Universidade de Aveiro

Mestrado em Engenharia de Computadores e Telemática

Arquitecturas de Alto Desempenho

VHDL Simulation

Academic year 2022/2023

1. A *population counter* is a digital circuit which counts the number of bits of a word that have the value one. Assume that words are 32 bits long. Many solutions are possible. You should seek for solutions that trade a balance between hardware complexity and processing time in two different situations. In the first, the word is available in parallel; in the second, the word is provided bit by bit. For each case, do the following:

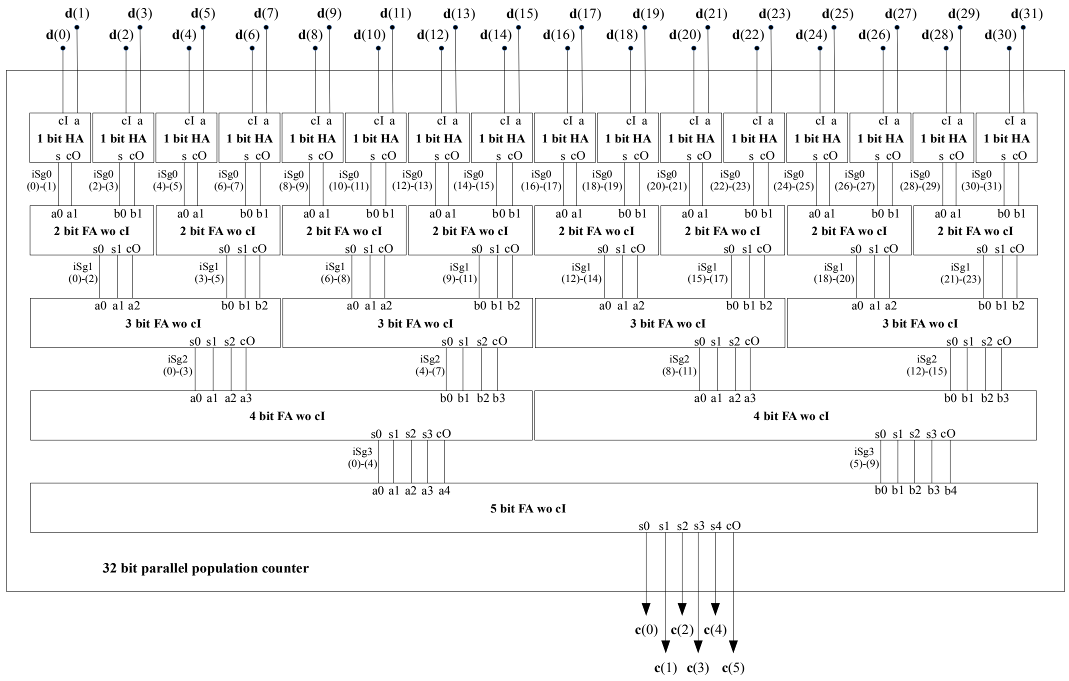
Parallel solution

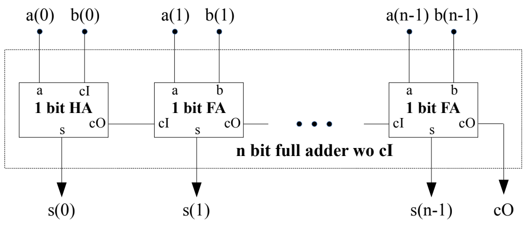
1.1. Specify the circuit interface and draw a schematics of its internal organization.

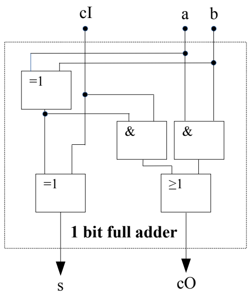
A straightforward solution to the problem arises from the following set of observations

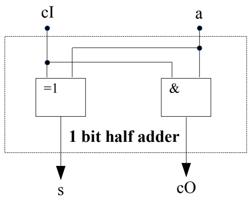
– a 32 bit population counter can be built using two 16 bit population counters plus a 5 bit full adder with *carry in* set to zero
– a 16 bit population counter can be built using two 8 bit population counters plus a 4 bit full adder with *carry in* set to zero
– a 8 bit population counter can be built using two 4 bit population counters plus a 3 bit full adder with *carry in* set to zero
– a 4 bit population counter can be built using two 2 bit population counters plus a 2 bit full adder with *carry in* set to zero
– a 2 bit population counter can be built using a 1 bit half adder.

