AN13101

LPC804 PLU Config Tool Demo

Rev. 0 — December 21, 2020 Application Note

by: NXP Semiconductors

Contents

1 Introduction

LPC804 has a Programmable Logic Unit (PLU) module. The PLU is used to create small combinatorial and/or sequential logic circuits that operate completely independently from the Arm® Cortex® M0+ Core. The logic operation can be programmed using software, and NXP provides free tools to help designers easily implement circuit designs and automatically generate the required register settings.

PLU features are listed as below:

- Up to 6 inputs from I/O pads
- Up to 8 outputs to I/O pads
- Up to 26 Look Up Tables (LUTs)
- Up to 4 state Flip Flops (FFs)
- Clock input from external I/O pad
- Feedback to MCU via I/O pads or through switch matrix

1.1 Look Up Tables (LUTs)

The PLU module has 26 LUTs, which can implement any combinational function of up to five possible inputs.

The inputs include:

- Other LUT outputs
- State FF outputs
- All inputs available from PLU I/O pads

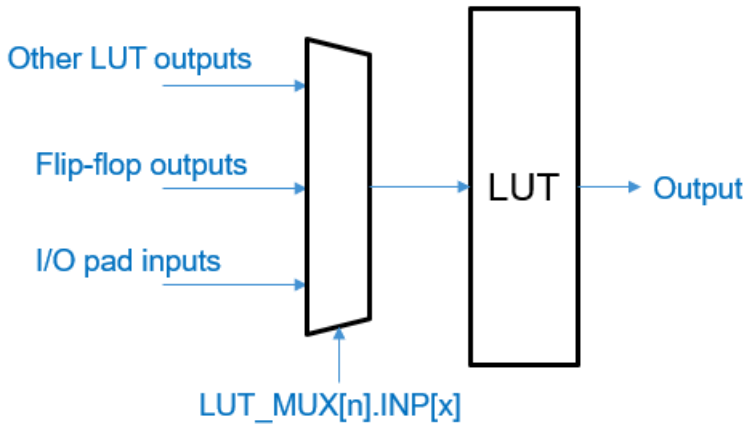

Figure 1. Look Up Tables (LUTs)

Each LUT is programmable using LPC804 registers which implements the logic mapping from 5 input lines to 1 output line with the **Logic Truth Value Table**. The output is immediately determined (allowing for propagation delays) by matching the input values of input lines with the entry in the table that matches the input values.

An example of Logic Truth Value Table for a 5-input customized LUT

IDX	IN4	IN3	IN2	IN1	IN0	OUT	IDX	IN4	IN3	IN2	IN1	IN0	OUT
0x0	0	0	0	0	0	0	0x10	1	0	0	0	0	1
0x1	0	0	0	0	1	0	0x11	1	0	0	0	1	1
0x2	0	0	0	1	0	0	0x12	1	0	0	1	0	1
0x3	0	0	0	1	1	0	0x13	1	0	0	1	1	1
0x4	0	0	1	0	0	0	0x14	1	0	1	0	0	1
0x5	0	0	1	0	1	0	0x15	1	0	1	0	1	1
0x6	0	0	1	1	0	0	0x16	1	0	1	1	0	1
0x7	0	0	1	1	1	0	0x17	1	0	1	1	1	1
0x8	0	1	0	0	0	1	0x18	1	1	0	0	0	1
0x9	0	1	0	0	1	1	0x19	1	1	0	0	1	0
0xA	0	1	0	1	0	1	0x1A	1	1	0	1	0	0
0xB	0	1	0	1	1	1	0x1B	1	1	0	1	1	0
0xC	0	1	1	0	0	1	0x1C	1	1	1	0	0	0
0xD	0	1	1	0	1	1	0x1D	1	1	1	0	1	0
0xE	0	1	1	1	0	1	0x1E	1	1	1	1	0	0
0xF	0	1	1	1	1	1	0x1F	1	1	1	1	1	0

Figure 2. An example of Logic Truth Value Table for a 5-input customized LUT

1.2 Multiplexer

Multiplexers in the PLU are used to set up the connections between units and select the right output signal.

Figure 3 shows the LUT input and output multiplexers for each channel.

