In order to implement additions with DSPs we should use the RESOURCE directive and bind the addition to AddSub_DSP core.

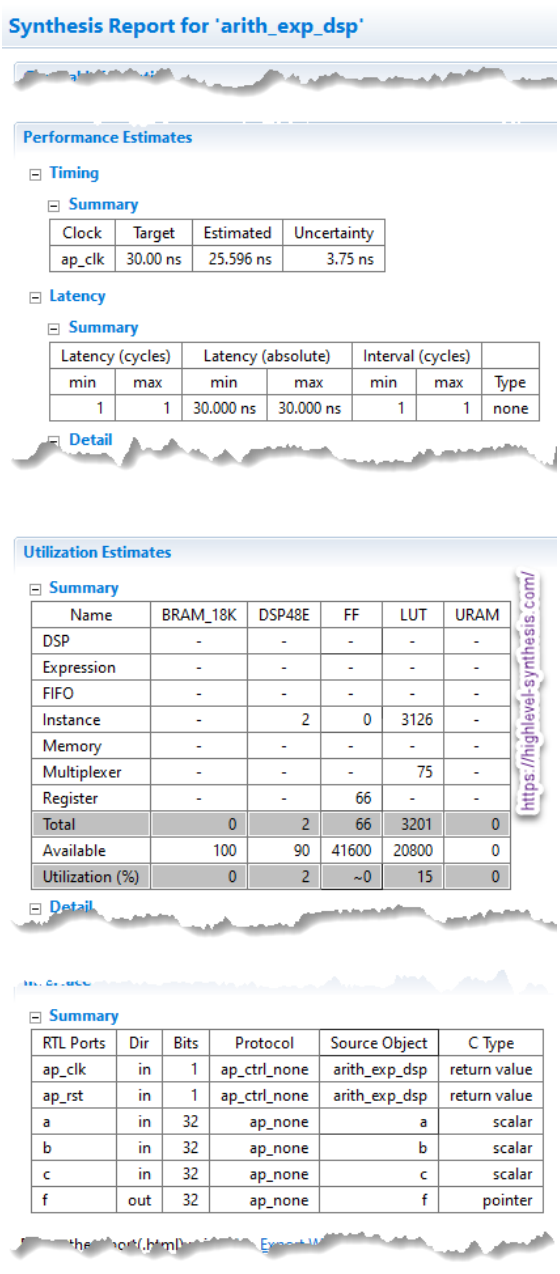
```
#pragma HLS RESOURCE variable=r core=AddSub_DSP
```

In this directive, we should explicitly mention the name of the target variable (here r). But some additions may be removed or added to the expression by the synthesis optimisation techniques.

To prevent this phenomenon, we can define a separate function for addition and disable the *inilining* feature. Then add the resource directive as shown in the code below. In this code, all additions are implemented by DSPs, and LUTs implement all multiplications.

```
int mult_func(int a, int b) {
#pragma HLS INLINE off
	int r;
#pragma HLS RESOURCE variable=r core=Mul_LUT
	r =  a*b;
	return r;
}

int add_func(int a, int b) {
#pragma HLS INLINE off
	int r;
#pragma HLS RESOURCE variable=r core=AddSub_DSP
	r =  a+b;
	return r;
}

void arith_exp_dsp(int a, int b, int c, int *f ) {

#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=a
#pragma HLS INTERFACE ap_none port=b
#pragma HLS INTERFACE ap_none port=c
#pragma HLS INTERFACE ap_none port=f

    int m1 = mult_func(a, b);
    int m2 = mult_func(a, c);
    int m3 = mult_func(c, b);
    int m4 = mult_func(m1, c);

    int a1 = add_func(m1, m2);
    int a2 = add_func(a1, m3);
    *f = add_func(a2, m4);

}
```

The synthesis report shown in Figure 1 shows that the resulted hardware utilises only 2 instances of DSPs.