Hence, resulting in









Implementation cost

16 1-bit half adders + 8 2-bit full adders wo cI + 4 3-bit full adders wo cI + 2 4-bit full adders wo cI + 1 5-bit full adder wo cI

31 1-bit half adders + 26 1 bit full adders

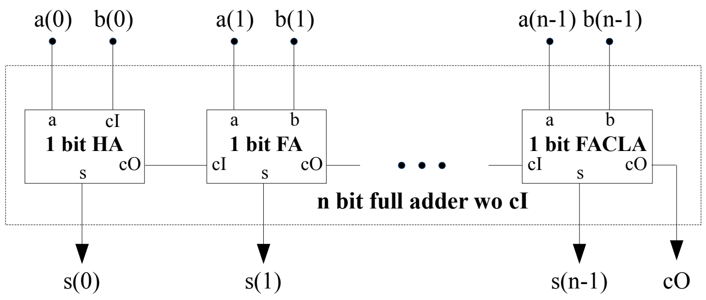
83 *and* gates + 26 *or* gates + 83 *xor* gates

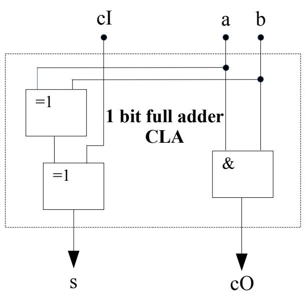
Worst case propagation delay

(it arises from the propagation of the carry out signal at each stage but the first)

stage 0: 1 *xor* gate
stage 1: 1 *xor* gate + 1 *and* gate + 1 *or* gate
stage 2: 1 *xor* gate + 2 *and* gates + 2 *or* gates
stage 3: 1 *xor* gate + 3 *and* gates + 3 *or* gates
stage 4: 1 *xor* gate + 4 *and* gates + 4 *or* gates
total: 5 *xor* gates + 10 *and* gates + 10 *or* gates

A further observation leads to more simplification: *not all operand values can appear at stages 1 through 5*. In fact, at stage 1, 2 (10, in binary) is the maximum value; at stage 2, 4 (100, in binary) is the maximum value; at stage 3, 8 (1000, in binary) is the maximum value; at stage 4, 16 (10000, in binary) is the maximum value.
Thus, the n-bit full adder wo cI can be modified as





Implementation cost

16 1-bit half adders + 8 2-bit full adders wo cI + 4 3-bit full adders wo cI + 2 4-bit full adders wo cI + 1 5-bit full adder wo cI

31 1-bit half adders + 11 1 bit full adders + 15 1 bit full adders CLA

68 *and* gates + 11 *or* gates + 83 *xor* gates

Worst case propagation delay

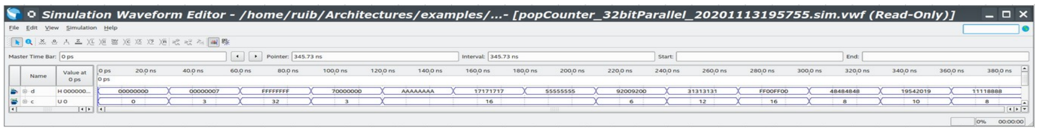
(it arises from the propagation of the most significant *s* signal at each stage)

stage 0: 1 *xor* gate
stage 1: 2 *xor* gates
stage 2: 2 *xor* gates + 1 *and* gate + 1 *or* gate
stage 3: 3 *xor* gates + 2 *and* gates + 2 *or* gates
stage 4: 4 *xor* gates + 3 *and* gates + 3 *or* gates
total: 10 *xor* gates + 6 *and* gates + 6 *or* gates

1.2. Write the VHDL code that describes it.

In a separate file.

