Adder/Subtracter v11.0

DS214 April 24, 2009 **Product Specification**

Introduction

The Xilinx® LogiCORE™ IP Adder/Subtracter core provides LUT and single XtremeDSP™ slice add/sub implementations. The Adder/Subtracter module can create adders (A+B), subtracters (A–B), and dynamically configurable adder/subtracters which operate on signed or unsigned data. The function can be implemented in a single XtremeDSP slice or LUTs (but currently not a hybrid of both). The module can be pipelined.

Features

- Drop-in module for Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA, Spartan-3A/AN/3A DSP/XA FPGAs
- Backwards compatible with version 9.1
- Generates Adder, subtracter and Adder/Subtracter functions
- Supports twos complement-signed and unsigned operations
- Supports fabric implementation inputs ranging from 1 to 256 bits wide
- Supports XtremeDSP slice implementations with inputs ranging from 1 to 36 or 48 bits wide (varies with device family)
- Optional carry input and output
- Optional clock enable and synchronous clear
- Optional bypass (load) capability
- Option to set the B Value to a constant
- Optional pipelined operation
- For use with Xilinx CORE Generator™ , Xilinx AccelDSP™ Synthesis Tool, and Xilinx System Generator™ v11.1 or later

Pinout

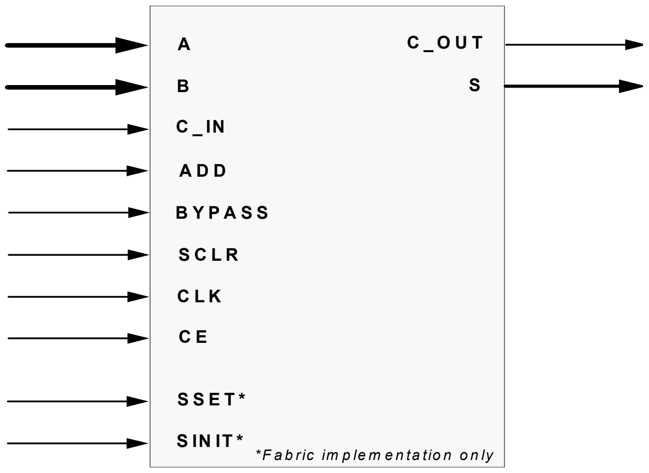

Figure 1: **Core Symbol**

Signal names for the core are shown in Figure 1 and described in Table 1. Note that Figure 1 shows the SSET and SINIT pins which appear only on fabric implementations. The XtremeDSP slice implementations do not support SSET and SINIT.

Table 1: **Core Signal Pinout**

Name	Direction	Description
A[N:0]	Input	A Input bus
B[M:0][1]	Input	B Input bus
ADD	Input	Controls the operation performed by an Adder/Subtracter (High = Addition, Low = Subtraction)
C_IN	Input	Carry Input
C_OUT	Output	Carry Output
S[P:0]	Output	Output bus
BYPASS	Input	Bypass control signal loads B port onto S port
CE	Input	Active high Clock Enable
CLK	Input	Clock signal: rising edge
SCLR	Input	Synchronous Clear: forces outputs to a low state when driven high
SINIT[2]	Input	Synchronous Initialization - forces outputs to a user defined state when driven high
SSET[2]	Input	Synchronous Set - forces outputs to a high state when driven high

1. B port will not be present if **Constant Input** = true and **Bypass** = false. A user-defined core-internal constant will be applied in place of the B operand.
2. Available only for **Implementation** = Fabric.

CORE Generator Graphical User Interface Parameters

The CORE generator GUI parameters for this module are described below:

- **Implement using**: Sets the implementation type to Fabric or DSP48. The default value is Fabric.
- **A Input Width**: Sets the width of the Port A input. The valid range is 2 to 256 for signed fabric, 1 to 255 for unsigned fabric, 2 to 48 for signed DSP48, and 1 to 47 for unsigned DSP48. The default value is 15. Note that mixed signed/unsigned input types can require a bit growth of 2 bits on the output in some cases. See Table 3.
- **A Input Type**: Sets the type of the Port A data to Signed or Unsigned. The default value is Signed.
- **B Input Width**: Sets the width of the Port B input. The valid range is 2 to 256 for signed fabric, 1 to 255 for unsigned fabric, 2 to 36 or 48 for signed DSP48, and 1 to 35 or 47 for unsigned DSP48. The B input is the concatenated A:B input of the DSP48 variant and its width is family dependent. (Virtex-4 DSP48 is limited to 36 bits max on the B input; all others can reach 48 bits max on the B input). The default value is 15. Note that mixed signed/unsigned input types can require a bit growth of 2 bits on the output in some cases. See Table 3.
- **B Input Type**: Sets the type of the Port B data to Signed or Unsigned. The default value is Signed.
- **Constant Input** and **Constant Value**: When **Constant Input** is true, Port B is set to the value that is specified with the parameter **Constant Value**. **Constant Value** must be entered in binary format and must not exceed **B Input Width**. In most cases specifying Port B to be a constant will create a module without Port B. The only exception to this is when bypass functionality is requested, as Port B is needed to provide the bypass data in this case. The default setting is for the Port B value to be provided via Port B.
- **Output Width** : Sets the output width. The default value is 16. The valid range varies depending on the settings of **A Input Width**, **A Input Type**, **B Input Width**, and **B Input Type,** as shown in Table 3. See "Output Widths" for more information about sufficiency and warning messages.

- **Add Mode**: Sets the mode of operation of the module. Valid values are Add, Subtract, and Add_Subtract. If an adder/subtracter is specified, the ADD pin sets the mode of operation. The default is Add.
- **Carry In**: When this parameter is set to true, a C_IN port will be created. This is an active-high, carry-in port for adders and a programmable (active-high/active-low with **Borrow In/Out Sense**) carry-in port for subtracters and adder_subtracters in subtract mode. The default value is false.
- **Carry Out**: When set to true, this parameter creates port C_OUT which is the synchronous active-high carry-out from the adder and adder/subtracter in add mode and the programmable (active-high/active-low with **Borrow In/Out Sense**) borrow-out from the subtracter or adder/subtracter in subtract mode. See Table 3 for information about when these outputs are permitted.
- **Bypass**: When set to true, creates a BYPASS pin. Activating the BYPASS pin sets the output to be the value given on Port B. This functionality is used for creating loadable counters and accumulators. The default is for no BYPASS pin to be generated. The default value is false.
- **Bypass and Clock Enable (CE) Priority**: This parameter controls whether or not the BYPASS input is qualified by **Clock Enable**. When set to Bypass_Overrides_CE, the activation of the BYPASS signal also enables the register. When set to CE_Overrides_Bypass, the register must have CE active to load the B port data. By default this parameter is set to Bypass_Overrides_CE.
- **Bypass Sense**: When set to Active_Low, the BYPASS pin is active low. BYPASS has a parameter to control its active sense because an historical implementation made significant speed gains with an active-low BYPASS, instead of active-high BYPASS. This is no longer necessarily the case, as sometimes active-high is as efficient, or more so. The details depend on the exact set of parameters. The default value is Active-High.
- **Borrow In/Out Sense**: When set to Active_Low, the C_IN and C_OUT pins are active low for subtraction. This conforms to the legacy fabric implementation where this was more optimal. The default value is Active_Low. By setting **Borrow In/Out Sense** to Active_High, an active high C_IN and C_OUT on subtraction is obtained.
- **Clock Enable**: When set to true, the module is generated with a clock enable input. The default setting is true.
- **Power on Reset Init Value**: Specifies in hex the value the S register will initialize to during power-up reset. Valid values are 0 up to $2^{\textbf{Output Width}}$ - 1. The default value is 0.
- **Synchronous Clear**: Specifies if an SCLR pin is to be included. Valid values are true or false; default is false.
- **Synchronous Set**: Specifies if an SSET pin is to be included. SSET pin is not valid in DSP48 implementations. See **Synchronous Set and Clear (Reset) Priority** for SCLR/SSET priorities. Valid values are true or false; default is false.
- **Synchronous Init**: Specifies if an SINIT pin is to be included which, when asserted, will synchronously set the S value to the value defined by **Init Value**. Note that if SINIT is present, then neither SSET nor SCLR may be present. Valid values are true or false, default is false. SINIT pin is not valid in DSP48 implementations.
- **Init Value**: Specifies in hex the value the S register will initialize to when SINIT is asserted. Valid values are 0 up to $2^{\textbf{Output Width}}$ - 1. Ignored if **Synchronous Init**= false. The default value is 0.
- **Synchronous Controls and Clock Enable (CE) Priority**: This parameter controls whether or not the SCLR (and if fabric: SSET and SINIT) inputs are qualified by CE. When set to Sync_Overrides_CE, the synchronous controls override the CE signal. When set to CE_Overrides_Sync, the control signals have an effect only when CE is high. The parameters are ignored unless **Clock Enable** is true. Note that on the fabric primitives, the SCLR and SSET controls override CE, so choosing CE_Overrides_Sync will generally result in extra logic. The default is Sync_Overrides_CE.

