Floating Point Add/Sub

Floating point adder subtractor that support single precision/double precision and custom precision floating point in binary format

1. Features

Fully pipelined architecture with adjustable latency

Single precision, double precision and custom non-standard precision

Compliant with IEEE754 binary format with the following exceptions:

- Subnormal numbers are not supported and will be rounded to be zero before calculation is carried out. Subnormal results are rounded to zero.
- Only round-to-neatest even is supported

2. Interface

Parameter	IO	Required	Description and *Default* value
pTechnology		Y	Possible values: ***"ALTERA"***
pFamily		Y	Possible values: ***"CYCLONE"*** "ARRIA II GX"
pPrecision		Y	Possible Values 0: Custom precision, widths of mantissa and exponent are defined by pWidthMan and pWidthExp ***1: Single Precision*** 2: Double Precision
pWidthExp		Y	Width of Exponent for custom Precision, do not assign if pPrecision = 1 or 2
pWidthMan		Y	Width of Mantissa for custom Precision, do not assign if pPrecision = 1 or 2
pPipelineBarrelShifter		Y	Latency of the denormalizer shifter Default = 1
pPipelineNormalizer		Y	Latency of the normalizer shifter Default = 1 Total Latency of the Adder is pPipelineBarrelShifter+ pPipelinNormalizer+5

Name	Size	Required	Description
iv_InputA	I[S-1:0]	Y	Input A following IEEE754 binary format of single or double precision S = pWidthMan+pWidthExp+1
iv_InputB	I[S-1:0]	Y	Input B following IEEE754 binary format of single or double precision
i_SubNotAdd	I	Y	Operation 1: Subtraction 0: Addition
i_Dv	I	Y	Data valid input
o4_InputID	O[3:0]	N	ID of the input, this is to track input and output values When i_Dv is asserted, a non-zero ID is returned in the same cycle at o4_InputID. When i_Dv is deasserted, o4_InputID = 0
o4_OutputID	O[3:0]	N	ID of the output, non-zero values indicate valid data
ov_Result	O[S-1:0]	Y	Floating point output Result
o_Overflow	O	N	Overflow flag
o_NAN	O	N	Not a number flag
o_Underflow	O	N	Underflow flag
o_PINF	O	N	Positive infinity
o_NINF	O	N	Negative Infinity
i_Clk	I	Y	Clock input
i_ClkEn	I	Y	Clock enable input, deassert this signal to shutdown the whole engine
i_Arst	I	Y	Async active high reset

3. Performance

Table Area and Performance for Altera Devices

Device	Setting	Post-fit LEs/LUT LAB/ALUT	Register	DSP block	Post Fit Max Frequency (MHz)
Cyclone III	Single Precision Latency = 7 SpeedGradeC8	766/668	428	6 x 9-bit	169
Cyclone IV GX	Single Precision Latency = 7 SpeedGradeC8	765/665	428	6 x 9-bit	165
Cyclone V GX	Single Precision Latency = 7 SpeedGradeC8	538	429	1 x 27-bit	197
Arria II GX	Single Precision Latency = 7 SpeedGradeC6	538	452	4 x 18-bit	186
	Single Precision Latency = 8 SpeedGradeC8	538	494	4 x 18-bit	240
Cyclone III	Double Precision Latency = 10 SpeedGradeC8	1876/1581	947	12 x 9-bit	128
Cyclone IV GX	Double Precision Latency = 10 SpeedGradeC8	1891/1581	947	12 x 9-bit	130
Cyclone V GX	Double Precision Latency = 10 SpeedGradeC8	1310	1184	3 x 27-bit	183
Arria V GX[1]	Double Precision Latency = 10 SpeedGradeC4	1314	1188	3 x 27 bit	259
Note	Quartus II 12.0sp2 Webpack Build 263 8/2/2012 (1) Quartus II 12.0sp2 SE Build 263 8/2/2012				

4. Verification

This core has been verified against Intel CPU's result in real-time with 10 million random numbers. The test cases includes but not limits to the following:

Case	Op	Input A	Input B	Result	Flag
Substraction results in Subnormal Number	-	32'h01000000	32'h00800001	Subnormal Zero	Underflow
Substraction results in Subnormal Number	-	32'h01000000	32'h00BFFFF	Underflow Zero	Underflow
Addition results in Infinity	+	32'h7F7FFFFF	32'h7F000001	32'h7F80000	Overflow
Addition results in Infinity	+	32'h7F7FFFFF	32'h7F7FFFFF	32'h7F80000	Overflow
Substraction results in Infinity	+	32'hFF7FFFFF	32'hFF000001	32'hFF80000	Overflow
Substraction results in Infinity	-	32'hFF7FFFFF	32'h7F7FFFFF	32'hFF80000	Overflow

Verification algorithm:

```
for(T=0;T<10000000;T++)
{
    *(unsigned *)(&valA)=rand();
    *(unsigned *)(&valB)=rand();
    //Check The exponent part is not zero
    if(((*(unsigned*)&valA)&0x7F800000)==0) {T--;continue;}
    if(((*(unsigned*)&valB)&0x7F800000)==0) {T--;continue;}
    if(((*(unsigned*)&valA)&0x7F800000)==0x7F800000) {T--;continue;}
    if(((*(unsigned*)&valB)&0x7F800000)==0x7F800000) {T--;continue;}

    *(TYPE*)virt_addr=valA;          //Sendvalue to FPGA
    *(TYPE*)(virt_addr+4)=valB;      //Sendvalue to FPGA
    wait(200);
    ref=valA+valB;
    res=*(TYPE*)(virt_addr+12);      //Read back value from FPGA
    stat=*(unsigned *)(virt_addr+16);
    if(ref!=res && stat==0)
        {
        printf("%1.10e + %1.10e = %1.10e, %1.10e,%1.10e,\n",valA,valB,ref,res,(ref-res));
        printf("%08X %08X %08X %08X\n",*(unsigned*)&valA,*(unsigned*)&valB,*(unsigned*)&ref,*(unsigned*)&res);
        }
    if((T&0xFFFF)==0) {printf(".");fflush(0);}
}
```

5. Revision History

Date	Author	Core's Revision	Description
15/09/12	JeffLieu	1.0	Initial release
23/09/12	JeffLieu	1.0	Add details for Data valid signal and Input ID/Output ID Add performance for Arria V GX device