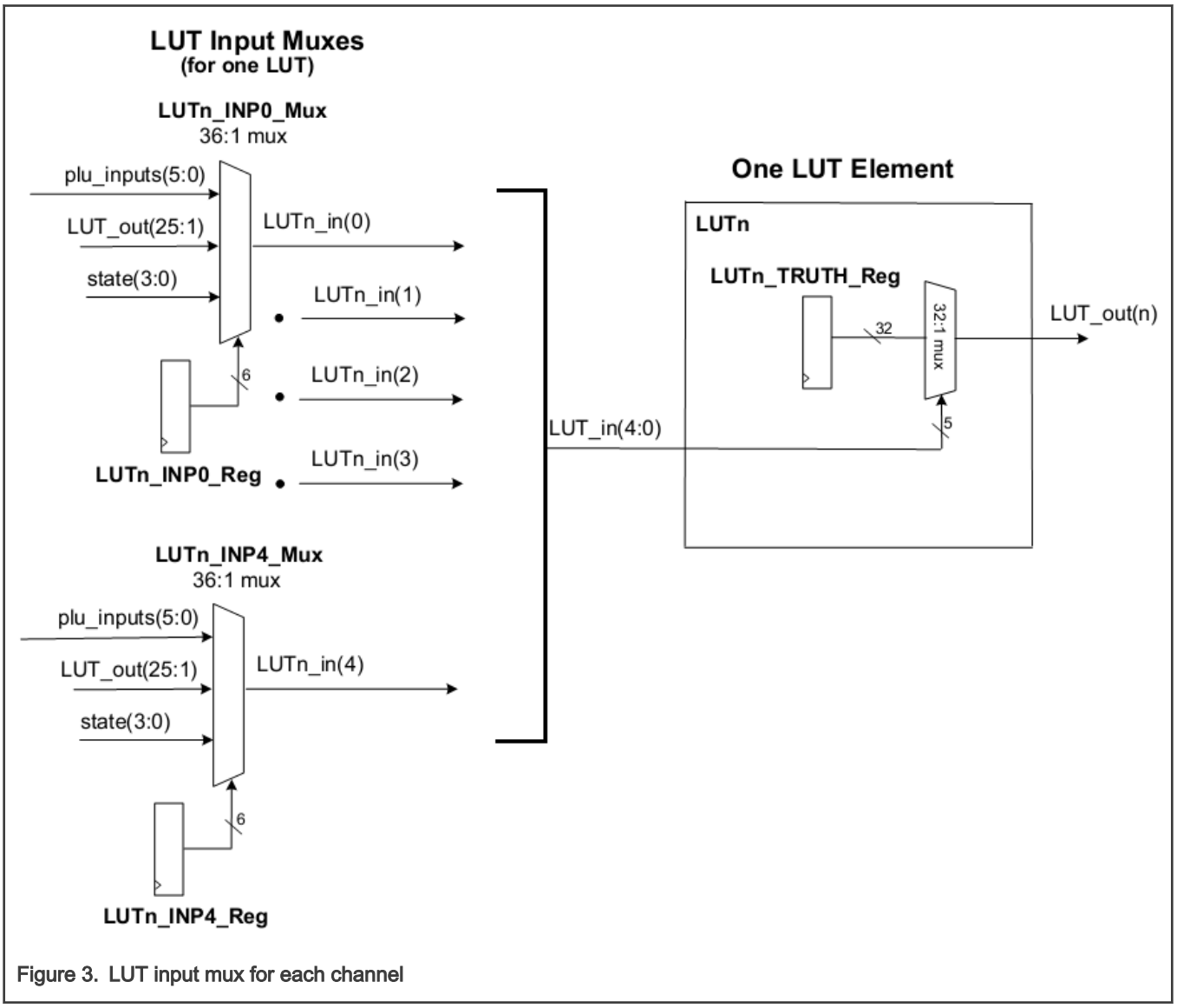

Figure 3. LUT input mux for each channel

Figure 4 shows the PLU output mux for each LUT.