- **Sync Set and Clear (Reset) Priority**: Controls the relative priority of SCLR and SSET. When set to Reset_Overrides_Set, SCLR overrides SSET. The default is Reset_Overrides_Set, as this is the way the primitives are arranged. Making SSET take priority requires extra logic.
- **Latency Configuration**: Automatic or Manual; Automatic sets optimal latency for maximum speed; Manual allows user to set Latency to one of the allowed values. The default value is Manual.
- **Latency** : Value used for latency when **Latency Configuration** is set to Manual: 0 to **Output Width** (fabric); 0, 1, 2 (DSP48) See the section, "Pipelined Operation" for more information. The default value is 1.

Table 2 is a cross-reference table from the GUI parameters listed above to the XCO parameter names in the XCO file. The ranges and default values are also shown to facilitate data retrieval from the text above.

Table 2: **CORE Generator GUI and XCO Parameters**

GUI Name	Default Value	Valid Range	XCO Parameter
Component Name	AddSub	..	Component_Name
Implement using	Fabric	..	Implementation
A Input Type	Signed	..	A_Type
B Input Type	Signed	..	B_Type
A Input Width[1]	15	1..256 (fabric) 1..36 or 48 (DSP48)	A_Width
B Input Width[1]	15	1..256 (fabric) 1..36 or 48 (DSP48)	B_Width
Output Width	16	see Table 3	Out_Width
Latency Configuration	Manual	Manual, Automatic (Automatic sets optimal Latency for max speed)	Latency_Configuration
Latency	1	0..min(**Output Width**,64) (fabric) 0, 1, 2 (DSP48)	Latency
Clock Enable	true	..	CE
Synchronous Clear	false	..	SCLR
Synchronous Set	false	..	SSET
Synchronous Init	false	..	SINIT
Bypass	false	..	Bypass
Bypass Sense	Active_High	..	Bypass_Sense
Carry In	false	..	C_In
Carry Out	false	..	C_Out
Borrow In/Out Sense	Active_Low	..	Borrow_Sense
Add Mode	Add	Add, Subtract, Add_Subtract	Add_Mode
Constant Input	false	..	B_Constant
Constant Value[2]	1	..	B_Value

Table 2: **CORE Generator GUI and XCO Parameters *(Cont'd)***

GUI Name	Default Value	Valid Range	XCO Parameter
Synchronous Set and Clear (Reset) Priority	Reset_Overrides_Set	..	Sync_Ctrl_Priority
Synchronous Controls and Clock Enable (CE) Priority	Sync_Overrides_CE	..	Sync_CE_Priority
Bypass and Clock Enable (CE) Priority	CE_Overrides_Bypass	..	Bypass_CE_Priority
Power on Reset Init Value	0	..	AINIT_Value
Init Value	0	..	SINIT_Value

1. See details above for exact ranges allowed for signed and unsigned data types.
2. See details above for an explanation of the allowed range on Constant Value.

Core Use through CORE Generator

The CORE Generator GUI performs error-checking on all input parameters. Resource estimation and latency information are also available.

Several files are produced when a core is generated, and customized instantiation templates for Verilog and VHDL design flows are provided in the `.veo` and `.vho` files, respectively. For detailed instructions, see the CORE Generator software documentation.

Simulation Models

The core has a number of options for simulation models:

- VHDL behavioral model in the xilinxcorelib library
- VHDL unisim structural model
- Verilog unisim structural model

Xilinx recommends that simulations utilizing unisim-based structural models are run using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution to work properly in either functional or timing simulation. The unisim-based structural models might produce incorrect results if simulation with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in *Synthesis and Simulation Design Guide* for more information. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

Core Use through System Generator

The Adder/Subtracter core is available through Xilinx System Generator, a DSP design tool that enables the use of The Mathworks® model-based design environment Simulink for FPGA design. The Adder/Subtracter core is one of the DSP building blocks provided in the Xilinx blockset for Simulink. The core can be found in the Xilinx Blockset in the Math section. The block is called "AddSub". See the System Generator User Manual for more information.

