Exercise 8: Data Path and FSM Modelling and Synthesis

In this exercise you will

- model a data path in VHDL,
- model a finite state machine (FSM) to control the data path,
- put them together and simulate them,
- synthesize FSM and data path using different optimization options.

Use the directory **~/vhdl/ex8**. We will work on an implementation that numerically solves the differential equation

$$\frac{d^2y}{dx^2} + 5\frac{dy}{dx}x + 3y = 0$$

- starting from a known point (*x*, *y*) that is part of the solution,
- where *u* is the gradient of the curve at (*x*, *y*),
- *dx* is the increment in direction *x*,
- and (*x*1, *y*1) is the new computed point on the curve.

Figure 8.0 shows a listing of an algorithm that computes an approximation of the solution point by point until the *x* coordinate reaches the limit *a*:

```
loop
    x1 := x + dx;
    y1 := y + (u * dx);
    u  := u - (5 * x * u * dx) - (3 * y * dx);
    x  := x1;
    z  := y1;
exit when not (x1 < a);
end loop;
```

Figure 8.0: Differential equation solver

1. The files diff_eq_solver.vhd (algorithm) and tb_diff_eq_solver.vhd (test bench) are provided. Compile and simulate (run –all), trace the signals.

The following Figure shows which operation of the algorithm is computed in which cycle. This is called the schedule. The boxes on top of the figure represent the hardware resources (2 multipliers, 1 adder/subtractor, 1 comparator) that will be used in the data path you will implement. The colors show the binding of operations to resources, i.e. which operation is computed by which resource.