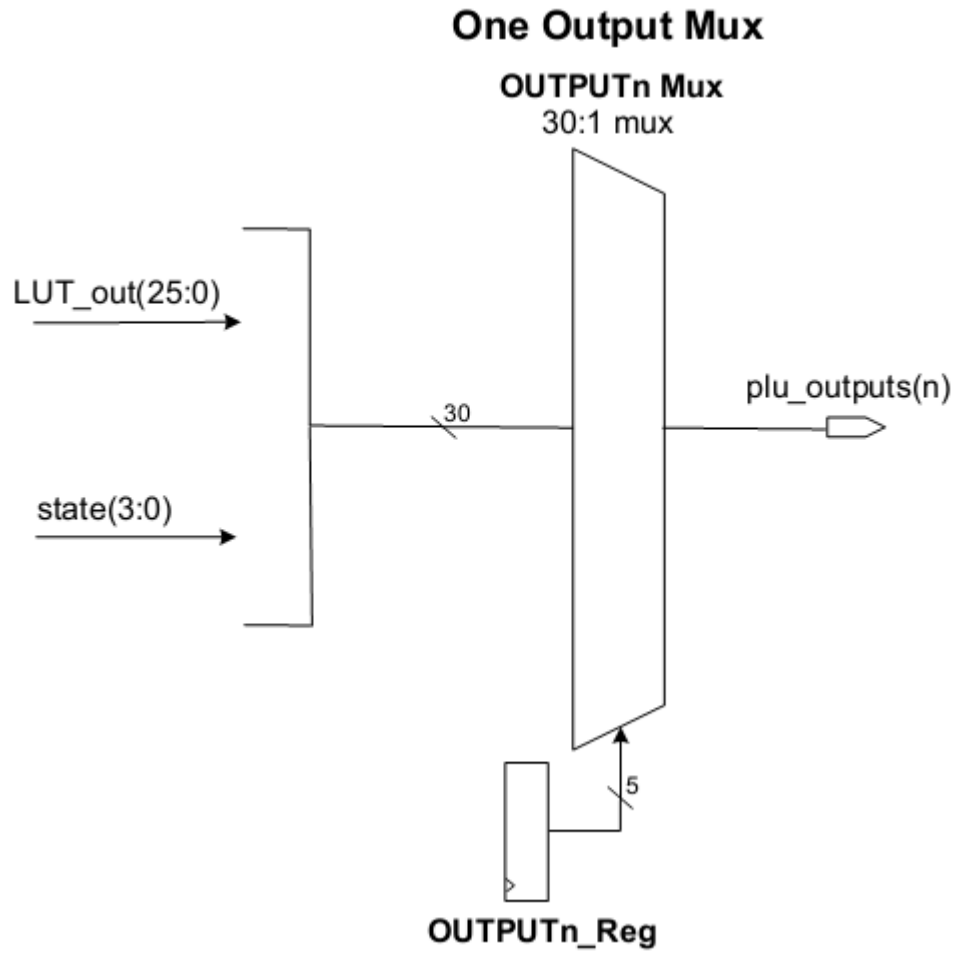

Figure 4. PLU output mux for each LUT

1.3 Steps for using the PLU

The PLU works as a general peripheral. Configure the PLU as below:

1. **Enable clocks**

   Enable PLU and SWM clock in `SYSCON_SYSAHBCLKCTRLx` register and toggle the **PLU Reset** bit in the `SYSCON_PRESETCTRLx` register.

2. **Assign pins**

   Assign PLU inputs and outputs to specific pins using `SWM_PINASSIGNFIXED0`. Some pins can be configured as both input and output of the PLU. For example, `PIO0_08` can be assigned to both `PLU_OUT1` and `PLU_INPUT0`.

3. **Create a logic network**

   Perform as below to implement a logic design with the PLU:

   - Considering the approach to be used, direct LUT implementation or using primitive logic gates (or a mix of both approaches)
   - Programming the customized LUTs, if primitive logic gates are not used
   - Programming the connection of PLU inputs and outputs to LUTs

To help developers create a logic design, NXP provides a PLU Config Tool to help user create logic networks easily. This tool can be downloaded for free from PLU configuration tool page. The following illustrates how to build a simple design with the PLU Design Tool. MCUXpresso Configuration tools also include support for setting up the PLU, assistant to pin connection and error checking.

2 Build user App. with PLU Config tool

PLU Config tool can run on Windows 8 or 10.

With the PLU Config tool, a user can:

- Create customized LUTs by direct, LUT-based and/or gate-level design.
- Automatically generate LUT configurations from a schematic design.
- Implement a design from Verilog.
- Generate C code to set up the PLU configuration registers automatically.

Users still need to connect PLU inputs, outputs, PLU `clk_in` to the physical pins via the switch matrix.

In this document, we build two example circuits to illustrate how to use the PLU Config tool, one using the direct LUT-based design and the other using a schematic design approach.

2.1 Required environment for demos

2.1.1 Hardware

- LPCXpresso804 evaluation board
- A USB cable
- Host computer

Board rework:

PLU input pins connection (using wires to connect):

- Input1 - CN3_7 (GPIO0_8) to CN8_2 (PLU_IN2, GPIO0_21).
- Input2 - CN3_6 (GPIO0_9) to CN3_10 (PLU_IN3, GPIO0_20).
- Input3 - CN5_1 (GPIO0_10) to CN8_5 (PLU_IN4, GPIO0_19).

PLU output pins connection (using wires to connect):

- Output1 - CN3_1 (PLU_OUT0, GPIO0_14) to CN8_4 (LED_RED, GPIO0_13).

2.1.2 Software

Examples built using MCUXpresso IDE v11.2.1 and SDK 2.8.0 are provided on NXP to complement this application note. Two sets of demo code are included:

- Using direct, LUT-based design to generate C code to set up PLU configuration register
- Using schematic design to generate C code to set up PLU configuration register

2.2 Direct, LUT-based design demo

This demo illustrates how to use LUT-based method in PLU config tool to create an example LUT-based design with no specific purpose, just for illustrative purpose, which has the logic needed between inputs and outputs as below.

2.2.1 Logic between inputs and outputs

For this demo, we set up the design for the relationship between inputs and outputs, as shown in Table 1.