Core Use through AccelDSP

The Adder/Subtracter core is available through Xilinx AccelDSP Synthesis Tool, a DSP synthesis tool that transforms a MATLAB® floating-point design into a hardware module that can be implemented in a Xilinx FPGA. The Adder/Subtracter core is invoked with a project option and automatically uses the core wherever the function is needed in the design. Certain parameters can be set through global directives. See the AccelDSP Synthesis Tool User's Guide or help menu for more information.

Migrating to Adder/Subtracter v11.0 from Earlier Versions

Updating from Adder/Subtracter v9.0 and later

The CORE Generator core update feature may be used to update an existing Adder/Subtracter XCO file to version 11.0 of the core. The core may then be regenerated to create a new netlist. See the CORE Generator documentation for more information on this feature.

Updating from versions prior to Adder/Subtracter v9.0

It is not currently possible to automatically update versions of the Adder/Subtracter core prior to v9.0. Xilinx recommends that customers use the Adder/Subtracter v11.0 GUI to customize a new core. Note that some features and configurations may be unavailable in Adder/Subtracter v11.0. Also, some port names may differ between versions.

Table 3: **Availability of Carry Outputs and Output Data Type/Size vs. Input Data Type**

Input types	Output Type	Conditions	Valid Output Widths	Carry Out
Both Unsigned	Unsigned	Add Mode =Add, Subtract	Q[1]	Available
			Q + 1	Not Available
		Add Mode =Add_Subtract	Q + 1	Available
One Unsigned, One Signed	Signed	Signed input wider	Q, Q + 1	Not Available
		Inputs equal width or unsigned input wider	Q + 1, Q + 2	Not Available
Both Signed	Signed	None	Q	Not Available
			Q + 1	Not Available

1. Q = Max(**A Input Width**, **B Input Width**).

Output Widths

Caution must be exercised when choosing output widths to accommodate input widths and sign types, as described in Table 3. The error trapping of inadequate output width is disabled to allow the System Generator tool to zero pad or sign-extend inputs and to allow input and output widths to be equal. Warnings are given when an inadequate output width is chosen but CORE generator *does not error*. It assumes that the appropriate padding has been added to allow for bit growth and only issues a warning.

Pipelined Operation

The Adder/Subtracter module can be optionally pipelined to improve speed. The pipelined operation is controlled by the latency parameters. Set **Latency Configuration** to Automatic to achieve optimal pipelining for maximum speed. Set Latency Configuration to Manual to allow a valid number of pipeline stages to be entered in the **Latency** parameter .

DSP48

For XtremeDSP slice (DSP48) implementations, the single DSP48 macro can be pipelined with 0, 1, or 2 stages of registers. **Latency Configuration =** Automatic will optimize the latency for speed. For Latency = 1, only output registers will be present. For Latency = 2, output and input registers will be present.

FABRIC

For Fabric implementations, pipelining is achieved by splitting the input buses into a number of bus-slices (that equals the number of pipelining stages), and doing as much work as possible on each bus-slice in the first stage; adding them together and storing the results and the carry-out of each. In the second stage, the carry-out from the least-significant slice is then fed into the next-higher result, which produces a carry-out that will be fed into the next result in the following stage, until the carry has propagated to the top.

Because less data needs to be stored, this is more efficient than the more intuitive technique which simply stores the inputs for each slice until the carry-in for that slice is generated. Additionally, the design is smaller and more easily routed.

After power up or reset, the pipelined module will take a number of clock cycles, specified by the latency control, for the outputs to become valid.

If bypass is requested on a pipelined module, the bypass value will appear on the outputs after the number of clock cycles specified by the latency control. Note that if both bypass and clock enable are requested, bypass priority must be set so that bypass does not override clock enable. For pipelined modules, the resource usage will be roughly **Latency** times bigger than the non-pipelined equivalent.

Notes:

Pipelining results in a significant increase in area usage in order to increase clock speed. If latency is required but area is more important than speed, add a SRL16-based shift register to the S output of this module for optimal area usage. See the RAM-based Shift Register core for this functionality.

Performance and Resource Utilization

Table 4 through Table 7 provide Addsub performance and resource usage for a number of different Adder/Subtracter configurations.

