**Summary of DSP Designing For Optimal Results**

The DSP48 slice is a new element in the Xilinx development model referred to as “Application  
Specific Modular Blocks” (ASMBL). The purpose of this model is to deliver off-the-shelf  
programmable devices with the best mix of logic, memory, I/O, processors, clock management, and  
digital signal processing. ASMBL is an efficient FPGA development model for delivering off-theshelf, flexible solutions ideally suited to different application domains.  
Each XtremeDSP tile contains two DSP48 slices to form the basis of a versatile coarse-grain DSP  
architecture. Many DSP designs follow a multiply with addition. In Virtex™-4 devices these elements  
are supported in dedicated circuits.  
The DSP48 slices support many independent functions, including multiplier, multiplieraccumulator (MAC), multiplier followed by an adder, three-input adder, barrel shifter, wide bus  
multiplexers, magnitude comparator, or wide counter. The architecture also supports connecting  
multiple DSP48 slices to form wide math functions, DSP filters, and complex arithmetic without the  
use of general FPGA fabric.  
The DSP48 slices available in all Virtex-4 family members support new DSP algorithms and  
higher levels of DSP integration than previously available in FPGAs. Minimal use of general FPGA  
fabric leads to low power, very high performance, and efficient silicon utilization.

The DSP48 slices facilitate higher levels of DSP integration than previously possible in FPGAs. Many

DSP algorithms are supported with minimal use of the general-purpose FPGA fabric, resulting in low

power, high performance, and efficient device utilization.

At first look, the DSP48 slice is an 18 x 18 bit two’s complement multiplier followed by a 48-bit sign-extended adder/subtracter/accumulator, a function that is widely used in digital signal processing (DSP).

A second look reveals many subtle features that enhance the usefulness, versatility, and speed of this arithmetic building block. Programmable pipelining of input operands, intermediate products, and accumulator outputs enhances throughput.

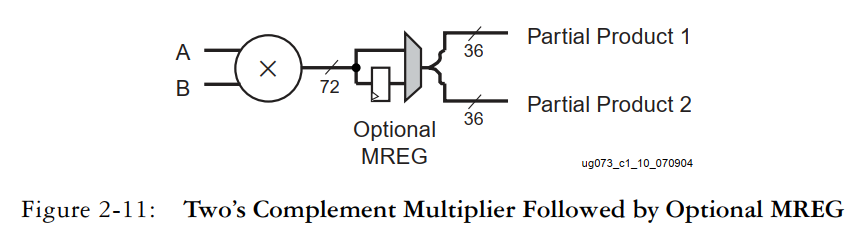
The 48-bit internal bus allows for practically unlimited aggregation of DSP slices.

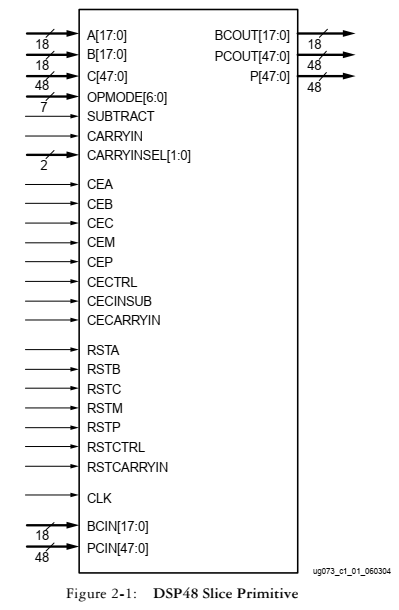
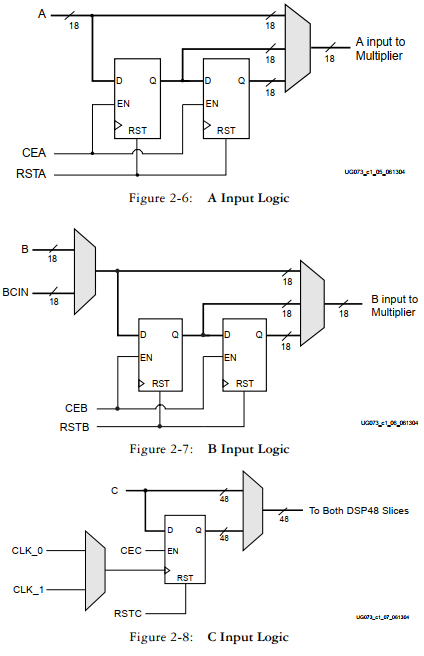
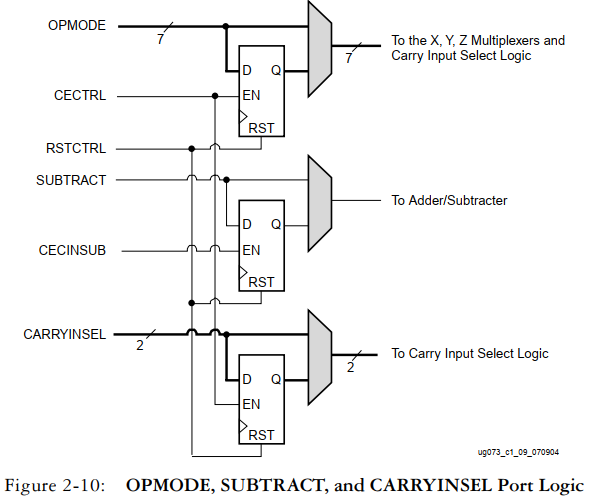
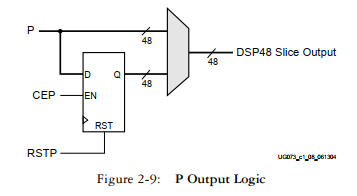
One of the most important features is the ability to cascade a result from one XtremeDSP Slice to the next without the use of general fabric routing. This path provides high-performance and lowpower post addition for many DSP filter functions of any tap length.