Table 1. Logic between inputs and outputs for LUT-based design demo

Input3	Input2	Input1	Output1
PLU_IN4	PLU_IN3	PLU_IN2	PLU_OUT0
P0_19	P0_20	P0_21	P0_14 (Red LED, ON for **0**)
LUT_IN2	LUT_IN1	LUT_IN0	LUT_OUT0
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

2.2.2 PLU Config tool usage

1. Open the PLU Config tool and choose working flow for Direct, LUT-based design. Place Input ports 1, 2, and 3, then output1 and LUT0 in the schematic, draw the connections between inputs, LUT and output, as shown in Figure 5.

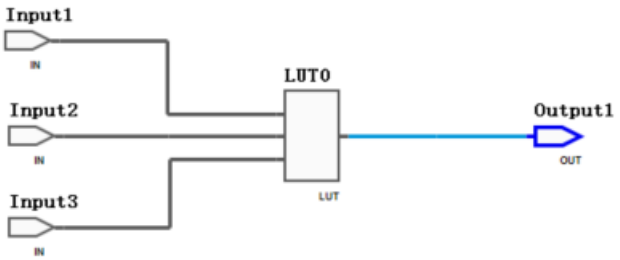

Figure 5. Setup connections between input, LUT and output

2. After finishing the connections, on the left of the window, there is a **mapping** setup window ready for setup. It specifies connections from the inputs and outputs to specific PLU resources. Map PLU inputs and output to the LUT by choosing the needed pins from the pull-down menu. The demo code uses `PLU_INPUT2` for input1, `PLU_INPUT3` for input2, `PLU_INPUT4` for input3, and `PLU_OUTPUT0` for output1. These relationships are shown in Table 1.

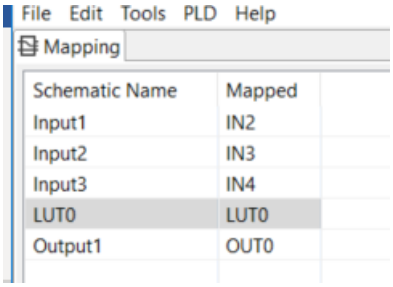

Figure 6. Map PLU input and output to LUTs

3. In the right panel of the Config Tool, there is a LUT setup form which can be used to set up the LUT values directly, according to Table 1. Click on the table output column to change the **0** or **1** value as desired.

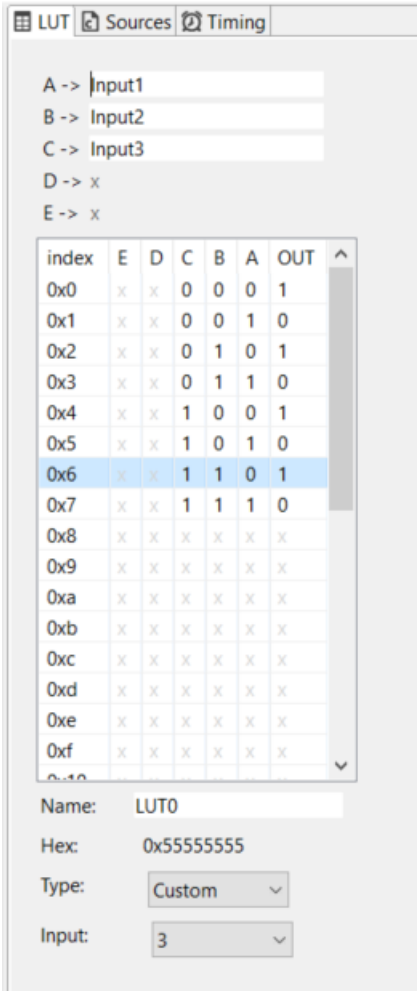

Figure 7. Set up the LUT value directly

4. After completing the LUT setup, press the **Sources** tab in this window. The tool generates the C code automatically and this code can be used to set up PLU configuration registers.

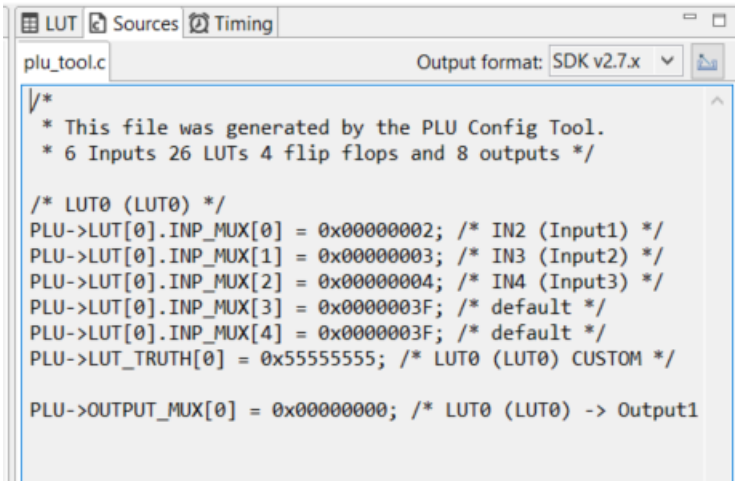

Figure 8. Generate the C code to set up PLU configuration register

5. Refer to the SDK PLU demo code in the example project, *lpcxpresso804_plu_AN_direct_LUT_design*. The example code has a value of `0x55555555` for `LUT_TRUTH[0]`. Because in this demo only three inputs are used, as shown in Table 1, the `LUT_OUT0` only uses eight values in the truth table, only the first eight bits are needed to configure the LUT, so the configuration value, `0x00000055`, could be used in the output code.

