The code has defined a 32-bit static variable at line 5 that will be synthesised into a 32-bit register. So we expet to see 32 flip-flip used in the synthesis report.

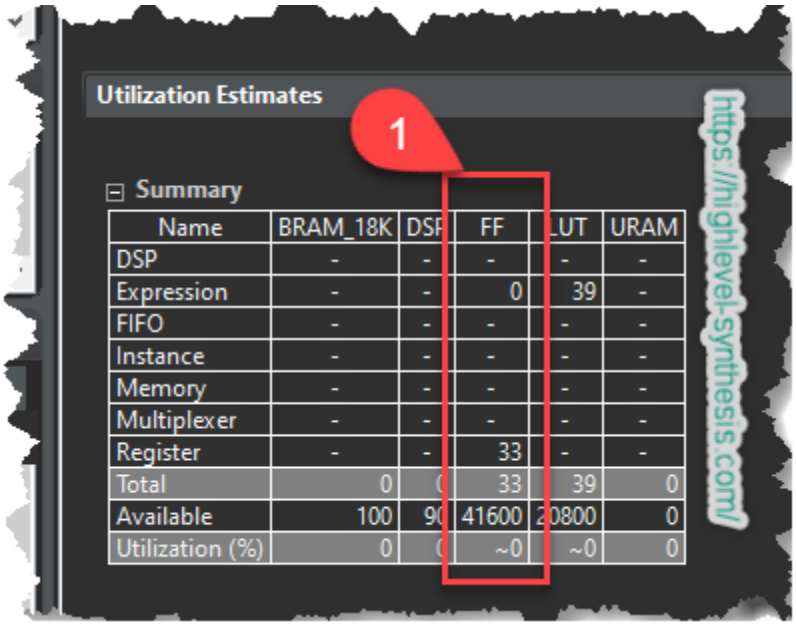
```
1- void quiz_2(unsigned int a, unsigned int &b) {
2- #pragma HLS INTERFACE ap_none port=a
3- #pragma HLS INTERFACE ap_none port=b
4- #pragma HLS INTERFACE ap_ctrl_none port=return

5-     static unsigned int s = 0;

6-     s = s + a;

7-     b = s;
8- }
```

If you synthesise the code then this would be the report

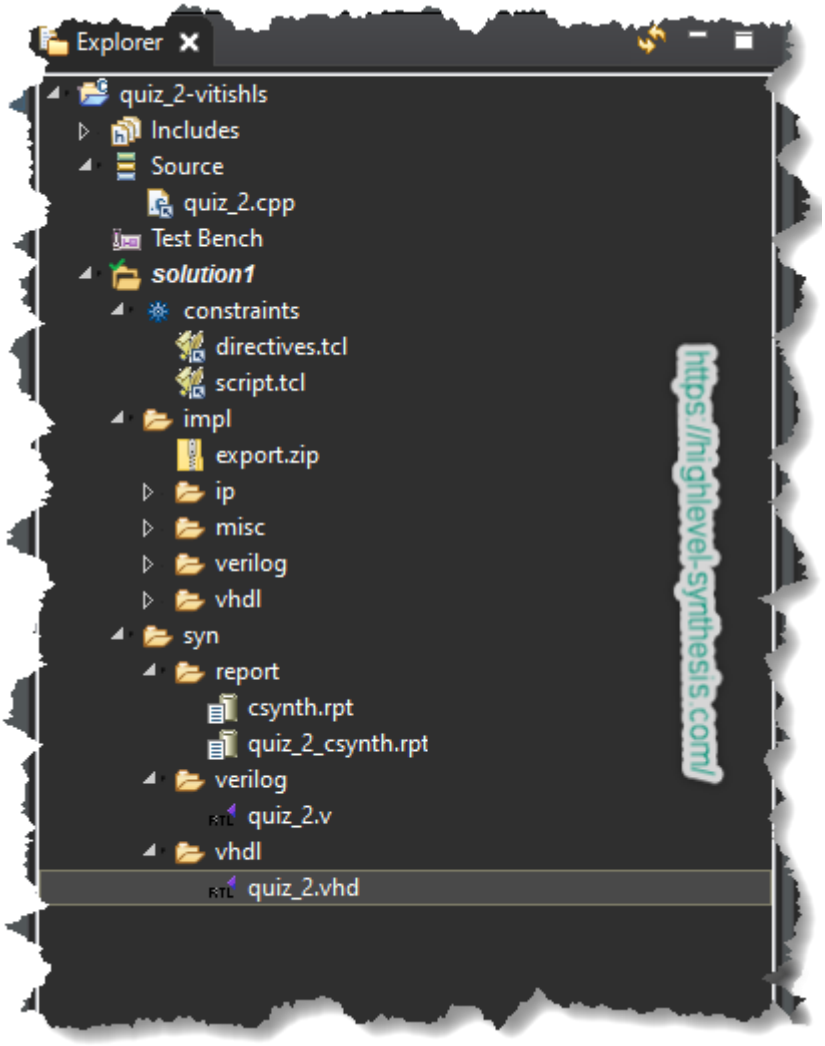


Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	39	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	33	-	-
Total	0	0	33	39	0
Available	100	90	41600	20800	0
Utilization (%)	0	0	~0	~0	0

As can be seen, the RTL design uses 33 flip-flops which is one more than what we expected.

Let's have a look at the RTL HDL source fil.

For this reason, please open the file at Solution1→syn→vhdl→quiz_2.vhd



There are three processes in the code.
In addition to the 32-bit s register, the processes use another flip-flop called ap_CS_fsm to implement the third process state machine.

```
...
ap_CS_fsm_assign_proc : process(ap_clk)
begin
    if (ap_clk'event and ap_clk =  '1') then
        if (ap_rst = '1') then
            ap_CS_fsm <= ap_ST_fsm_state1;
        else
            ap_CS_fsm <= ap_NS_fsm;
        end if;
    end if;
end process;

process (ap_clk)
begin
    if (ap_clk'event and ap_clk = '1') then
        if ((ap_const_logic_1 = ap_CS_fsm_state1)) then
```

```
                s <= add_ln10_fu_43_p2;
            end if;
        end if;
    end process;

    ap_NS_fsm_assign_proc : process (ap_CS_fsm)
    begin
        case ap_CS_fsm is
            when ap_ST_fsm_state1 =>
                ap_NS_fsm <= ap_ST_fsm_state1;
            when others =>
                ap_NS_fsm <= "X";
        end case;
    end process;
    add_ln10_fu_43_p2 <= std_logic_vector(unsigned(s) + unsigned(a));
    ap_CS_fsm_state1 <= ap_CS_fsm(0);
    b <= std_logic_vector(unsigned(s) + unsigned(a));
end behav;
```

Takeaway message:

The HLS synthesis tool may add some registers and flip-flops to the HLS description to define state machines implementing the design behaviour.

In addition, the synthesis tool can optimise the code and remove some registers defined in the HLS description.