Design of a memory-mapped multicore coprocessor

Lecture 12 on Dedicated systems

Teacher: Giuseppe Scollo

University of Catania
Department of Mathematics and Computer Science
Graduate Course in Computer Science, 2017-18

Table of Contents

lecture topics

outline:

- codesign evolution for computations on Collatz trajectories
- evolution of the delay computation
  - performance enhancement
  - domain extension
- a glimpse at functions over Collatz trajectories
- a few ideas for functional extensions of the codesign

design space

possible evolutions of the codesign implemented in the previous lab tutorial may be conceived along two orthogonal development directions:

- performance enhancement
- functional extensions

an example of orthogonal combination of both directions is given by the following objectives for a first project, dealt with in the next lab tutorial:

- a definite performance enhancement may be delivered by parallel replication of the hardware unit in charge of the delay computation
- on the other hand, a minimal functional extension of definite interest is the widening of the input width, to make the delay computation applicable to a wider domain of trajectories

more significant functional extensions may be identified from consideration of functions, defined over Collatz trajectories, other than the delay but whose computation requires trajectory generation anyway

the challenge here is in the design of multifunction hardware units as well as of a more complex HW/SW interface, apt to selection of the functions of interest and to the related data transfer

codesign evolution of the delay computation

a first design alternative to be considered about the replication of the hardware unit for the delay computation is:

- multiple instances of the computational component on the Avalon bus, or
- construction of a more complex component, embedding multiple copies of the individual computational component

the latter option is preferable in view of possible further extensions that would require access by the different instances to shared data, e.g. defined as configuration parameters

bus transactions for this purpose are avoided in this way, moreover a control hierarchy is implemented whereby the coprocessor is assigned local control functions

other design decisions relate to the number of parallel instances of the computational component, henceforth termed *cores*, and the size of the coprocessor I/O data

- taking it into account that the Nios II processor may transfer at most 32 bits in a single bus transaction, and that it may be useful to have a status register in the coprocessor, with one bit per core, it proves convenient to set the limit on the number of cores at 32
- the data available in the website on the 3x+1 problem by Eric Roosendaal, e.g. in the Class Records Table, indicate a 64-bit minimum size for input data of actual interest, while a 16-bit data size suffices for the delay output

structure of a multicore coprocessor

the 64-bit extension of the single core input is easily obtained by a straightforward modification of the Gezel source from the previous lab tutorial, with the same correction to the VHDL output of the fdlvhd translator

the structural description of the multicore coprocessor in VHDL is eased by the iterative construct **for** <identifier> **in** <range> **generate**, whose usage is exemplified in Zwolinski, 4.5.2, that in our case allows the generation of the $2^n = 32$ core instances, with $n = 5$

the multicore coprocessor then ought to have circuits for the correct dispatching of I/O data between the interface and a core selected by the processor:

- a multiplexer with *n*-bit selection input and 16-bit data ports for the core delay output
- a demultiplexer with *n*-bit selection input and 64-bit data ports for the core initial data input, as well as a similar one but with 1-bit data ports for the core start signal input

description in VHDL of multiplexing is simple if the core outputs are chained in a 2^nx16-bit vector, as it is enough to use a selection operator on the vector

the more complex description in VHDL of demultiplexing is feasible by means of a logical shift operator, as it is exemplified for a generic decoder in Zwolinski, 4.2.3

hardware interface constraints

signal exchange at the multicore coprocessor I/O ports is to be adapted to the available signals of the Avalon-MM interface, taking a few constraints on these into account, such as:

- the data transfer signals, writedata and readdata, must have the same width
- signal address identifies an I/O register in the memory address space assigned to the coprocessor, starting from 0 and with word-addressing by default

since the Nios II processor may transfer at most 32 bits in a single bus transaction, it is convened that this be the coprocessor word width at the Avalon interface, that is to say, the width of the writedata and readdata signals

this entails that the delay output is to be zero-extended, whereas the trajectory input data is to be acquired in two bus cycles

the register address space of the coprocessor is thus the interval [0, 3x2^n], taking one address for the status register into account, hence address is (n+2)-bit wide

a suitable management of register addresses allows a quick identification of the core (or status register) from the value of the address input, for example:

address[n,1] if address[n+1] = 0
address[n-1,0] if address[n+1,n] = 10
status register if address[n+1,n] = 11

functions on Collatz trajectories

the aforementioned website on the 3x+1 problem is a valuable source of inspiration for the codesign of functional extensions of the system considered hitherto; here are a few functions over Collaz trajectories (defined on the initial values, which determine the trajectories):

- Glide(x_0): smallest k such that $x_k < x_0$
- Mx(x_0): peak value in the trajectory from x_0
- Completeness(x_0): ratio between the number of odd values and the number of even values in the the trajectory from x_0 to $x_{delay(x_0)-1}$

certain trajectories are more interesting than others because they establish records (like in a competitive race); in this sense, the following properties are defined: x_0 is a

- Class Record if it is the smallest x such that delay(x) = delay(x_0)
- Delay Record if every $x < x_0$ has delay(x) smaller than that of x_0
- Path Record if every $x < x_0$ has peak Mx(x) smaller than that of x_0
- Completeness Record if every $x < x_0$ has Completeness(x) smaller than that of x_0

a few considerations follow that are useful to design functional extensions of the codesign finalized to the search of such rarities

codesign ideas for functional extensions

functional extensions of the delay computation core, in order to provide the additional computation of the three functions introduced in the previous page, are fairly simple ones

for the completeness computation, the circuit should only provide the number of odd values in the trajectory (final 1 excluded); the number of even values is obtained by difference from the delay, and it is better to compute their ratio (a floating point number) in software

the selection of the desired functions is to be considered in two different contexts:

- as a configuration parameter of the hardware component (before its synthesis); this is useful to save area when not all available functions are meant to be utilized
- as an input to the hardware component for read operations, viz. to select desired results out of the computed ones

it is useful to the search of the aforementioned Records the study of properties of theirs, that *a priori* exclude whole classes of values of x_0 from the search space

for example, 5 is the only Class Record in the congruence class 5 (mod 8); this situation, which is explained by the so-called *coalescence* of trajectories, generalizes to congruence classes $(2^{k-2} + (k \bmod 2)(2^{k-1} - 1)) \pmod{2^k}$ for $k > 2$

optimization techniques for Record search algorithms may be found in the references

references

Zwolinski, Ch. 4, Sect. 4.2.3, 4.5.2

E. Roosendaal, *On the 3x+1 problem*, www.ericr.nl/wondrous

E. Roosendaal, *Technical details*, www.ericr.nl/wondrous/techpage.html

J. Lagarias, The 3x+1 problem and its generalizations, *Amer. Math. Monthly* 92 (1985) 3-23

G.T. Leavens and M. Vermeulen, 3x+1 search programs, *Computers Math. Applic.* 24:11 (1992) 79–99

G. Scollo, Looking for Class Records in the 3x+1 Problem by means of the COMETA Grid Infrastructure, in: R. Barbera (Ed.) *Proc. Symp. "GRID Open Days at the University of Palermo"*, Palermo, 6-7 Dec. 2007; Consorzio COMETA (2008) pp. 255-263