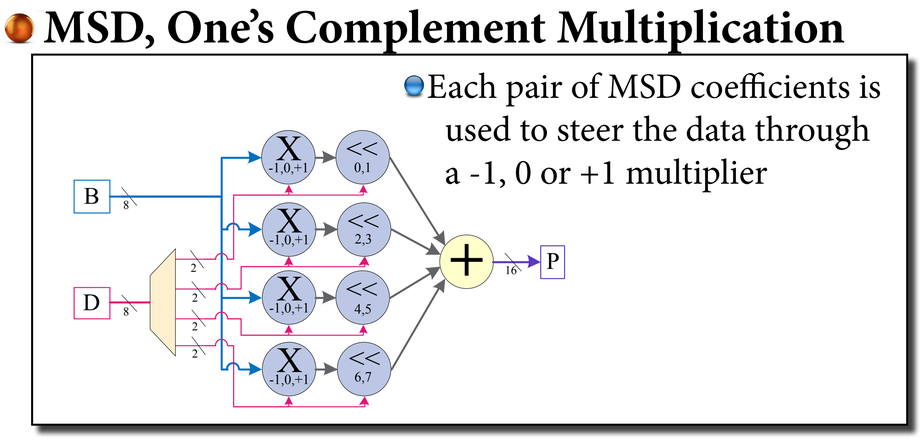
- Defined as a set of $\{0,1,\bar{1}\}$ digits where $\bar{1}$=-1
- Guarantees at most one non-zero bit for each consecutive pair of bits in the MSD coefficient
- Reduces the number of partial product terms in a multiplier by half
  - Decreases the size and increases the speed of the multiplier

$$M = \sum_{i=0}^{N-1} m_i 2^i \text{ where } m_i \subset \{0, 1, \bar{1}\},\ \bar{1} = -1$$

### One's Complement

- The negative of a number is its bitwise complement
- Left shift operations are end-around-carry
- Any carry-out bit from addition is added back to the LSB

