Gray counter in VHDL

Ing. Ivo Viščor, 2nd year of PhD study, (ivovi@isibrno.cz)
worked out on the IREL FEE BUT
Supervisor: Doc. Ing. Jaromír Kolouch, CSc.

Abstract: This paper presents the simple structural description of the Gray counter with variable width in VHDL.

Introduction

Programmable logic devices (PLDs) and field programmable gate arrays (FPGAs) can be used to integrate large amounts of logic in a single IC. As the capacity of PLDs and FPGAs is grooving, designers can no longer use Boolean equations or gate-level descriptions to quickly and efficiently complete a design. The solution is VHDL. Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) became standard for designing with programmable logic devices and Application Specific Integrated Circuits (ASICs). VHDL was established as the IEEE 1706 standard in 1987 [1]. The main advantage of VHDL is portability of code across vendors and devices. The main disadvantage is the loosing control of gate level circuit implementation, which is the penalty for portability.

Gray code is the code with only one bit transition between adjacent words. The direct description of Gray counter is based on the equation extraction from the truth table. Such solution of n-bit counter demands 2^{n-2} product terms [2]. Implementation may be difficult for greater width of counter. The alternative is the using of the auxiliary bit [2]. This bit is changed every clock period and corresponds to bit 0 of a binary counter. With assumption that the auxiliary bit extends Gray code word to the right, the particular bit is changed whenever the less significant bits create word 1,0,…0. The exception is MSB which is changed in addition by word 0,…0. Another description of Gray counter use conversion of Gray code to binary code. After the increment is done in binary code, the back conversion follows.

Description of the Gray counter in VHDL

The forms of VHDL circuit design are: behavioural description (e.g. if..than statement), dataflow description (e.g. Boolean equations) and structural description (netlist of blocks). The good VHDL synthesis tools find near optimal solution regardless of the form of description [2]. Although all of the forms are used in proposed description, the structural description is the most important. The main requirements on the design were code simplicity and adjustable width of the counter.

The design is based on the auxiliary bit generation, series of one-bit blocks and glue logic for MSB generation. The basic block of the counter is one-bit block ‘gray_1’ (Fig.1.). This block is encapsulated into a package and used repeatedly in top-level module ‘gray_n’ (Fig.2.). The top-level module is figured with counter width parameter of 3.

The one-bit block ‘gray_1’ generates the particular bit of Gray code ‘qout’ from input signals ‘qin’ and ‘zin’ in a T-type flip-flop (Fig.1). The ‘qin’ is connected to the less significant bit in the chain. Second generated signal is ‘zout’. The ‘zout’ represents expression: “All the less significant bits (including auxiliary bit) are zero”. Each one-bit block shares the same asynchronous reset and clock signals (Fig.2.). The ‘q(1)’ to ‘q(3)’ outputs are valuable bits of the Gray code. The D-type flip-flop generates ‘q(0)’ signal which represents the auxiliary (parity) bit. The MSB reset at the end of the cycle is assured by the OR gate.

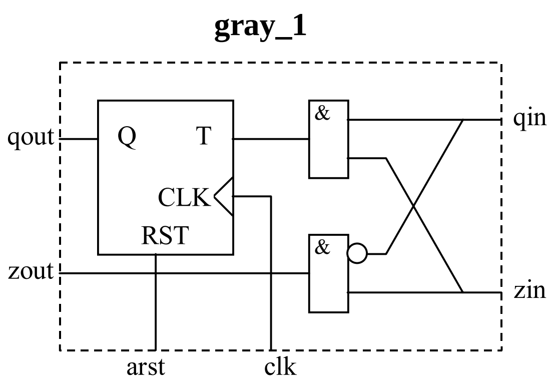

Fig.1. Schematic of one-bit block package ‘gray_1’