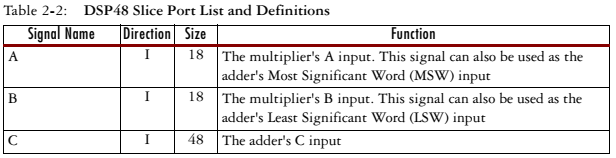
For multi-precision arithmetic this path supports a right-wire-shift. Thus a partial product from one XtremeDSP Slice can be right-justified and added to the next partial product computed in an adjacent such slice. Using this technique, the XtremeDSP Slices can be configured to support any size operands.

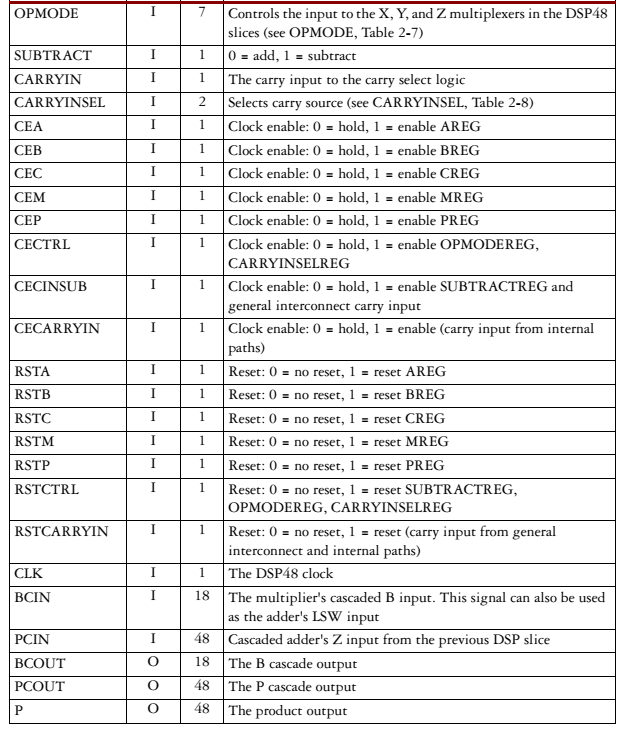
Another key feature for filter composition is the ability to cascade an input stream from slice to slice.

The C input port, allows the formation of many 3-input mathematical functions, such as 3-input addition, 2-input multiplication with a single addition. One subset of this function is the very valuable support of rounding a multiplication “away from zero”.

Architecture highlights of the DSP48 slices are:  
• 18-bit by 18-bit, two's-complement multiplier with a full-precision 36-bit result, sign  
extended to 48 bits  
• Three-input, flexible 48-bit adder/subtracter with optional registered accumulation feedback  
• Dynamic user-controlled operating modes to adapt DSP48 slice functions from clock cycle to  
clock cycle  
• Cascading 18-bit B bus, supporting input sample propagation  
• Cascading 48-bit P bus, supporting output propagation of partial results  
• Multi-precision multiplier and arithmetic support with 17-bit operand right shift to align  
wide multiplier partial products (parallel or sequential multiplication)  
• Symmetric intelligent rounding support for greater computational accuracy  
• Performance enhancing pipeline options for control and data signals are selectable by  
configuration bits  
• Input port “C” typically used for multiply-add operation, large three-operand addition, or  
flexible rounding mode  
• Separate reset and clock enable for control and data registers