   In line30 of the *plu_combination.c* file, modify the configured value according to the generated C code. And in order to watch the Red LED only, configure LUT1 (Green) and LUT2 (Blue) to `0xFF`.

```
11 #include "fsl_debug_console.h"
12
13 #include "pin_mux.h"
14 /*******************************************************************************
15  * Definitions
16  ******************************************************************************/
17
18 /* PLU module */
19 #define DEMO_PLU_BASE         PLU
20 #define DEMO_PLU_LUT_IN_SRC_0 kPLU_LUT_IN_SRC_PLU_IN_2
21 #define DEMO_PLU_LUT_IN_SRC_1 kPLU_LUT_IN_SRC_PLU_IN_3
22 #define DEMO_PLU_LUT_IN_SRC_2 kPLU_LUT_IN_SRC_PLU_IN_4
23
24 #define DEMO_PLU_LUT_OUT_SRC_0     kPLU_OUT_SRC_LUT_0
25 #define DEMO_PLU_LUT_OUT_SRC_1     kPLU_OUT_SRC_LUT_1
26 #define DEMO_PLU_LUT_OUT_SRC_2     kPLU_OUT_SRC_LUT_2
27 //#define DEMO_PLU_LUT_0_TRUTH_TABLE 0x000004D /* 0b01001101 */
28 //#define DEMO_PLU_LUT_1_TRUTH_TABLE 0x000002B /* 0b00101011 */
29 //#define DEMO_PLU_LUT_2_TRUTH_TABLE 0x0000017 /* 0b00010111 */
30 #define DEMO_PLU_LUT_0_TRUTH_TABLE 0x0000055 /* 0b01010101  using for AN demo code for direct LUT design */
31 #define DEMO_PLU_LUT_1_TRUTH_TABLE 0x00000FF /* 0b11111111  Let only RED LED Lighting in the Demo*/
32 #define DEMO_PLU_LUT_2_TRUTH_TABLE 0x00000FF /* 0b11111111  Let only RED LED Lighting in the Demo*/
33
```

Figure 9. Modify the SDK PLU demo code according generated C code

2.2.3 Direct_LUT design demo result

1. Set up the board as described in Board rework.
2. Build and run the provided demo code, *lpcxpresso804_plu_AN_direct_LUT_design.zip*.
3. Input **0** or **1** for input 1, 2, 3 from the MCUXpresso IDE Console window, and watch the Red LED status. The results are shown in Table 2.

Table 2. Direct_LUT design demo result

Input3	Input2	Input1	Output1 - RED LED
0	0	0	OFF
0	0	1	ON
0	1	0	OFF
0	1	1	ON
1	0	0	OFF
1	0	1	ON
1	1	0	OFF
1	1	1	ON

2.3 Schematic design demo

The difference between LUT-based method and schematic method is that in LUT-based method the user defines the LUT truth table directly. In the Schematic method,the users place inputs, outputs and logic primitive gates, such as **AND**, **OR**, **XOR**, etc., connect them, and allows the tool to synthesize the logic network into LUTs.

This demo illustrates how to use schematic method in PLU config tool to configure a wanted LUT which is used to make synthesis according to the logic units used.

In the schematic design demo, we config an example LUT with no real purpose, just for illustration, which has the logic needed between inputs and outputs as below.

2.3.1 Logic between inputs and outputs

For this demo, we set logic between inputs and outputs as shown in Table 3.

Table 3. Logic between input and output for LUT-based design demo

Input3	Input2	Input1	Output1 (Red LED, ON for 0)	Output2 (Green LED, ON for 0)
PLU_IN4	PLU_IN3	PLU_IN2	PLU_OUT0	PLU_OUT1
P0_19	P0_20	P0_21	P0_14	P0_15
LUT_IN2	LUT_IN1	LUT_IN0	LUT_OUT0	LUT_OUT1
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	

Table continues on the next page...