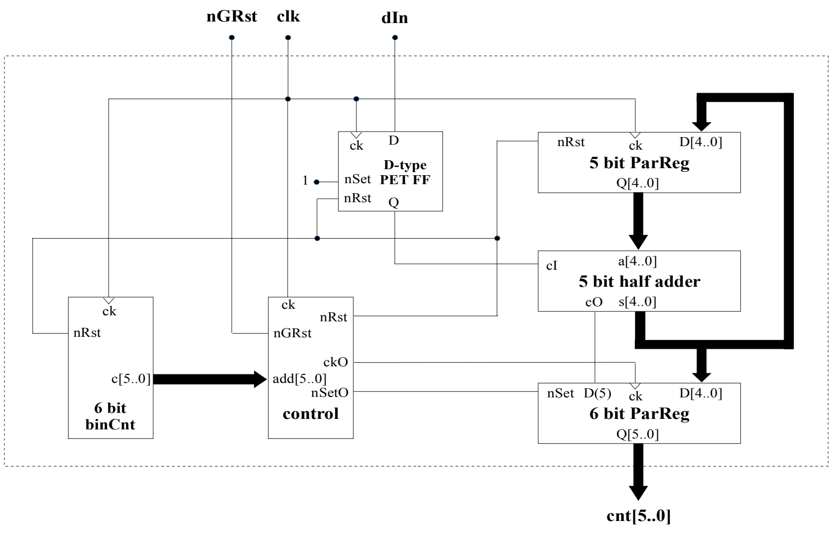
1.3. Create a Quartus project and simulate its operation.

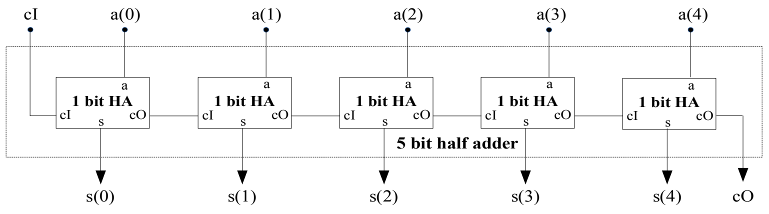

Simulation Waveform Editor - /home/ruib/Architectures/examples/...- [popCounter_32bitParallel_20201113195755.sim.vwf (Read-Only)]
File
Edit
View
Simulation
Help

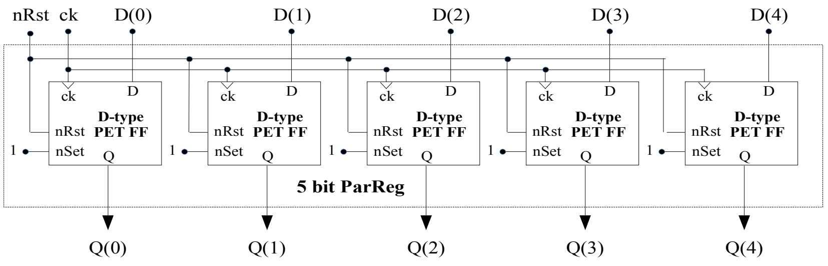
```
Master Time Bar: 0 ps    Pointer: 345.73 ns    Interval: 345.73 ns    Start:    End:

Name   Value at 0 ps   0 ps  20.0 ns  40.0 ns  60.0 ns  80.0 ns  100.0 ns  120.0 ns  140.0 ns  160.0 ns  180.0 ns  200.0 ns  220.0 ns  240.0 ns  260.0 ns  280.0 ns  300.0 ns  320.0 ns  340.0 ns  360.0 ns  380.0 ns
d      H 000000...     00000000 | 00000007 | FFFFFFFF | 70000000 | AAAAAAAA | 17171717 | 55555555 | 92009200 | 31313131 | FF00FF00 | 48484848 | 19542019 | 11118888
c      U 0             0        | 3        | 32       | 3        | 16                             | 6        | 12       | 16       | 8        | 10       | 8

0%   00:00:00
```