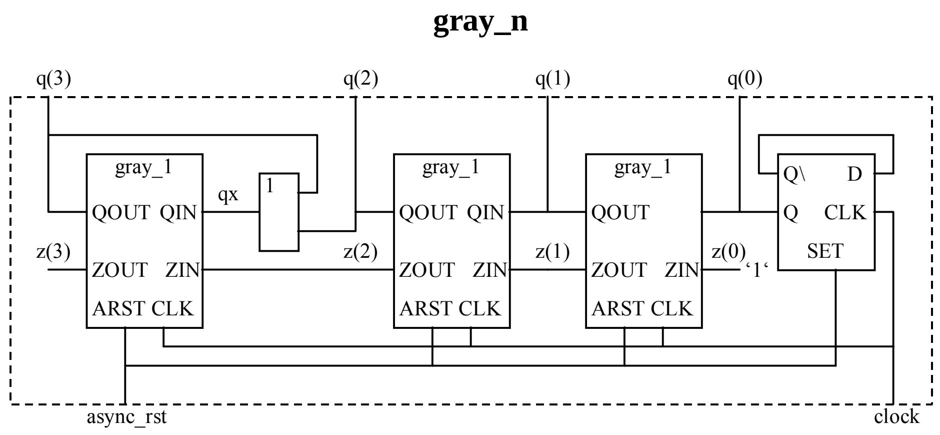

Fig.2. Schematic of top-level module ‘gray_n’ (width = 3)

Although pictured schematics provide sufficient information to write the VHDL code, it is not easy to fulfil the VHDL syntax, especially for the beginners. The great helper is the VHDL help [3]. Full listing of code exceeds size of this contribution and will be presented on the poster.

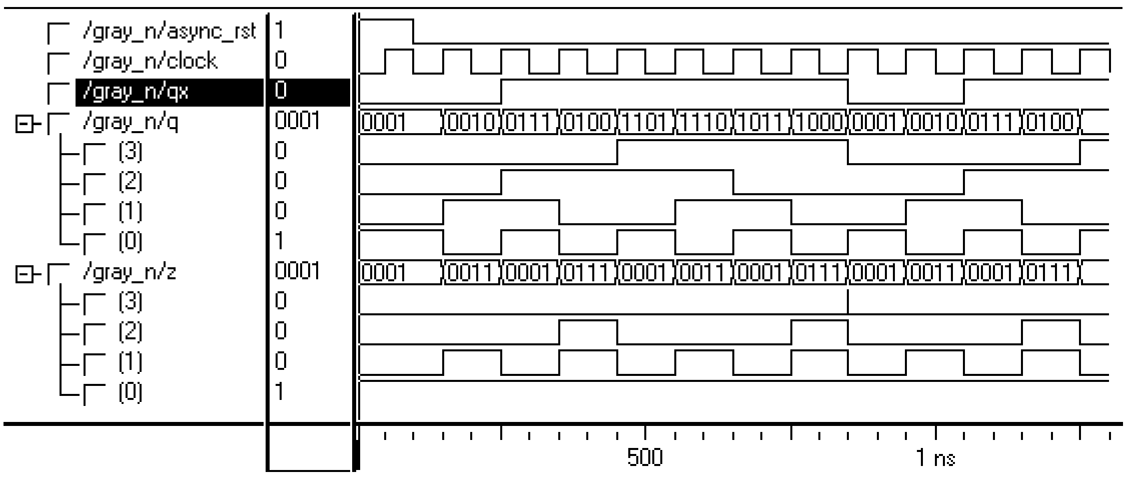

Fig.3. Functional simulation of the VHDL code in the ModelSimXE Starter tool (width = 3)

Described VHDL code was successfully tested in two simulators. The Warp 4.0 software provides only postlayout ('timing') simulation. The VHDL code had to be fitted to a particular device first. The Webpack from Xilinx on the opposite side provides only the source code ('functional') simulation. This simulation tool is free licensed and is restricted up to 500 rows of code. The snapshot from the functional simulation of the Gray counter is figured at Fig.3.

Implementation of the Gray counter

The implementation of the proposed VHDL code (Tab.1.) illustrates the fact, that small counter designs are only I/O pin limited, whilst large counters are device architecture limited. No speed performance and compile options was explored. These results are in no way the device benchmarks, the far more complex and definite benchmarks failed (e.g. PREP [1]).

