Digital System Design with High-Level Synthesis for FPGA

Note: All the HLS codes are attached to this lecture.

Exercise 1

The following code can implement this design

```
void odd_even(ap_uint<8> a, bool &led1, bool &led2) {

#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=a
#pragma HLS INTERFACE ap_none port=led1
#pragma HLS INTERFACE ap_none port=led2


	if (a[0] == 1) {
		led1 = 0;
		led2 = 1;
	} else {
		led1 = 1;
		led2 = 0;
	}

}
```

Exercise 2

The following code show the code to find the 1011 binary pattern in a 9-bit unsigned integer number.

```
bool find_pattern(ap_uint<9> a) {
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=a

	bool p;
	if (
		  ( a(8, 5) == 0b1011)
		|| ( a(7, 4) == 0b1011)
		|| ( a(6, 3) == 0b1011)
		|| ( a(5, 2) == 0b1011)
		|| ( a(4, 1) == 0b1011)
		|| ( a(3, 0) == 0b1011)
	) {
		p = 1;
	} else {
		p = 0;
	}

	return p;
}
```

Exercise 3

First, we can ave all the 4-bit binary codes in an array.

```
const ap_uint<4> gray_code[16] = {
       0000,
       0001,
       0011,
       0010,
       0110,
       0111,
       0101,
       0100,
       1100,
       1101,
       1111,
       1110,
       1010,
       1011,
       1001,
       1000
};
```

Then we can access each code using its corresponding index.

You may say that the following code will do the task

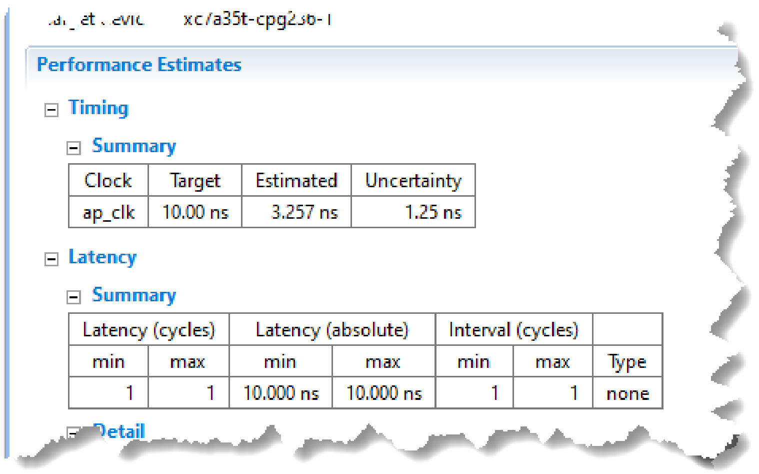
```
ap_uint<4> binary2gray(ap_uint<4> a) {
#pragma HLS INTERFACE ap_none port=a
#pragma HLS INTERFACE ap_ctrl_none port=return

       ap_uint<4> g;

       g = gray_code[a];


       return g;
}
```

This code is correct and can be synthesised by vivado HLS, but the resulted circuit is not combinational. Let’s have a look at the synthesis report.



xc7a35t-cpg236-1

Performance Estimates

⊟ **Timing**

⊟ **Summary**

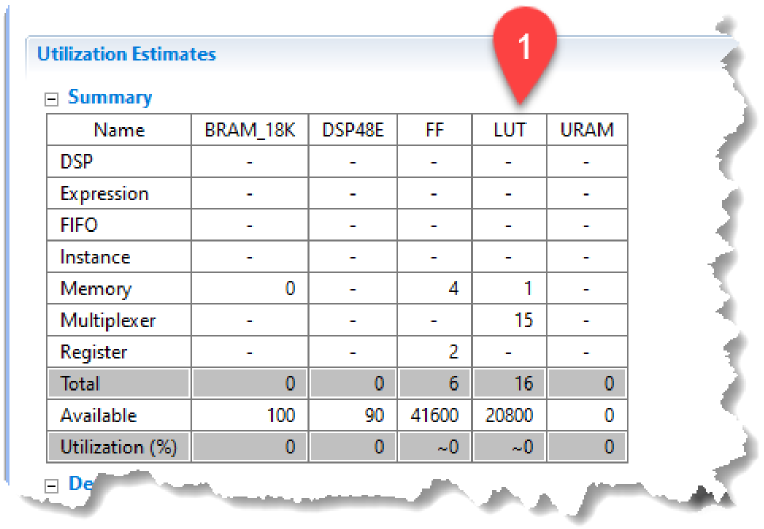
Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	3.257 ns	1.25 ns

⊟ **Latency**

⊟ **Summary**

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	1	10.000 ns	10.000 ns	1	1	none

⊟ **Detail**

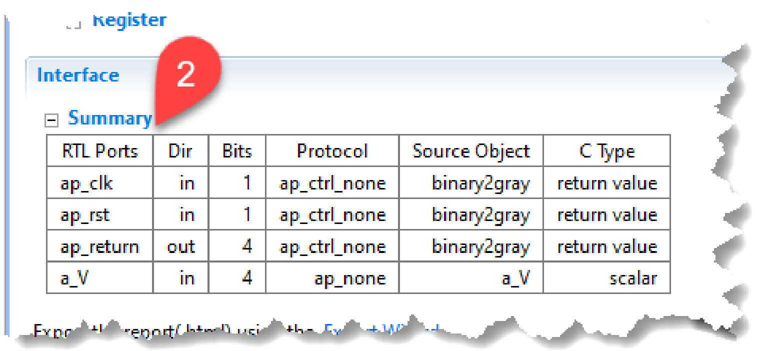


Utilization Estimates

⊟ **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	0	-	4	1	-
Multiplexer	-	-	-	15	-
Register	-	-	2	-	-
Total	0	0	6	16	0
Available	100	90	41600	20800	0
Utilization (%)	0	0	~0	~0	0