**DSP48 Slice Attributes**  
The synthesis attributes for the DSP48 slice are described in detail throughout this section. With the  
exception of the B\_INPUT and LEGACY\_MODE attributes, all other attributes call out pipeline  
registers in the control and datapaths. The value of the attribute sets the number of pipeline registers.  
The attribute settings are as follows:  
• The AREG and BREG attributes can take a value of 0, 1, or 2. The values define the number  
of pipeline registers in the A and B input paths. See the “A, B, C, and P Port Logic” section  
for more information.  
• The CREG, MREG, and PREG attributes can take a value of 0 or 1. The value defines the  
number of pipeline registers at the output of the multiplier (MREG) (shown in Figure 2-11)  
and at the output of the adder (PREG) (shown in Figure 2-9). The CREG attribute is used to  
select the pipeline register at the 'C' input (shown in Figure 2-8).  
• The CARRYINREG, CARRYINSELREG, OPMODEREG, and SUBTRACTREG attributes  
take a value of 0 if there is no pipelining register on these paths, and take a value of 1 if there  
is one pipeline register in their path. The CARRYINSELREG, OPMODEREG, and  
SUBTRACTREG paths are shown in Figure 2-10, and the CARRYINREG path is shown in  
Figure 2-12.  
• The B\_INPUT attribute defines whether the input to the B port is routed from the parallel  
input (attribute: DIRECT) or the cascaded input from the previous slice (attribute:  
CASCADE).  
• The LEGACY\_MODE attribute serves two purposes. The first purpose is similar in nature to  
the MREG attribute. It defines whether or not the multiplier is "flow through" in nature (i.e.,  
LEGACY\_MODE value equal to MULT18x18) or contains a single pipeline register in the  
middle of the multiplier (i.e., LEGACY\_MODE value equal to MULT18x18S is the same as  
MREG value equal to one). While this is redundant to the MREG attribute, it was deemed  
useful for customers used to the Virtex-II and Virtex-II Pro multipliers since the DSP48 setup  
and hold timing most closely matches those of the Virtex-II and Virtex-II Pro MULT18x18S  
when the MREG is used. Any disagreement between the MREG attribute and  
LEGACY\_MODE attribute settings are flagged as a software Design Rule Check (DRC) error.  
The second purpose for the attribute is to convey to the timing tools whether the A and B port  
through the combinatorial multiplier path (slower timing) or faster X multiplexer bypass  
path for A:B should be used in the timing calculations. Since the OPMODE can change  
dynamically, the timing tools cannot determine this without an attribute.  
To summarize the timing tools behavior:  
♦ If (attribute: NONE), then timing analysis/simulation bypasses the multiplier for the  
highest performance. The lowest power dissipation is achieved by setting MREG to one  
while CEM input is grounded.  
♦ If (attribute: MULT18x18), then timing analysis/simulation uses the combinatorial path  
through the multiplier. In this case, MREG must be set to zero or a DRC error occurs.  
♦ If (attribute: MULT18x18S), then timing analysis/simulation uses a pipelined multiplier.  
In this case MREG must be set to one or a DRC error occurs.

**Attributes in VHDL**DSP48 generic map(

AREG => 1,-- Number of pipeline registers on the A input, 0, 1 or 2

BREG => 1,-- Number of pipeline registers on the B input, 0, 1 or 2

B\_INPUT => “DIRECT”, -- B input DIRECT from fabric or CASCADE from another DSP48

CARRYINREG => 1, -- Number of pipeline registers for the CARRYIN input, 0 or 1

CARRYINSELREG => 1, -- Number of pipeline registers for the -- CARRYINSEL, 0 or 1

CREG => 1, -- Number of pipeline registers on the C input, 0 or 1

LEGACY\_MODE => “MULT18X18S”, -- Backward compatibility, NONE, MULT18X18 or MULT18X18S

MREG => 1, -- Number of multiplier pipeline registers, 0 or 1

OPMODEREG => 1,-- Number of pipeline registers on OPMODE input, 0 or 1

PREG => 1, -- Number of pipeline registers on the P output, 0 or 1

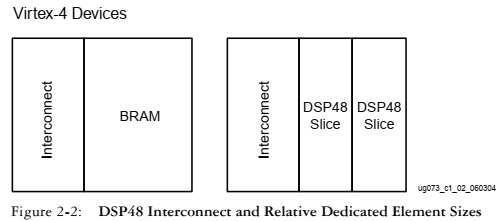
SIM\_X\_INPUT => “GENERATE\_X\_ONLY”,-- Simulation parameter for behavior for X on input.