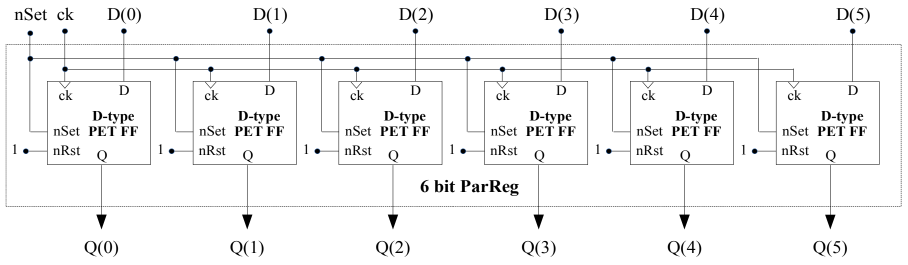
Bit-serial solution

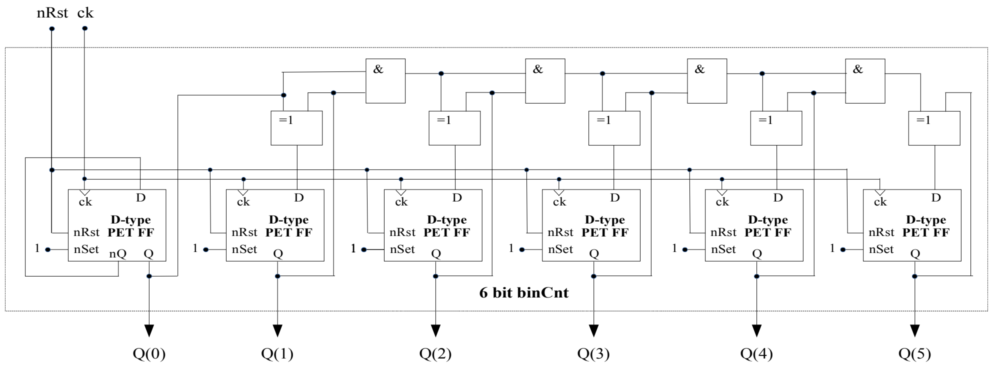
1.1. Specify the circuit interface and draw a schematics of its internal organization.

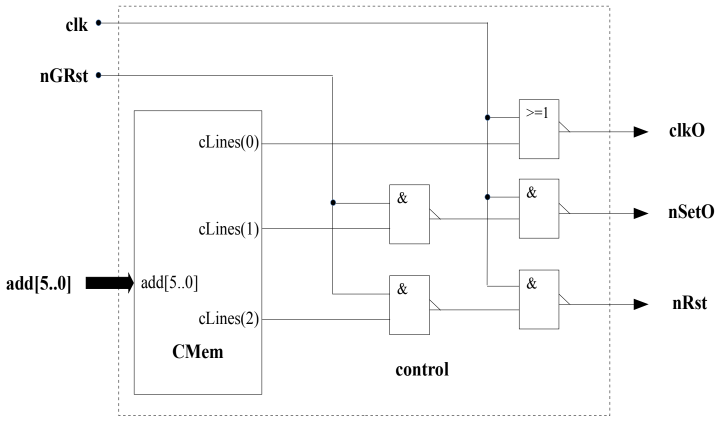












<u>ROM contents</u>

add[5..0]	cLines[2..0]
00	7
01	5
02	7
. . .	. . .
1F	7
20	6
21	3
22	7
. . .	. . .
3F	7

<u>Implementation cost</u>

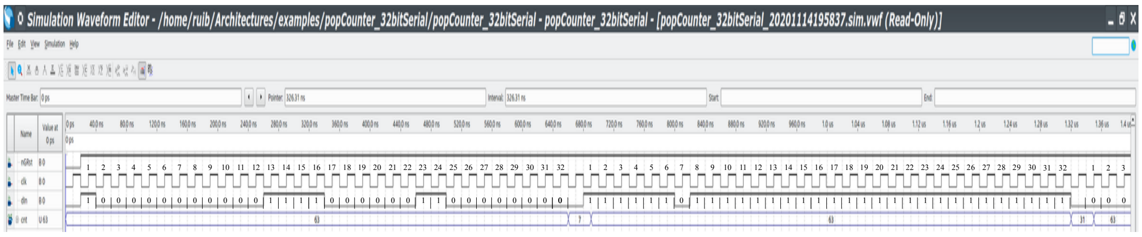
1 D-type flip flop + 1 5-bit parallel register + 1 5-bit half adder + 1 6-bit parallel register + 1 6-bit binary counter + 1 control unit

18 D-type flip flops + 9 *and* gates + 10 *xor* gates + 4 *nand* gates + 1 *nor* gate

1.2. Write the VHDL code that describes it.

In a separate file.

1.3. Create a Quartus project and simulate its operation.



Simulation Waveform Editor - /home/ruib/Architectures/examples/popCounter_32bitSerial/popCounter_32bitSerial - popCounter_32bitSerial - [popCounter_32bitSerial_20201114195837.sim.vwf (Read-Only)]

Master Time Bar: 0 ps Pointer: 326.31 ns Interval: 326.31 ns Start: End:

Name	Value at 0 ps
nGRst	B 0
clk	B 0
din	B 0
cnt	U 63