⊟ **De**



Register

Interface

⊟ **Summary**

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	binary2gray	return value
ap_rst	in	1	ap_ctrl_none	binary2gray	return value
ap_return	out	4	ap_ctrl_none	binary2gray	return value
a_V	in	4	ap_none	a_V	scalar

By using the switch-case statement, we can provide a combinational circuit description.

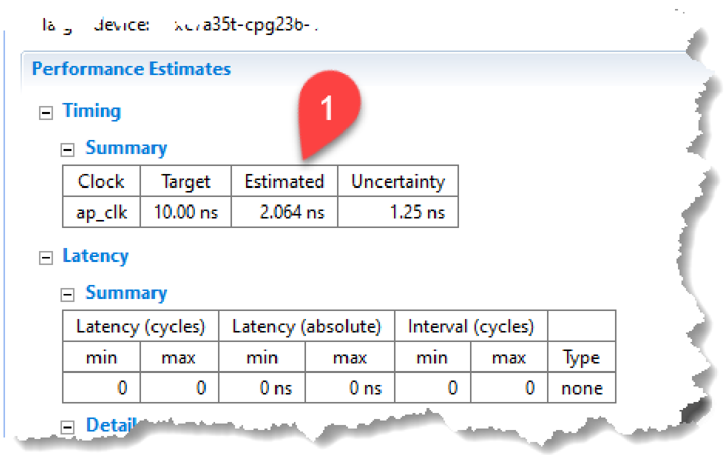
```
ap_uint<4> binary2gray(ap_uint<4> a) {
#pragma HLS INTERFACE ap_none port=a
#pragma HLS INTERFACE ap_ctrl_none port=return

	ap_uint<4> g;

	switch (a) {
	case 0:
		g = gray_code[0];
		break;
	case 1:
		g = gray_code[1];
		break;
	case 2:
		g = gray_code[2];
		break;
	case 3:
		g = gray_code[3];
		break;
	case 4:
		g = gray_code[4];
		break;
	case 5:
		g = gray_code[5];
		break;
	case 6:
		g = gray_code[6];
		break;
	case 7:
		g = gray_code[7];
		break;
	case 8:
		g = gray_code[8];
		break;
	case 9:
		g = gray_code[9];
		break;
	case 10:
		g = gray_code[10];
		break;
	case 11:
		g = gray_code[11];
		break;
	case 12:
		g = gray_code[12];
		break;
	case 13:
		g = gray_code[13];
		break;
	case 14:
```

```
                g = gray_code[14];
                break;
        case 15:
                g = gray_code[15];
                break;
        }
        return g;
}
```

The following figure shows the synthesis report.



Performance Estimates

Timing

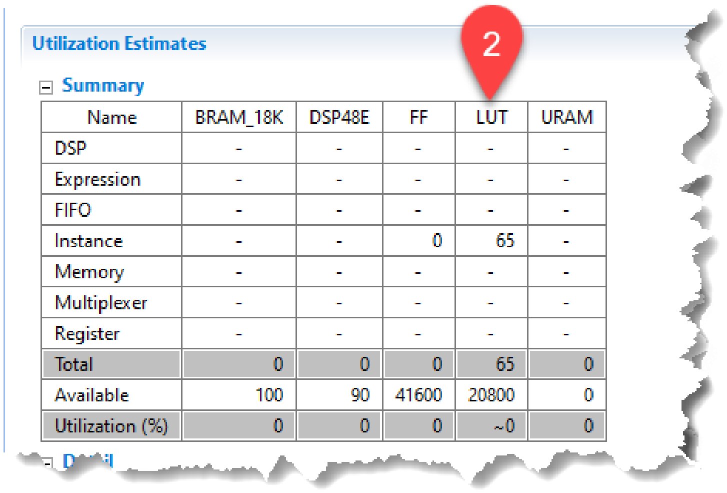
Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	2.064 ns	1.25 ns

Latency

Summary

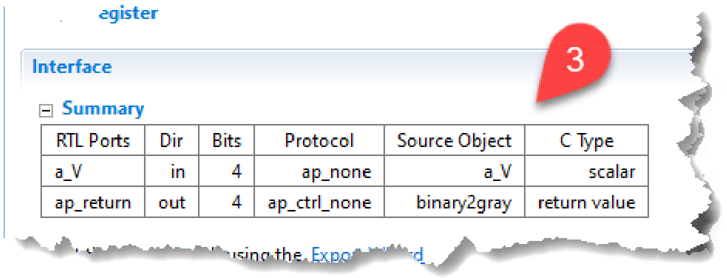
Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
0	0	0 ns	0 ns	0	0	none



Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	-	0	65	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	-	-	-
Total	0	0	0	65	0
Available	100	90	41600	20800	0
Utilization (%)	0	0	0	~0	0



Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
a_V	in	4	ap_none	a_V	scalar
ap_return	out	4	ap_ctrl_none	binary2gray	return value