-- Possible values: GENERATE\_X, NONE or WARNING

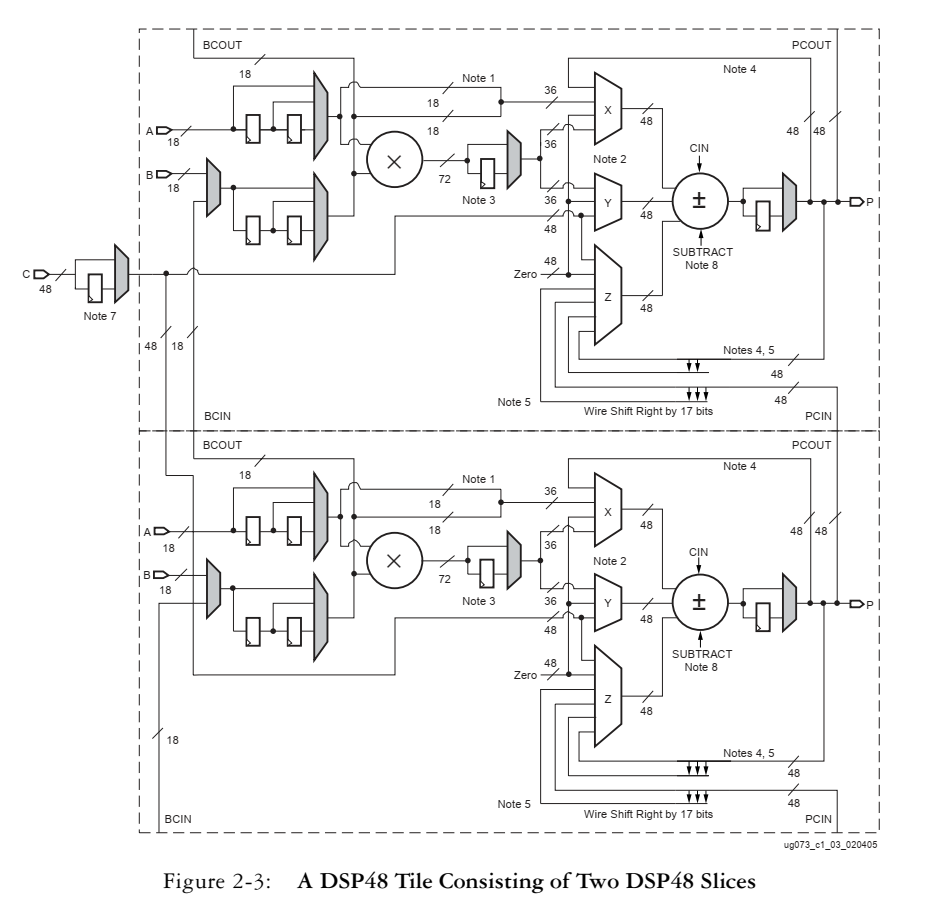
SUBTRACTREG => 1 -- Number of pipeline registers on the SUBTRACT input, 0 or 1

)

**DSP48 Tile and Interconnect**Two DSP48 slices, a shared 48-bit C bus, and dedicated interconnect form a DSP48 tile. The DSP48  
tiles stack vertically in a DSP48 column. The height of a DSP48 tile is the same as four CLBs and also  
matches the height of one block RAM. This “regularity” enhances the routing of wide datapaths.  
Smaller Virtex-4 family members have one DSP48 column while the larger Virtex-4 family members  
have two, four, or eight DSP48 columns. the multipliers and block RAM share interconnect resources in the  
Virtex-II and Virtex-II Pro architectures. Virtex-4 devices, however, have independent routing for the  
DSP48 tiles and block RAM, effectively doubling the available data bandwidth between the elements.