Table 3. Logic between input and output for LUT-based design demo (continued)

Input3	Input2	Input1	Output1 (Red LED, ON for 0)	Output2 (Green LED, ON for 0)
0	0			0
0	1			1
1	0			1
1	1			1

2.3.2 PLU Config tool usage

1. Open a new design in the PLU Config tool to choose the schematic design option. Place needed Inputs 1, 2, and 3, and output 1 and 2, add logic gates AND1, OR1 in the schematic, and draw the connections between inputs, logic units and output, as shown in Figure 10. The number of gates that can be added is unlimited. Only inputs, outputs and flip-flops have a restricted quantity.

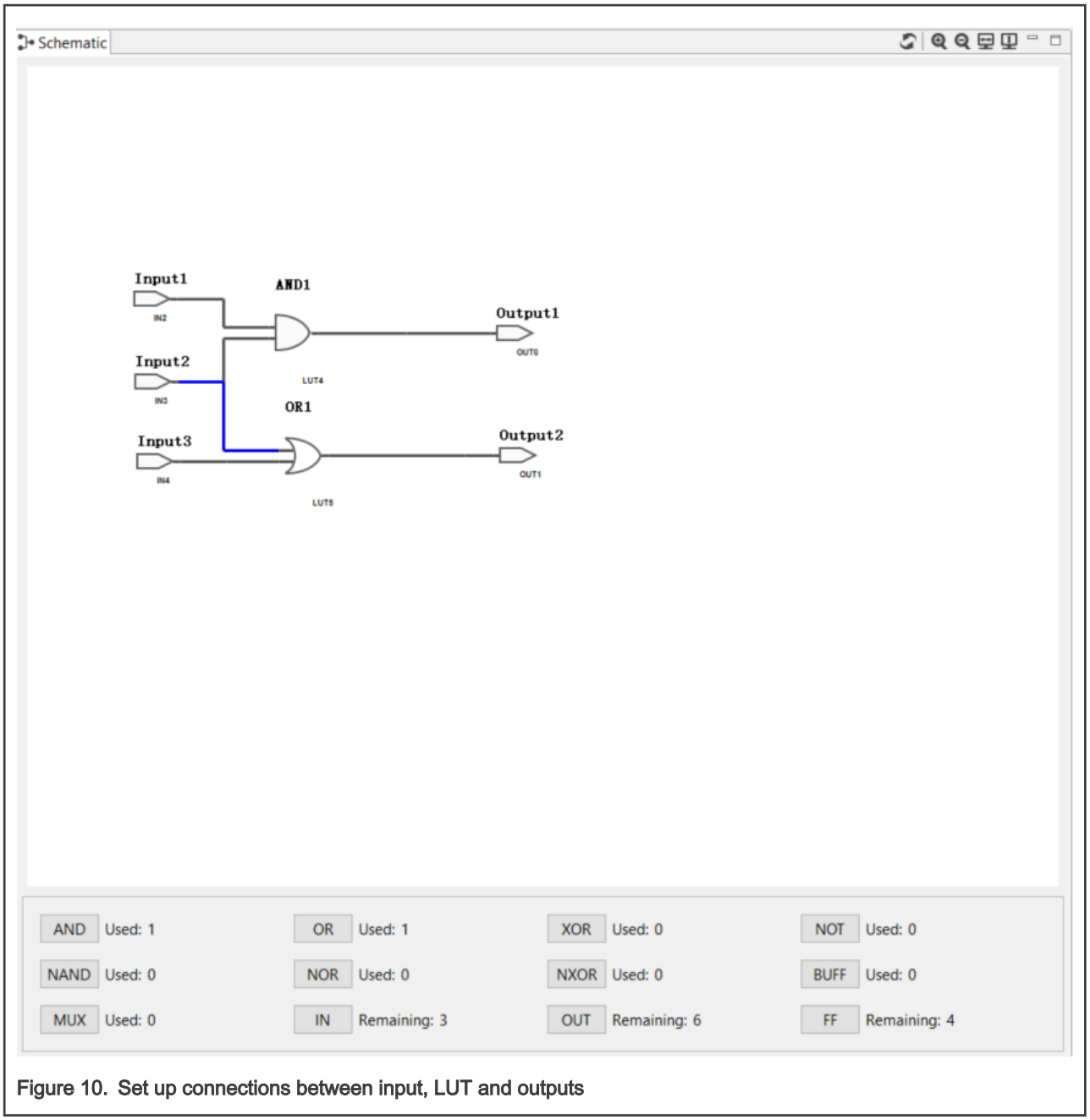

Figure 10. Set up connections between input, LUT and outputs

2. After completing the schematic, on the left panel of the tool, to map PLU input and output to specific input/output resources, click the **mapping** and select from the pull-down menu. The demo code uses `PLU_INPUT2` for input1, `PLU_INPUT3` for input2, `PLU_INPUT4` for input3, `PLU_OUTPUT0` for output1, `PLU_OUTPUT1` for output2, as shown in Table 3.