Synthesis Report for 'arith_exp_dsp'

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	30.00 ns	25.596 ns	3.75 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	1	30.000 ns	30.000 ns	1	1	none

Detail

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	2	0	3126	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	75	-
Register	-	-	66	-	-
Total	0	2	66	3201	0
Available	100	90	41600	20800	0
Utilization (%)	0	2	~0	15	0

Detail

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	arith_exp_dsp	return value
ap_rst	in	1	ap_ctrl_none	arith_exp_dsp	return value
a	in	32	ap_none	a	scalar
b	in	32	ap_none	b	scalar
c	in	32	ap_none	c	scalar
f	out	32	ap_none	f	pointer

Figure 1

To make sure that DSPs implement all additions, you can have a look at the analysis perspective, as shown in Figure 2. As can be seen, the addition operator (labelled with 1) inside the corresponding function is implemented by the AddSub_DSP code labelled by 2.

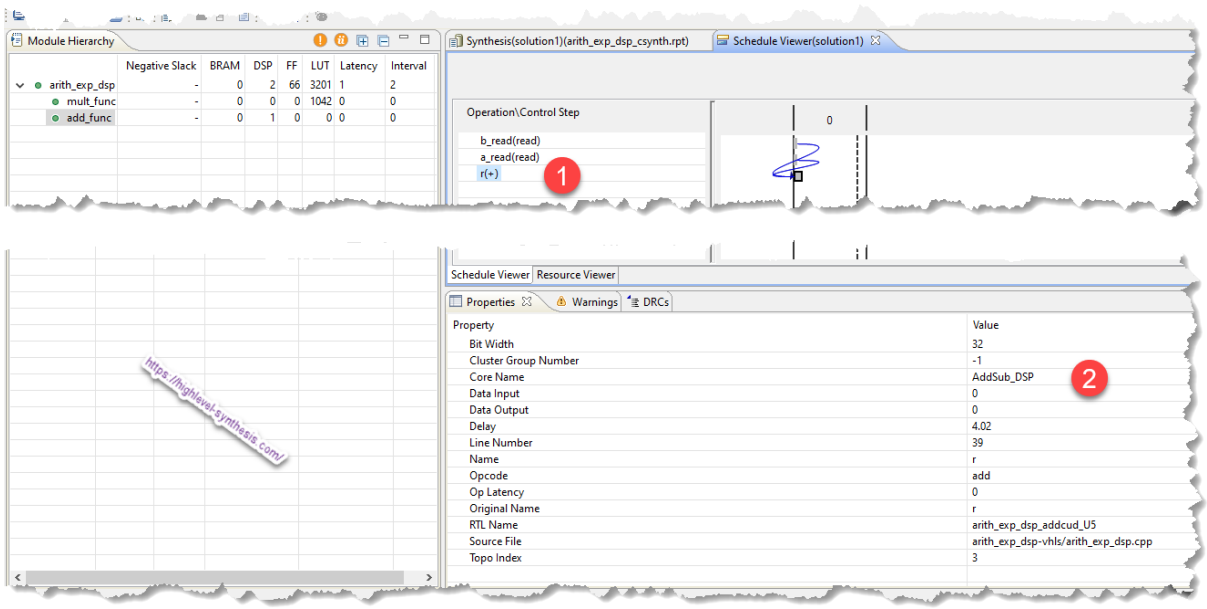

Module Hierarchy
Negative Slack BRAM DSP FF LUT Latency Interval
arith_exp_dsp - 0 2 66 3201 1 2
mult_func - 0 0 0 1042 0 0
add_func - 0 1 0 0 0 0
Synthesis(solution1)(arith_exp_dsp_csynth.rpt)
Schedule Viewer(solution1)
Operation\Control Step
b_read(read)
a_read(read)
r(+)
1
0
https://highlevel-synthesis.com/
Schedule Viewer
Resource Viewer
Properties
Warnings
DRCs
Property
Value
Bit Width 32
Cluster Group Number -1
Core Name AddSub_DSP
Data Input 0
Data Output 0
Delay 4.02
Line Number 39
Name r
Opcode add
Op Latency 0
Original Name r
RTL Name arith_exp_dsp_addcud_U5
Source File arith_exp_dsp-vhls/arith_exp_dsp.cpp
Topo Index 3
2

Figure 2