The maximum clock frequency results were obtained by double-registering input and output ports to reduce dependence on I/O placement. The inner level of registers used a separate clock signal to measure the path from the input registers to the first output register through the core.

The resource usage results do not include the above "characterization wrapper" registers and represent the true logic used by the core. LUT counts include SRL16s or SRL32s (according to device family).

The map options used were: “`map -pr b -ol high`.“

The par options used were: “`par -t 1 -ol high`.”

Clock frequency does not take clock jitter into account and should be derated by an amount appropriate to the clock source jitter specification.

The maximum achievable clock frequency and the resource counts may also be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors. The Xilinx Xplorer™ script can be used to find the optimal settings.

All characterization was done using the following parameter settings unless otherwise noted:

- **Add Mode** = 2
- **CE** = 1
- **Borrow In/Out Sense** = Active_Low

- **Latency Configuration** = Automatic
- all else = default

Table 4: **Fabric Addsub: Virtex-5 (Part = XC5VSX50T-1)**

Description	Small	Medium	Large	No Pipelining[1]	Bypass[2]	Carry Out[3]
A Input Width, **B Input Width**	8	32	100	32	32	32
Latency	1	3	9	0	3	3
Max Clock Frequency (MHz)	452	410	340	388	319	415
LUT6-FF pairs	8	93	393	32	97	96
LUTs	8	70	362	32	72	72
Flip-flops	0	91	384	0	94	94
DSP48Es	0	0	0	0	0	0

1. **Latency Configuration** = Manual and **Latency** = 0.
2. **Bypass** = true and **Bypass Sense** = Active_High.
3. **Carry Out** = true, **A Input Type** = Unsigned, and **B Input Type** = Unsigned.

Table 5: **XtremeDSP slice Addsub: Virtex-5 (Part = XC5VSX50T-1)**

Description	Small	Medium	Large	No Pipelining	Bypass[1]	Carry Out[2]
A Input Width, **B Input Width**	8	32	48	32	32	32
Latency	2	2	2	0	2	2
Max Clock Frequency (MHz)	450	450	450	241	450	450
LUT6-FF pairs	0	0	0	0	1	0
LUTs	0	0	0	0	1	0
Flip-flops	0	0	0	0	0	0
DSP48Es	1	1	1	1	1	1

1. **Bypass** = true and **Bypass Sense** = Active_High.
2. **Carry Out** = true, **A Input Type** = Unsigned, and **B Input Type** = Unsigned.

Table 6: **Fabric Addsub: Spartan-3A DSP (Part = XC3SD3400A-4)**

Description	Small	Medium	Large	No Pipelining[1]	Bypass[2]	Carry Out[3]
A Input Width, **B Input Width**	8	32	100	32	32	32
Latency	1	4	13	0	4	4
Max Clock Frequency (MHz)	251	230	197	188	191	230
LUTs	8	85	541	32	99	88
Flip-flops	0	105	601	0	108	108
DSP48As	0	0	0	0	0	0

1. **Latency Configuration** = Manual and **Latency** = 0.
2. **Bypass** = true and **Bypass Sense** = Active_High.
3. **Carry Out** = true, **A Input Type** = Unsigned, and **B Input Type** = Unsigned.

Table 7: **XtremeDSP slice Addsub: Spartan-3A DSP (Part = XC3SD3400A-4)**

Description	Small	Medium	Large	No Pipelining	Bypass[1]	Carry Out[2]
A Input Width, **B Input Width**	8	32	48	32	32	32
Latency	2	2	2	0	2	2
Max Clock Frequency (MHz)	251	251	251	86	251	251
LUTs	1	1	1	1	2	1
Flip-flops	0	0	0	0	0	0
DSP48As	1	1	1	1	1	1

1. **Bypass** = true and **Bypass Sense** = Active_High.
2. **Carry Out** = true, **A Input Type** = Unsigned, and **B Input Type** = Unsigned.

Support

Xilinx provides technical support for this LogiCORE™ product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator v11.1 and later. The Xilinx CORE Generator system is shipped with Xilinx ISE Foundation™ Series Development software.

To order Xilinx software, contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

Date	Version	Description of Revisions
4/28/2005	8.0	Updated to ISE Tools 7.1
4/24/2009	11.0	Updated to ISE Tools 11.1; added XtremeDSP slice implementations and support for Virtex-6 and Spartan-6 devices

Notice of Disclaimer