

Exercise 1

Bit-level access functions and overloaded operators can be used for solving this problem. The following code shows such an implementation.

```
ap_uint<6> extract_opcode(ap_uint<32> instruction) {
    return instruction(31, 26);
}

ap_uint<5> extract_rs(ap_uint<32> instruction) {
    return instruction(25, 21);
}

ap_int<16> extract_imm(ap_uint<32> instruction) {
    return instruction(15, 0);
}

void datatype_exercise_01(
    ap_uint<32> instruction,
    ap_uint<6> *opcode,
    ap_uint<5> *rs,
    ap_uint<5> *rd,
    ap_int<16> *immediate
) {
    #pragma HLS INTERFACE ap_ctrl_none port=return
    #pragma HLS INTERFACE ap_none port=instruction
    #pragma HLS INTERFACE ap_none port=opcode
    #pragma HLS INTERFACE ap_none port=rs
    #pragma HLS INTERFACE ap_none port=rd
    #pragma HLS INTERFACE ap_none port=immediate

    *opcode = extract_opcode(instruction);
    *rs = extract_rs(instruction);
    *immediate = extract_imm(instruction);
}
```

2 Digital System Design with High-Level Synthesis in FPGA

This picture shows parts of the synthesis report. As the resulted circuit is just a cope of wires, the propagation delay is 0 and there is no resource usage.

Performance Estimates

1

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	0 ns	1.25 ns

Latency

Utilization Estimates

2

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	-	-	-
Total	0	0	0	0	0
Available	100	90	41600	20800	0
Utilization (%)	0	0	0	0	0

Detail

Interface

3

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
instruction_V	in	32	ap_none	instruction_V	scalar
opcode_V	out	6	ap_none	opcode_V	pointer
rs_V	out	5	ap_none	rs_V	pointer
rd_V	in	5	ap_none	rd_V	pointer
immediate_V	out	16	ap_none	immediate_V	pointer

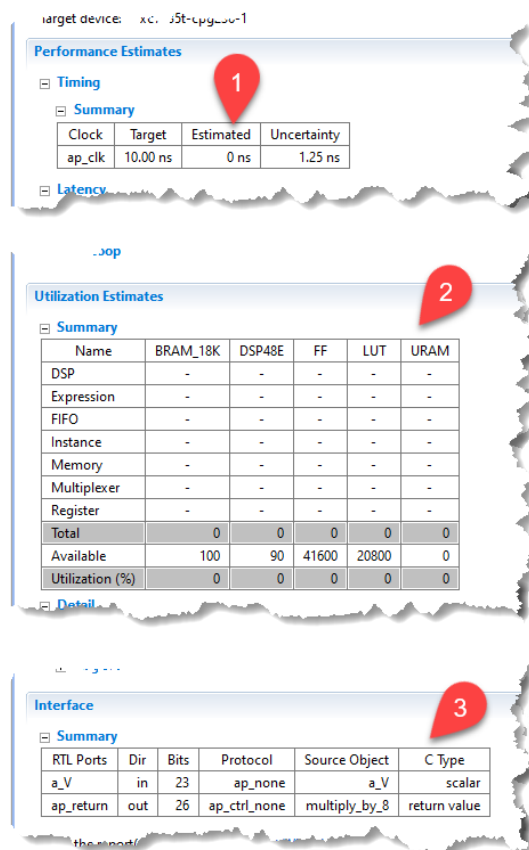
Exercise 2

The result of a 23-bit multiplication by 8 may have up to 26 bits. If we don't want to lose any information, the function should return a 26-bit result. The following code can do that

```
1- ap_uint<26> multiply_by_8(ap_uint<23> a) {
2- #pragma HLS INTERFACE ap_ctrl_none port=return
3- #pragma HLS INTERFACE ap_none port=a
4-   ap_uint<26> b;
5-   b = (ap_uint<26>)a << 3;
6-   return b;
7- }
```

Note that, the way that code use the type cast at the fifth line. Without using this type cast, HLS will generate a 23-bit result and will lose the extra-bit for large 23-bit numbers.

This picture shows parts of the synthesis report. As the resulted circuit is just a cope of wires, the propagation delay is 0 and there is no resource usage.



Exercise 3

The following top-function will solve the problem

```
ap_uint<1> even_parity(ap_uint<14> a) {
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=a

    ap_uint<1> p = 0;
    for (int i = 0; i < 7; i++) {
#pragma HLS UNROLL
        p = p^a[2*i];
    }
    return p;
}
```

This picture shows parts of the synthesis report. The circuit propagation delay is about 1.956 ns, and the circuit only uses 12 LUTs.