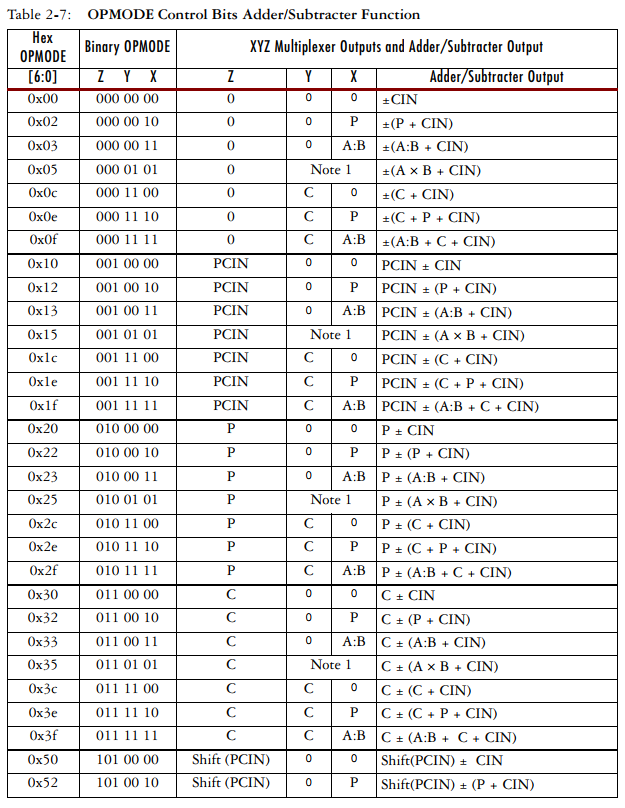


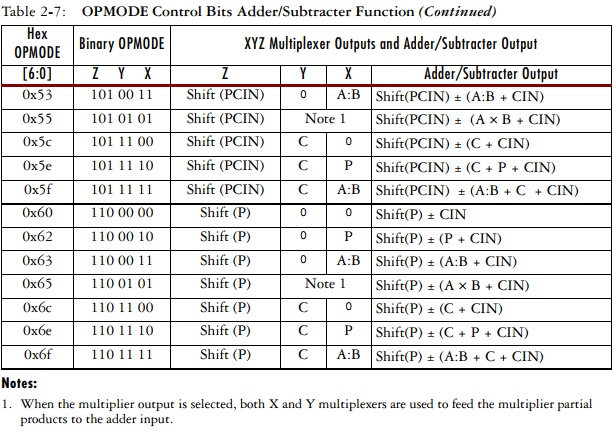
two DSP48 slices and their associated datapaths stacked vertically in a DSP48  
column. The inputs to the shaded multiplexers are selected by configuration control signals. These are  
set by attributes in the HDL source code or by the User Constraint File (UCF).



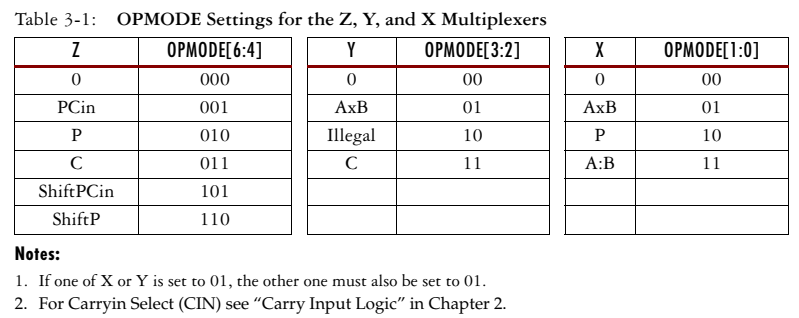
Notes:  
1. The 18-bit A bus and B bus are concatenated, with the A bus being the most significant.  
2. The X,Y, and Z multiplexers are 48-bit designs. Selecting any of the 36-bit inputs provides a  
48-bit sign-extended output.  
3. The multiplier outputs two 36-bit partial products, sign extended to 48 bits. The partial  
products feed the X and Y multiplexers. When OPMODE selects the multiplier, both X and Y  
multiplexers are utilized and the adder/subtracter combines the partial products into a valid  
multiplier result.  
4. The multiply-accumulate path for P is through the Z multiplexer. The P feedback through the X  
multiplexer enables accumulation of P cascade when the multiplier is not used.  
5. The “Right Wire Shift by 17 bits” path truncates the lower 17 bits and sign extends the upper 17  
bits.  
6. The grey-colored multiplexers are programmed at configuration time.  
7. The shared C register supports multiply-add, wide addition, or rounding.  
8. Enabling SUBTRACT implements Z – (X+Y+CIN) at the output of the adder/subtracter.