Type	**Part #**	Description	Available I/O	Package	**Max. width of counter (gray_n)**	Note
FPGA	**C388P**	8k gates	172	144 TQFP	**170**	1)
CPLD	**C375**	128 MCells	128	160 TQFP	**34**	1)
PLD	**22V10**	10 MCells	10	24 DIP	**9**	1), 3)
CPLD	**XC95144XL**	144 MCells	117	144 TQFP	**103**	2)
CPLD	**XCR3320**	320 MCells	112	160 TQFP	**72**	2)
CPLD	**XCR3032A**	32 MCells	28 (32)	44 TQFP	**27**	2), 4)

Note: 1) Vendor: CYPRESS, software: Warp 4.0.
2) Vendor: XILINX, software: WebPack (May 2000).
3) Modified VHDL code without asynchronous reset or without auxiliary bit pinout.
4) Dedicated pins for JTAG ISP lowers available I/O pins by 4.

Tab. 1. Examples of the Gray counter implementation

Conclusion

The presented VHDL design of the Gray counter with variable width is as simple as possible and uses all main technique of VHDL programming. The design is therefore suitable for education purposes. Implementation examples demonstrate fact, that even the VHDL software at no charge (Webpack) can be applicable for the first VHDL experiments.

Acknowledgement

This work was supported by the grant no. 102/00/1262 of the Grant Agency of the Czech Republic.

References

[1] Skahill, K.: *VHDL for programmable logic*. Menlo Park, CA: Addison-Wesley Publishing Company, 1996

[2] Kolouch, J.: *Čítače pracující v Grayově kódu realizované v obvodech PLD.* Brno: IREL FEE BUT, 1999

[3] VHDL Language Reference Guide, ver. 1.06., windows help file, 1998

Appendix A: VHDL code listing of the one-bit block (gray_1)

```
-- File: gray_1.vhd
-- One bit block for the Gray counter gray_n.vhd
-- 2/2000 IVOVI

-- qout: One bit output of the counter
-- zout: 1, if all the less significant bits are zero

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE pkggray_1 IS
  COMPONENT gray_1
    PORT( arst, clk, qin, zin : IN STD_LOGIC;
          qout                : INOUT STD_LOGIC;
          zout                : OUT STD_LOGIC);
  END COMPONENT;
END pkggray_1;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY gray_1 IS
  PORT( arst, clk, qin, zin : IN STD_LOGIC;
        qout                : INOUT STD_LOGIC;
        zout                : OUT STD_LOGIC);
END gray_1;

ARCHITECTURE archgray_1 OF gray_1 IS
BEGIN
  PROCESS(arst, clk)
    BEGIN
    IF arst='1' THEN
      qout <= '0';
    ELSIF clk'EVENT AND clk='1' THEN
      qout <= qout XOR (qin AND zin);
    END IF;
  END PROCESS;
  zout <= zin AND NOT qin;
END archgray_1;
```

Appendix B: VHDL code listing of the top-level of Gray counter (gray_n)

```
-- File: gray_n.vhd
-- Gray counter with variable width (generic width)
-- 2/2000 IVOVI

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY gray_n IS GENERIC(width: INTEGER:=3);
  PORT( async_rst, clock : IN STD_LOGIC;
        q                : INOUT STD_LOGIC_VECTOR(width DOWNTO 0));
END gray_n;

ARCHITECTURE archgray_n OF gray_n IS
  COMPONENT gray_1 PORT( arst, clk, qin, zin : IN STD_LOGIC;
                         qout                : INOUT STD_LOGIC;
                         zout                : OUT STD_LOGIC);
  END COMPONENT;
  -- inner interconnection of 1-bit sections
  SIGNAL z  : STD_LOGIC_VECTOR(width DOWNTO 0);
  -- auxiliary signal for MSB
  SIGNAL qx : STD_LOGIC;
BEGIN
  -- less significant bits
  create_lsb: FOR i IN 1 TO width-1 GENERATE

                createbit: gray_1 PORT MAP( async_rst, clock,
                                            q(i-1), z(i-1),
                                            q(i), z(i));
                END GENERATE;
  -- most significant bit
  create_msb: gray_1 PORT MAP( async_rst, clock,
                               qx, z(width-1),
                               q(width), z(width));
  -- auxiliary signal for MSB
  qx <= q(width-1) OR q(width);
  -- parity bit generation
  PROCESS(async_rst, clock)
    BEGIN
    IF async_rst='1' THEN
      q(0) <= '1';
    ELSIF clock'EVENT AND clock='1' THEN
      q(0) <= NOT q(0);
    END IF;
  END PROCESS;
  z(0) <= '1';
END archgray_n;
```