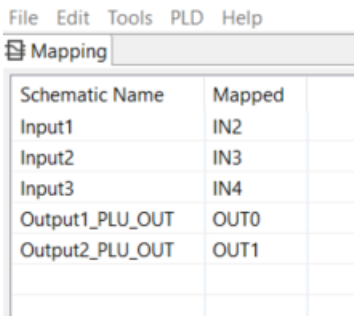

Figure 11. Map PLU inputs and output to LUTs

3. In the top-level menu, select PLD menu and then click **Finalize**. PLD is a tool to optimize the logic design, mapping it to the available LUTs. It is implemented using an open source and a third party (UCLA) tool, and is integrated in the PLU Config Tool installation. If you want to use the standard versions of these tools just click next when the Optimize wizard dialog appears.

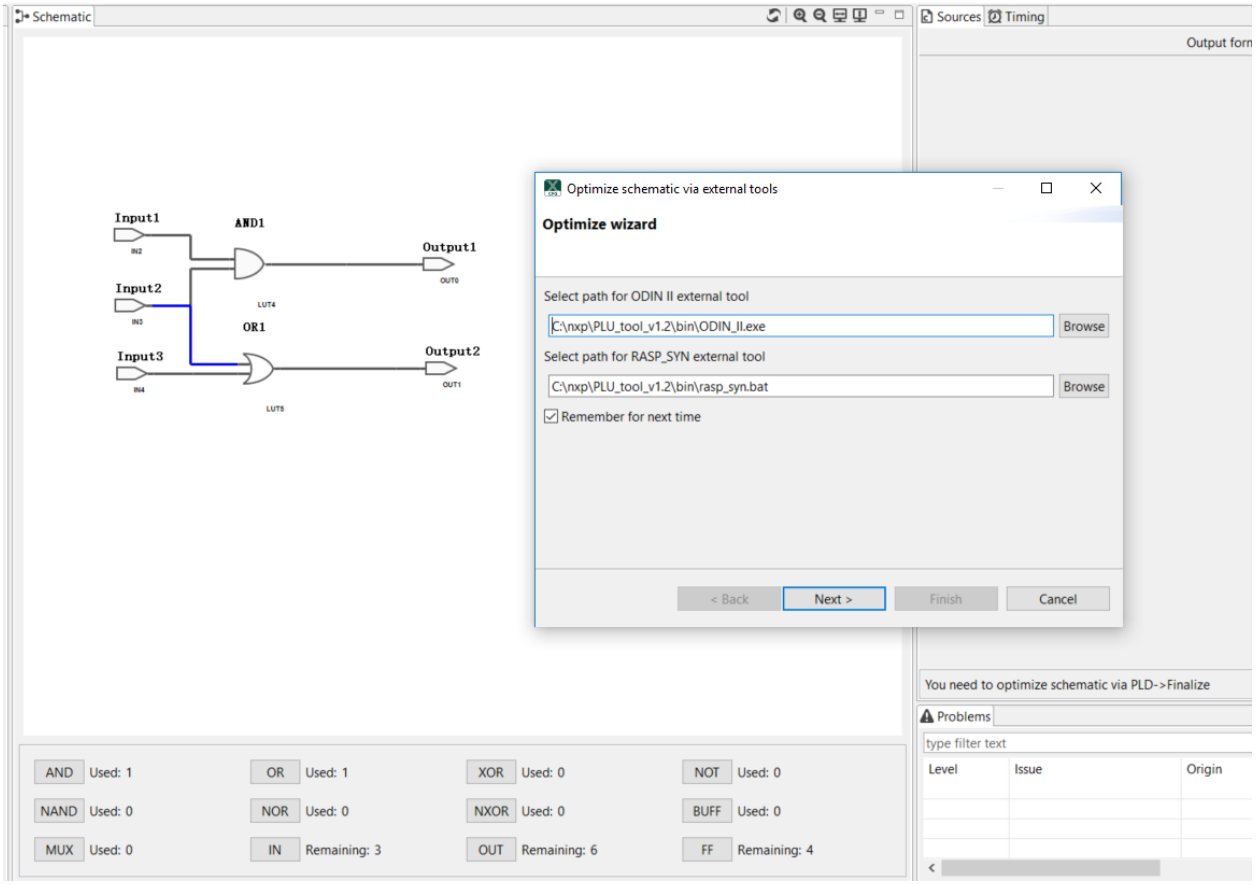

Figure 12. Use PLD to optimize the design

4. Once the finalize operation has completed, click on the **Sources** tab in the right-hand panel of the tool and observe the C code that has been generated. This C code can now be used to set up the PLU configuration register.