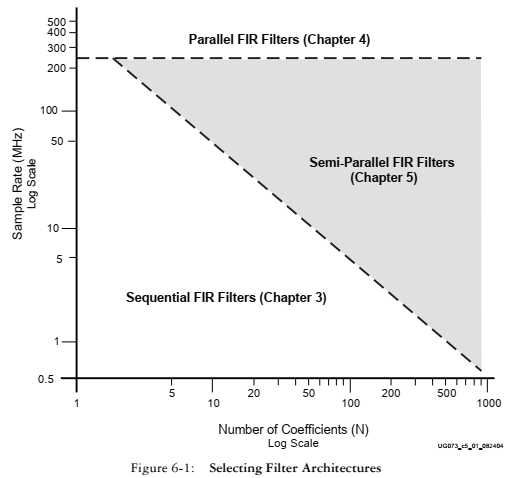
VHDL Instantiation Template

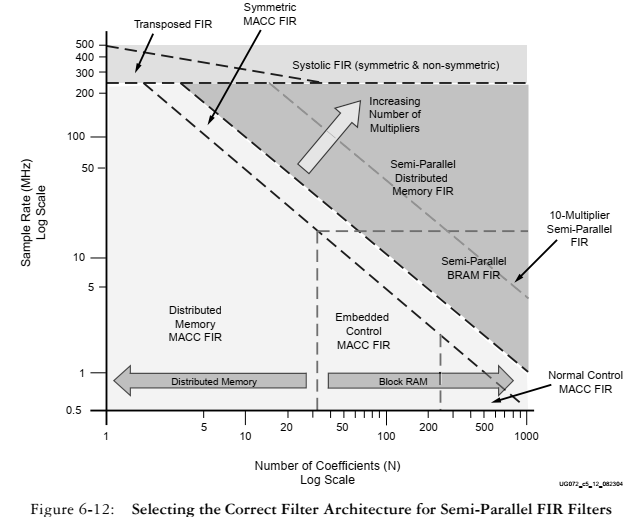
-- DSP48 : In order to incorporate this function into thedesign,  
-- VHDL : the following instance declaration needs to be placed  
-- instance : in the body of the design code. The instance name  
-- declaration : (DSP48\_inst) and/or the port declarations after the  
-- code : “=>” declaration maybe changed to properly reference and  
-- : connect this function to the design. All inputs and outputs must be connected.  
-- Library : In addition to adding the instance declaration, a use  
-- declaration : statement for the UNISIM.vcomponents library needs  
-- for : to be added before the entity declaration. This library  
-- Xilinx : contains the component declarations for all Xilinx  
-- primitives : primitives and points to the models that will be used for simulation.  
-- Copy the following two statements and paste them before the  
-- Entity declaration, unless they already exists.  
Library UNISIM;  
use UNISIM.vcomponents.all;  
-- <------------Cut code below this line and paste into the architecture body---------->  
-- DSP48: DSP Function Block  
-- Virtex-4  
-- Xilinx HDL Language Template version 6.1i  
DSP48\_inst: DSP48 generic map (  
AREG => 1, -- Number of pipeline registers on the A input, 0, 1 or 2  
BREG => 1, -- Number of pipeline registers on the B input, 0, 1 or 2  
B\_INPUT => “DIRECT”, -- B input DIRECT from fabric or CASCADE from anotherDSP48  
CARRYINREG => 1, -- Number of pipeline registers for the CARRYIN input,0 or 1  
CARRYINSELREG => 1, -- Number of pipeline registers for the CARRYINSEL,0 or 1  
CREG => 1, -- Number of pipeline registers on the C input, 0 or 1  
LEGACY\_MODE => “MULT18X18S”, -- Backward compatibility, NONE,MULT18X18 or MULT18X18S  
MREG => 1, -- Number of multiplier pipeline registers, 0 or 1  
OPMODEREG => 1, -- Number of pipeline registers on OPMODE input, 0 or 1  
PREG => 1, -- Number of pipeline registers on the P output, 0 or 1  
SIM\_X\_INPUT => “GENERATE\_X\_ONLY”, -- Simulation parameter forbehavior for X on input.  
-- Possible values: GENERATE\_X,NONE or WARNING  
SUBTRACTREG => 1) -- Number of pipeline registers on the SUBTRACT input,0 or 1  
port map (  
BCOUT => BCOUT, -- 18-bit B cascade output  
P => P, -- 48-bit product output  
PCOUT => PCOUT, -- 38-bit cascade output  
A => A, -- 18-bit A data input  
B => B, -- 18-bit B data input  
BCIN => BCIN, -- 18-bit B cascade input  
C => C, -- 48-bit cascade input  
CARRYIN => CARRYIN, -- Carry input signal  
CARRYINSEL => CARRYINSEL, -- 2-bit carry input select  
CEA => CEA, -- A data clock enable input  
CEB => CEB, -- B data clock enable input  
CEC => CEC, -- C data clock enable input  
CECARRYIN => CECARRYIN, -- CARRYIN clock enable input  
CECINSUB => CECINSUB, -- CINSUB clock enable input  
CECTRL => CECTRL, -- Clock Enable input for CTRL registers  
CEM => CEM, -- Clock Enable input for multiplier registers  
CEP => CEP, -- Clock Enable input for P registers  
CLK => CLK, -- Clock input  
OPMODE => OPMODE, -- 7-bit operation mode input  
PCIN => PCIN, -- 48-bit PCIN input  
RSTA => RSTA, -- Reset input for A pipeline registers  
RSTB => RSTB, -- Reset input for B pipeline registers  
RSTC => RSTC, -- Reset input for C pipeline registers  
RSTCARRYIN => RSTCARRYIN, -- Reset input for CARRYIN registers  
RSTCTRL => RSTCTRL, -- Reset input for CTRL registers  
RSTM => RSTM, -- Reset input for multiplier registers  
RSTP => RSTP, -- Reset input for P pipeline registers  
SUBTRACT => SUBTRACT -- SUBTRACT input  
); -- End of DSP48\_inst instantiation