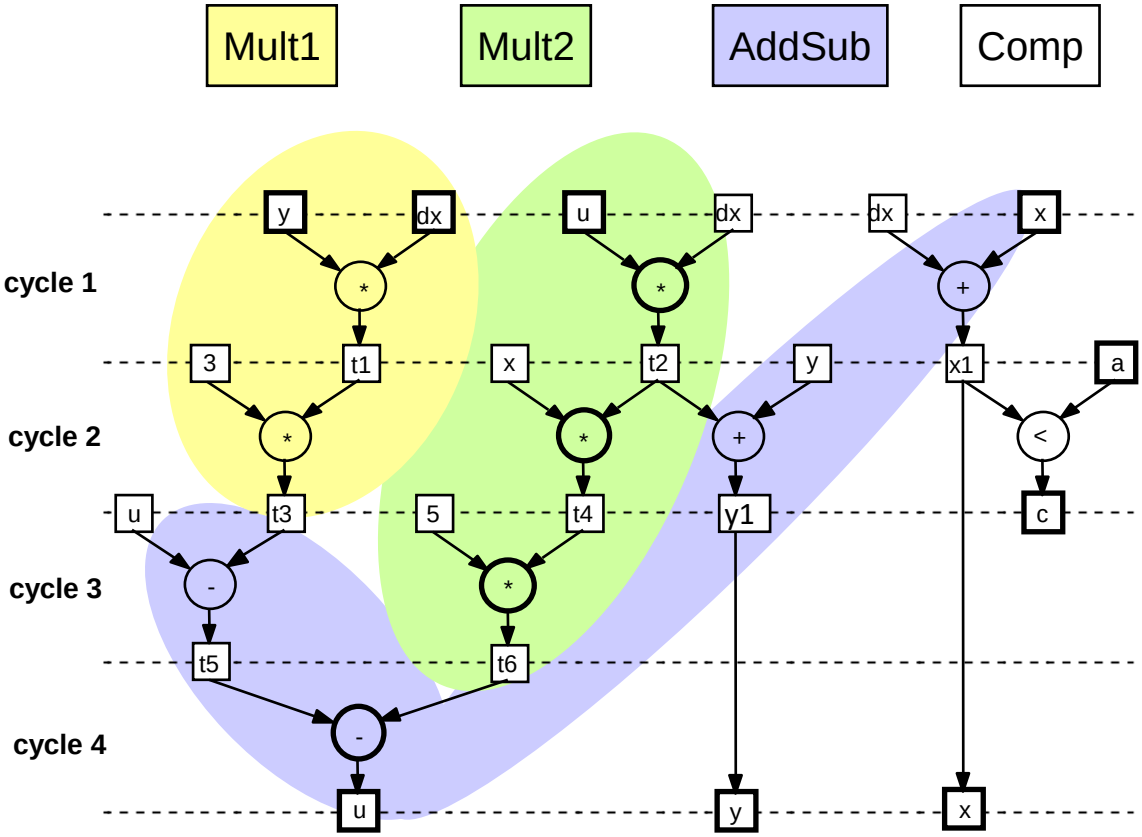

Figure 8.1: Schedule, resources, and binding

2. Figure 8.2 shows a data path that we will use to implement the algorithm according to the developed schedule and with the selected resources. Your job is to implement this data path in VHDL, using data type INTEGER for all variables / signals. The VHDL description shall be synthesizable. Do not care for overflow or bit-width minimization yet.

***Note** that additional names must be introduced for control signals of the data path.*

***Note** that you can specify the functionality of a register and the preceding combinational logic in a common process.*

The initialization of registers x, y and u to the initial values given by X0, Y0 and U0 should be performed upon reset.