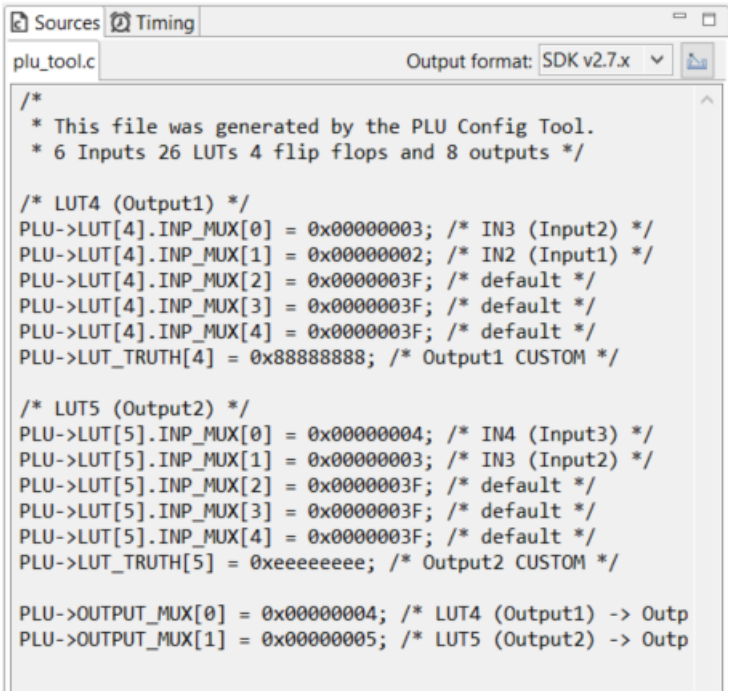

Figure 13. Generate C code to set up PLU configuration register

5. Finally, we can modify the SDK PLU demo code in the *lpcxpresso804_plu_AN_Schematic_design* project. The LUT4 and LUT5 are used by default by the tool, while in the demo we used LUT0 and LUT1, so the LUT4 and LUT5 in the C Code, can be changed to LUT0 and LUT1 in the output code.

   The generated C code sets a value of `0x88888888` for `LUT_TRUTH[4]`, and `0xeeeeeeee` for `LUT_TRUTH[5]`, because in this demo, only two inputs are used. As shown in Table 3, the `LUT_OUT0` and `LUT_OUT1` only use four values in the truth table, so we need only the first four bits to configure the LUT, the configuration value can use `0x00000008` for LUT0 and `0x0000000E` for LUT1.

   In lines 30 and 31 of the *plu_combination.c* file, modify the configured value according to the generated C code, and in order to enabled changes of the Red and Green LEDs only, configure LUT2 (Blue) to **0xFF**.

```
13 #include "pin_mux.h"
14 /*******************************************************************************
15  * Definitions
16  ******************************************************************************/
17
18 /* PLU module */
19 #define DEMO_PLU_BASE         PLU
20 #define DEMO_PLU_LUT_IN_SRC_0 kPLU_LUT_IN_SRC_PLU_IN_2
21 #define DEMO_PLU_LUT_IN_SRC_1 kPLU_LUT_IN_SRC_PLU_IN_3
22 #define DEMO_PLU_LUT_IN_SRC_2 kPLU_LUT_IN_SRC_PLU_IN_4
23
24 #define DEMO_PLU_LUT_OUT_SRC_0     kPLU_OUT_SRC_LUT_0
25 #define DEMO_PLU_LUT_OUT_SRC_1     kPLU_OUT_SRC_LUT_1
26 #define DEMO_PLU_LUT_OUT_SRC_2     kPLU_OUT_SRC_LUT_2
27 //#define DEMO_PLU_LUT_0_TRUTH_TABLE 0x000004D /* 0b01001101 */
28 //#define DEMO_PLU_LUT_1_TRUTH_TABLE 0x000002B /* 0b00101011 */
29 //#define DEMO_PLU_LUT_2_TRUTH_TABLE 0x0000017 /* 0b00010111 */
30 #define DEMO_PLU_LUT_0_TRUTH_TABLE 0x0000008 /* 0b1000  using for AN demo code for schematic design */
31 #define DEMO_PLU_LUT_1_TRUTH_TABLE 0x000000E /* 0b1110  using for AN demo code for schematic design*/
32 #define DEMO_PLU_LUT_2_TRUTH_TABLE 0x00000FF /* 0b11111111  Disable Blue LED Lighting in the Demo*/
33
```

Figure 14. Modify SDK PLU demo code according generated C code

2.3.3 Schematic design demo result

Build and run the code. Input **0** or **1** for input1, 2, or 3 in the **Console** window, and observe the Red and Green LED status, as shown in Table 4.

Table 4. Schematic design demo result

Input3	Input2	Input1	Output1 Red LED	Output2 Green LED
0 or 1	0	0	ON	
0 or 1	0	1	ON	
0 or 1	1	0	ON	
0 or 1	1	1	OFF	
0	0	0 or 1		ON
0	1	0 or 1		OFF
1	0	0 or 1		OFF
1	1	0 or 1		OFF

3 Reference

For further information, see Training Video.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: December 21, 2020

Document identifier: AN13101

arm