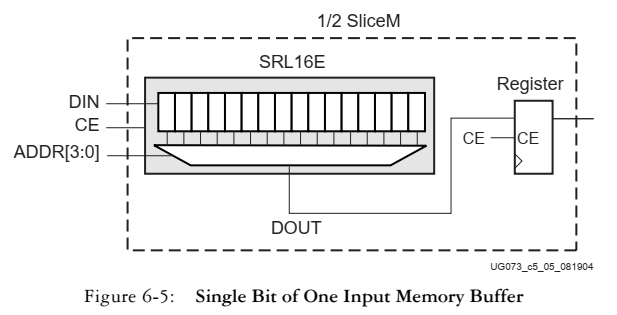
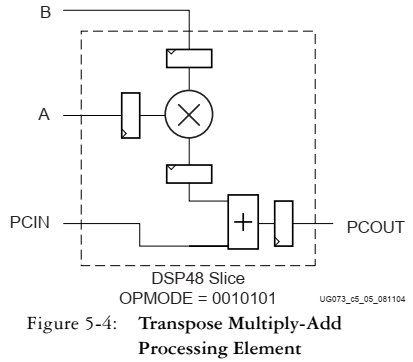


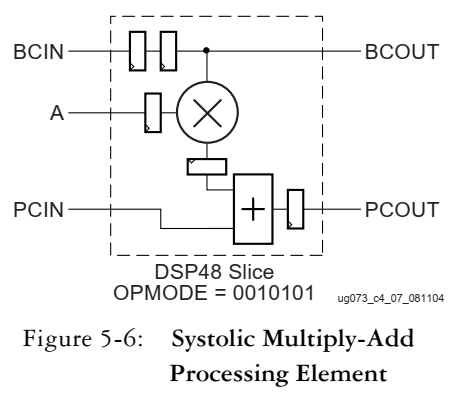
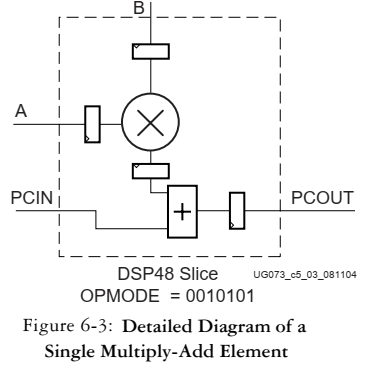
**DSP48 Slice Math Functions**  
The DSP48 slice efficiently performs a wide range of basic math functions, including **adders**,  
**subtracters**, **accumulators**, **MACs**, **multiply**, **multiplexers**, **counters**, **dividers**, **square-root functions**,  
and **shifters**. The optional pipeline stage within the DSP48 tile ensures high performance arithmetic  
functions. The DSP48 column structure and associated routing provides fast routing between DSP48  
tiles with less routing congestion to the FPGA fabric.

** Multiply Accumulate (MAC)**  
The DSP48 slice allows two 18-bit numbers to be multiplied together, and the product to be added to  
or subtracted from a previous product, a “0”, or a shifted product. In addition, rounding of any of the  
add, subtract, previous product, 0, or shifted product options is also possible.  
The input added or subtracted from the product is from the output of the Z multiplexer. This  
output is set using the corresponding OPMODE setting as shown in Table 3-1. Cascade the MAC tree  
by selecting the PCIN signal from the previous slice as the output from the Z multiplexer.

**FIR FILTERS**







The general FIR filter equation of a summation of products as defined in Equation 5-1.