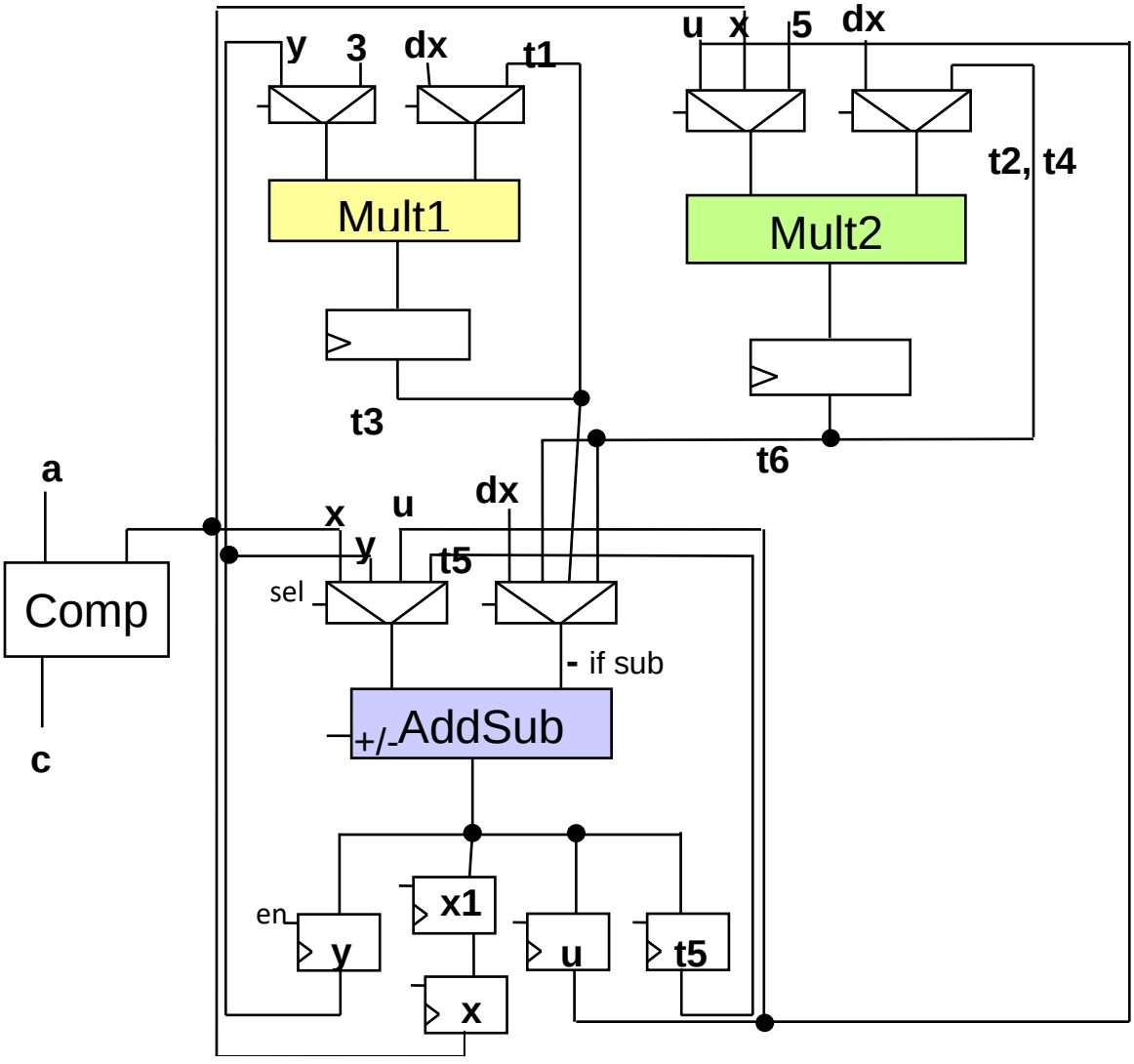

Figure 8.2: Data path

3. The next step is to implement a controller for the data path. For a specification of the controller, see Figure 8.3 and Figure 8.4. The names of control signals and their encoding may be different in your implementation. You may also have additional control signals. Make sure that your VHDL FSM works for your data path.

   ***Note** that the table only contains the data for the adder, its operands, and its results. You need to derive a similar table for the multipliers!*

cycle	mux1	mux2	+/-	reg y	reg x1	reg u	reg t5
1	select x	select dx	add	disable	enable	disable	disable
2	select y	select t2	add	enable	disable	disable	disable
3	sel. t3	select u	sub	disable	disable	disable	enable
4	sel. t6	select t5	sub	disable	disable	enable	disable

Figure 8.3 Cycles and resource actions for one iteration of the algorithm's loop

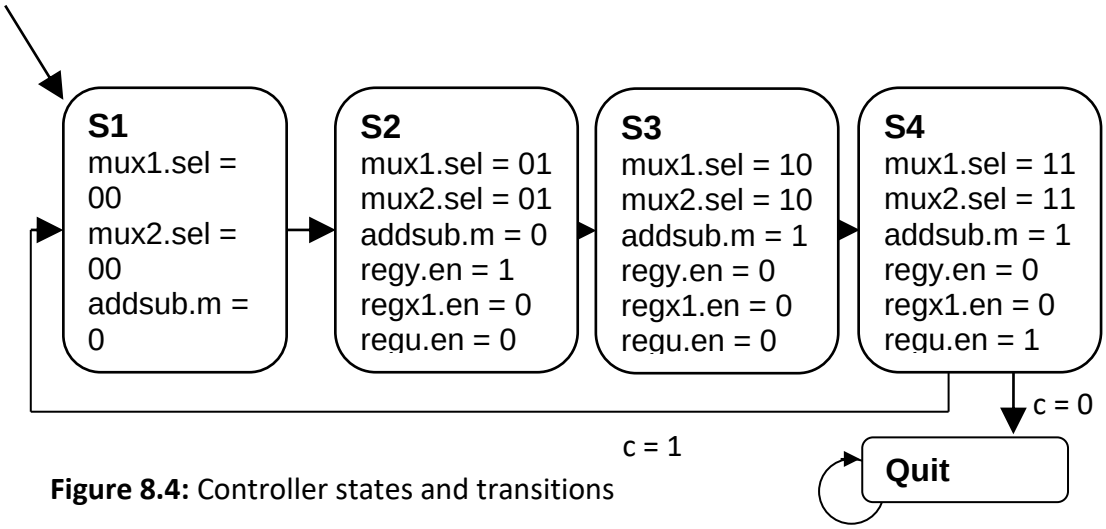

Figure 8.4: Controller states and transitions

4. Integrate the data path and the controller and write a test bench. To compute expected results, the test bench should run the original algorithm (Figure 8.0) as a reference model.

If you have time:

5. Synthesize your design. Look for FSM optimization options of the synthesis tool.