### MSD, One's Complement Multiplication

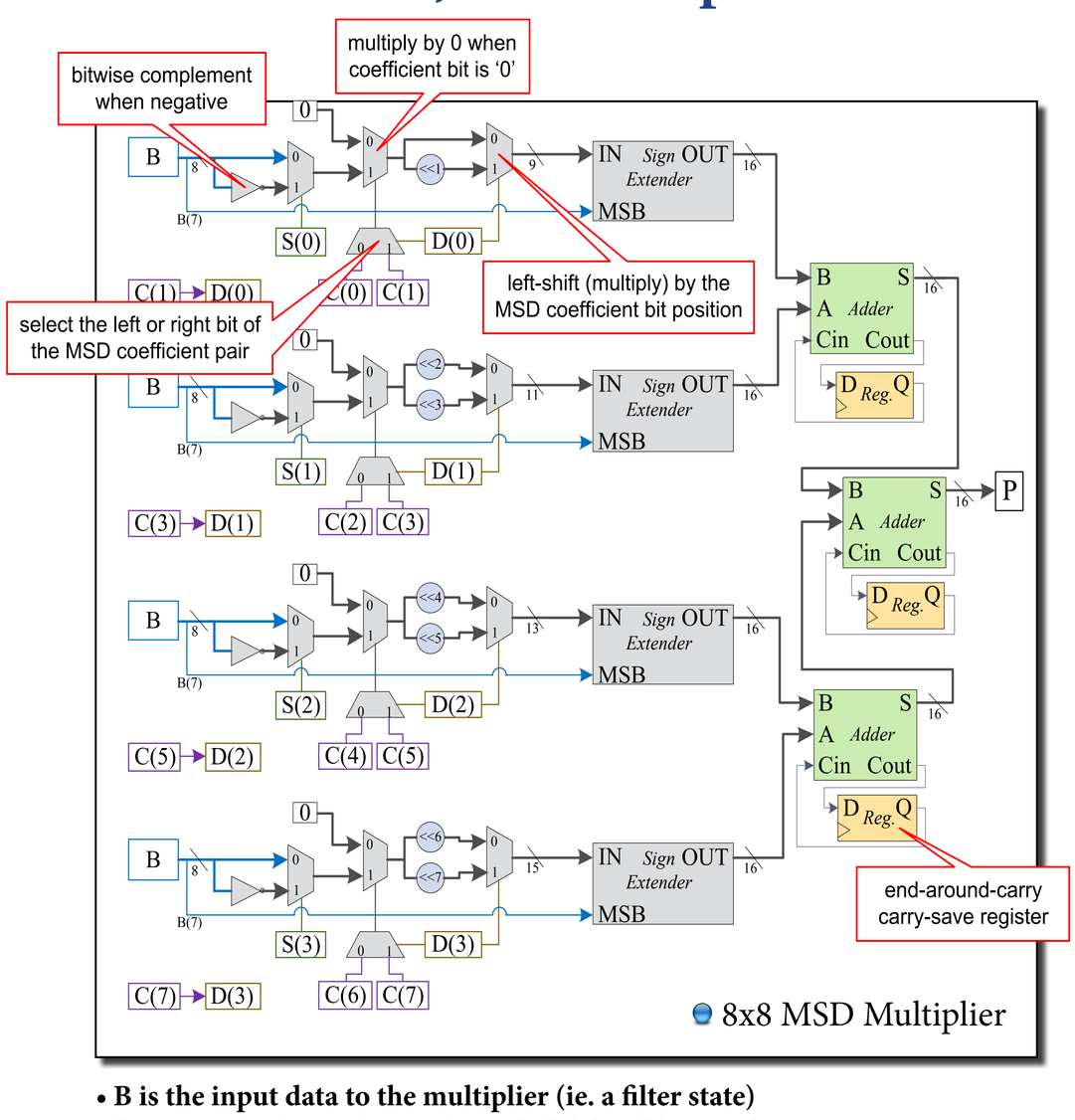



- Each pair of MSD coefficients is used to steer the data through a -1, 0 or +1 multiplier

- Preparing the MSD Coefficient:

| | |
|---|---|
| Determine MSD representation of coefficient (ie. 93): | $93 = 10\bar{1}00\bar{1}01$ |
| Break into pairs of digits: | $10\|\bar{1}0\|0\bar{1}\|01$ |
| Define C as the MSD magnitude: | $C = 10100101$ |
| Define D as the justification of every pair of digits (0 or 1 when both digits are 0): | $D = 1100$ |
| Define S as the sign of the non-zero digit (1 when the digit is -1): | $S = 0110$ |

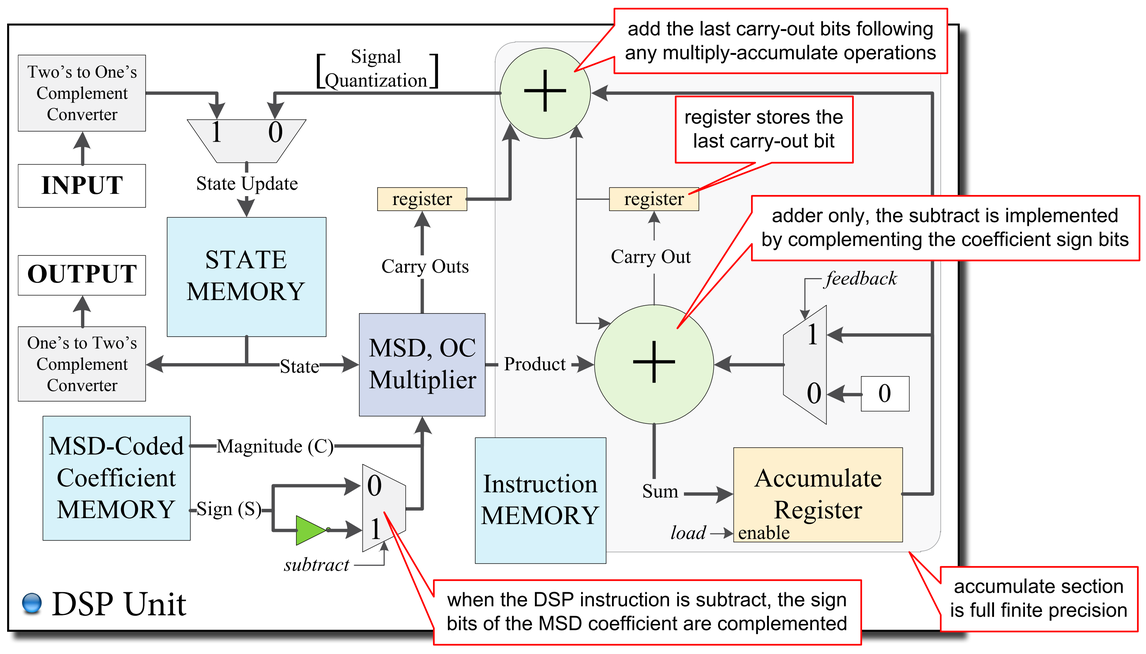
## MSD, OC Multiplier




8x8 MSD Multiplier

- **B is the input data to the multiplier (ie. a filter state)**
- **D can be derrived directly from C: D(n) = C(n*2 + 1)**
- **C and S are stored in memory and represent the multiplier coefficient**

