The 5-minute introduction to FSMDs for practitioners

A design approach that is widely used by HLS (high-level synthesis) tools but is not really advertised loud and proud to HLS users is the Finite-State Machine with Datapath, aka FSMD. For instance, the Wikipedia entry on FSMDs is really sketchy. FSMDs are the primary approach for dealing with generic/control-flow dominated codes in an HLS context.

An FSMD is a microarchitectural paradigm for implementing non-programmable/ hardwired processors with their control unit and datapath combined/merged. In FSMDs, the datapath actions are embedded within the actual next state and output logic decoder of your FSM description. From an RTL (Register Transfer Level) abstraction point you can view an FSM as comprising of:

- a current state logic process for updating state and register storage
- a next state logic process for calculating the subsequent state to transition
- an output logic process for producing the circuit's outputs.

[*NOTE*: There is an excellent writeup on alternate FSM description styles in VHDL by Douglas J. Smith that you can consult; any recent XST manual provides good advice if targeting Xilinx FPGAs for casual RTL coding of FSMs.]

Let's see FSMDs as considered by HercuLeS high-level synthesis (http://www.nkavvadias.com/hercules/); a manual for HercuLeS is here: http://www.nkavvadias.com/hercules-reference-manual/hercules-refman.pdf while a relevant book chapter can be downloaded from: http://cdn.intechweb.org/pdfs/29207.pdf if you want to go beyond these five minutes.

HercuLeS' FSMDs are based on Prof. Gajski's and Pong P. Chu's work, mostly on some of their books and published papers. When I had started my work on HercuLeS, I had rented a couple of Gajski's books from the local library and had actually bought two of P.P. Chu's works; the RTL Hardware Design using VHDL book is highly relevant. Gajski's work on SpecC and the classic TRs (technical reports such as Modeling Custom Hardware in VHDL) from his group were at some point night (by the bed) and day (by the desk) readings...

I believe Vivado HLS (aka AutoESL/xPilot) and the others do the same thing, following a very similar approach, with one key difference on how the actual RTL FSMD code is presented. Their datapath code is implemented with concurrent assignments and there are lots of control and status signals going in and out of the next state logic decoder. On the contrary I prefer datapath actions embedded within state decoding; produces a little slower and marginally larger hardware overall, but the user's intention in the RTL is much more clear and it is to grasp and follow.

In an FSMD, the key notion is understanding how the `_reg` and `_next` signals work as they represent a register, i.e. its currently accessible value and the value that is going to be written into that register. Essentially `_reg` and `_next` is what you can see if probing the register's output and input port at any time.

If following the basic principles from Pong P. Chu, every register is associated to a `_reg` and a `_next` signal. Some advice:

1) Have a `_reg` and `_next` version for each register as declared signals in VHDL code.

2) In each state, read all needed `_reg` signals and assign all needed `_next` ones.

3) Donnot reassign the same `_next` version of a register within a single FSMD state.

4) You can totally avoid variables in your code. Not all tools provide equally mature support for synthesizing code with variables.

5) Operation chaining is possible but requires that you write `_next` versions and read them in the same state. Then these are plain wires and donnot implement registers. Again, you can't peruse (for writing) the same `_next` version more than once in the same state.

At some point I had developed a technique for automatically modifying a VHDL FSMD code for adding controlled operation chaining. It just uses a lexer and to read more about it, see chapter III.E of http://www.nkavvadias.com/publications/kavvadias_asap12_cr.pdf.

If you have a deeper curiosity on HercuLeS, you can read http://www.nkavvadias.com/publications/hercules-pci13.pdf; a journal paper has been accepted for publication and will soon be available. I had to say it!