Due to FPGA resources and methodology, a wide variety of adaptations of structures such as FIR can be made. The architectural approach is determined by negotiating resource-performance, mostly focused on the needs of the design. The FIR structure that meets the needs of the design can be determined in the table below, which compares different types of FIR filters according to their structures.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FIR Filter Type** | **Architecture & Implementation** | **Resource Utilization** | **Performance** | **Pros** | **Cons** | **Additional Metrics (Complexity/Scalability)** |
| Basic FIR Filter | Straightforward multiply–accumulate chain without advanced pipelining or optimization | Low to moderate; uses general logic and a few DSP slices | Suitable for low/moderate speeds; higher latency at scale | Simple design; easy to implement | Limited performance; not optimized for high tap counts | Very low design complexity; limited scalability |
| Single-Multiplier MAC FIR Filter | Single multiplier cascaded with MAC operations using pipelined DSP48 slices | Very low; one DSP slice per filter chain | Moderate throughput; fits moderate sample rates | Low hardware cost; simple control logic | Throughput limited by single multiplier usage | Low complexity; not easily scaled for very high-speed applications |
| Symmetric MAC FIR Filter | Exploits coefficient symmetry to reduce the number of operations and improve speed | Slightly higher control/memory demands for symmetry | Improved throughput (potentially near–double effective rate) | Enhanced performance when coefficients are symmetric; efficient coefficient usage | Only applicable for symmetric designs; requires additional memory port considerations | Moderate complexity; efficient for symmetric filter designs |
| Dual-Multiplier MAC FIR Filter | Implements two multipliers working in parallel with an accumulation chain | Higher (uses two DSP slices per filter) | High throughput; well suited for high sample rate applications | Increased data processing speed; better performance for demanding applications | Increased resource and power consumption | Higher design complexity; scalability limited by available DSP resources |
| Parallel FIR Filter (Regular) | Fully parallel implementation where each tap is computed concurrently with its dedicated multiplier and adder | Very high; one multiplier (and associated logic) per tap | Very high throughput; low latency | Maximizes speed and minimizes delay | Significant resource usage; impractical for very long (high-tap) filters | High complexity; scalability limited by FPGA area |
| Transposed FIR Filter | Transposed structure that inherently pipelines the operations; data flows “backward” compared to direct form | High (similar to fully parallel, though critical-path may be reduced) | High throughput with improved pipelining characteristics | Lower critical-path delay; high operating frequency | Resource intensive; requires careful clock and routing management | Moderate-to-high complexity; scalable if resources permit |
| Systolic FIR Filter | Implements a systolic array with regular, time–scheduled data propagation among processing elements | Moderate to high; efficient mapping onto FPGA fabric | High throughput with balanced latency | Highly regular and scalable structure; excellent for pipelined designs | Can introduce extra latency for short filters; more complex data scheduling | Good scalability; moderate design complexity |
| Symmetric Systolic FIR Filter | Systolic array architecture optimized for symmetric coefficients (reducing the number of multipliers required) | Lower than full systolic (saves multipliers via symmetry) | High throughput with resource savings | Resource–efficient for symmetric filters; maintains high speed | Limited to filters with symmetric coefficients; adds design constraints | Moderate complexity; excellent scalability for symmetric applications |
| Four-Multiplier Distributed-RAM-Based Semi-Parallel FIR Filter | Semi-parallel structure sharing four multipliers across several taps using distributed RAM for data and coefficient buffering | Moderate; fewer multipliers than full parallel; uses distributed RAM | Balances throughput and resource savings | Reduced resource usage compared to full parallel; flexible coefficient storage | Increased control logic complexity; potential extra latency due to sharing | Good scalability; moderate design complexity |
| Three-Multiplier Block RAM-Based Semi-Parallel FIR Filter | Similar to the four–multiplier version but employs block RAM for coefficient storage to optimize memory usage | Lower than full parallel; efficient use of embedded block RAM | Competitive throughput; may be slightly lower than four–multiplier variant | Efficient resource utilization; lower power consumption | Fixed coefficient memory size; more complex control and scheduling logic | High scalability in resource–constrained designs; moderate complexity |
| Semi-Parallel Transposed Four-Multiplier FIR Filter | Combines transposed architecture with semi–parallel multiplier sharing (four multipliers) to improve pipelining and resource sharing | Moderate; trade–off between multiplier count and pipelined structure | High throughput with improved pipeline performance | Merges benefits of transposed (speed) and semi–parallel (resource saving) approaches | Increased design complexity; potential scheduling latency issues | High scalability if well–designed; moderate-to-high complexity |
| Multi-Channel FIR Filter | Integrated design that interleaves multiple FIR filter channels using shared DSP resources and coefficient RAM | Optimized per channel; shared resources reduce per–channel cost | High overall throughput for multi–channel processing; individual channel rate may be lower | Excellent resource sharing across channels; highly scalable for multi–channel systems | Complex control logic for channel interleaving; potential increased per–channel latency | High design complexity; excellent scalability for large multi–channel applications |
| **FIR Filter Type** | **Architecture & Implementation** | **Resource Utilization** | **Performance** | **Pros** | **Cons** | **Additional Metrics (Complexity/Scalability)** |

“DSP: Designing for Optimal Results” – Xilinx Advanced Design Guide, 2005