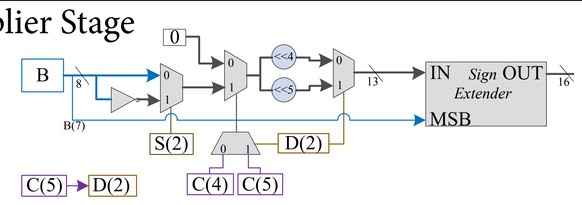
## DSP Unit Architecture




- **Multiplier is full finite precision**
- **The multiplier coefficient (sign and magnitude) is pipelined so there is no set-up instruction to reconfigure the multiplier**

## MSD, OC Multiplication

Single MSD Multiplier Stage




- **Partial Product of a single MSD multiplier stage:**

$$PP(i) = B * [S(i)? -1 : 1] * C(2 * i + D(i)) * 2^{2*i+D(i)}$$

- **Product of the MSD multiplier:**

$$\sum_{i=0}^{i=\frac{N}{2}} PP(i) = \sum_{i=0}^{i=\frac{N}{2}} B * [S(i)? -1 : 1] * C(2 * i + D(i)) * 2^{2*i+D(i)}$$

- **Partial Product Summation Example:**

MSD Coefficient (ie. 93): $C = 10100101$ $D = 1100$ $S = 0110$

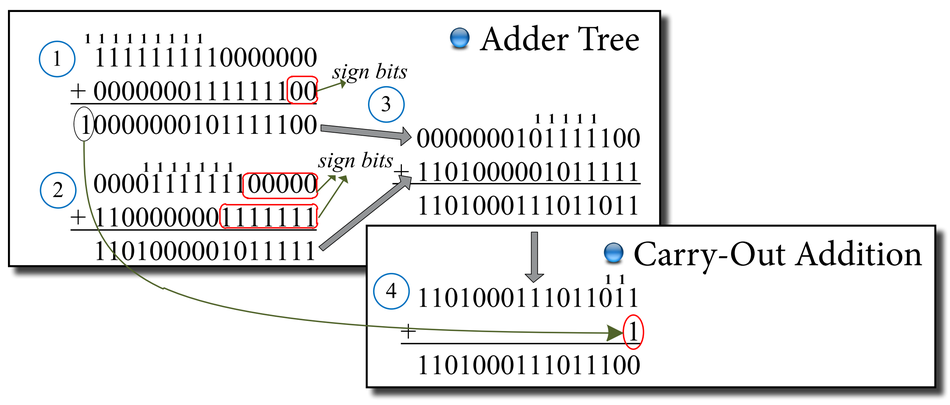
$$\begin{array}{l} B * [S(0)? -1 : 1] * C(0 + D(0)) * 2^{0+D(0)} \\ +B * [S(1)? -1 : 1] * C(2 + D(1)) * 2^{2+D(1)} \\ +B * [S(2)? -1 : 1] * C(4 + D(2)) * 2^{4+D(2)} \\ +B * [S(3)? -1 : 1] * C(6 + D(3)) * 2^{6+D(3)} \end{array} = \begin{array}{l} \text{B*1*1*}2^0 \\ +\ \text{B*-1*1*}2^2 \\ +\ \text{B*-1*1*}2^5 \\ +\ \text{B*1*1*}2^7 \end{array} = \begin{array}{l} \text{B*1} \\ +\ \text{B*-4} \\ +\ \text{B*-32} \\ +\ \text{B*128} \end{array} = \textbf{B*93}$$

Multiplication of One's Complement Number Example:

Input Data (ie. -127) in One's Complement: $B = 10000000$

$$\text{Product (P)} = \begin{array}{l} PP(0) \\ +\ PP(1) \\ +\ PP(2) \\ +\ PP(3) \end{array} = \begin{array}{l} \text{10000000*1*1*}2^0 \\ +\ \text{10000000*-1*1*}2^2 \\ +\ \text{10000000*-1*1*}2^5 \\ +\ \text{10000000*1*1*}2^7 \end{array} = \begin{array}{l} \text{1111111110000000} \\ +\ \text{0000000111111100} \\ +\ \text{0000111111100000} \\ +\ \text{1100000001111111} \end{array}$$

- **Addition of Partial Products:**




So, **P = 1101000111011100 = -11811**

## Results

Synthesis Results using Xiling ISE

| Xilinx FPGA | MSD, One's Complement | | Two's Complement LUT Ripple-Carry Multiplier | | Using DSP hard IP (DSP48) | |
|---|---|---|---|---|---|---|
| | LUT's | Latency | LUT's | Latency | LUT's | Latency |
| Spartan-3A (4-input LUT's) | 1902 | 19.126 ns | 1307 | 21.077 ns | 160 | 21.630 ns |
| Virtex-5 (6-input LUT's) | 1186 | 7.650 ns | 1743 | 9.633 ns | 160 | 10.440 ns |

- Up to 20% faster than a LUT based multiplier
- Up to 30% smaller than a LUT based multiplier
- Marginally faster than the DSP48 slice found in select Xilinx FPGAs
- MSD One's Complement DSP Unit could be used in an ASIC