Introduction to Register-Transfer Level (RTL) Design:  
from Generalized ASM’s to a Microarchitecture  
[PART 2]

(Continuation of Handout #30a [Part I].)

Step #4. Identify Status Signals

Status signals are the outputs of the datapath which are inputs to the control unit (see Handout #29, fig. 8.1). In the Moore Generalized ASM, all status signals are indicated as conditions in decision boxes.

Note, though, that not all inputs in decision boxes are status signals: decision boxes also can contain “control inputs”, such as Start.

In the 1’s counter example, there are only 2 status signals:

- Data_{LSB}
- Data \neq 0

Each of these status signals is a single bit: Data_{LSB} is the least significant bit of variable Data, and Data \neq 0 is a 1-bit signal which is the output result of the comparison of Data and 0.

Note that START is not a status signal: it is an external control input coming from the environment. Of course, it is also an input to the controller.
**Step #5. Draw Final Microarchitecture**

The microarchitecture is the final set of datapath and control blocks, showing all their top-level structural interconnections.

In Step #3, we allocated 4 datapath blocks, and we determined all “hardwired inputs.”

In Step #4, we identified all “status signals” (i.e. outputs of the datapath block which are inputs to the control block.)

Finally, we assemble the pieces:
Step #6. Derive Controller Specification: Translate “Generalized ASM” to “Control ASM”

All that remains is to specify and design the control unit (FSM). The idea is simple: we take the generalized ASM, and replace each datapath operation by the control signals that initiate the corresponding datapath operation.

In a Moore generalized ASM, datapath operations are listed in state boxes. So, we simply modify the state boxes, as follows. (Refer to Step #3 for the control truth tables for each allocated datapath block.)

Note: all “external inputs” in the generalized ASM (e.g. “Done” asserted to 1) are unmodified, and simply copied into the final control ASM.

Translation Step: datapath operation (generalized ASM) \(\rightarrow\) control signals (control ASM)

<table>
<thead>
<tr>
<th>State</th>
<th>RTL Datapath Operation</th>
<th>Allocated Component</th>
<th>Mode</th>
<th>Corresponding Control Signals</th>
<th>Final ASM State</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Data:=Input</td>
<td>Shift Reg w/PL</td>
<td>LD (load)</td>
<td>S1 S0 = 01</td>
<td>S1 S0 = 01</td>
</tr>
<tr>
<td></td>
<td>Ocount:=0</td>
<td>Up/Down Ctr w/PL</td>
<td>load</td>
<td>LOAD =1 [E=X (DC)]</td>
<td>LOAD = 1</td>
</tr>
</tbody>
</table>
| S3    | Ocount:=Ocount + 1     | Up/Down Ctr w/PL    | INC 1 | LOAD = 0  
E = 1 | LOAD E = 0 1 |
| S4    | Data:=Data >> 1        | Shift Reg w/PL      | sh right 1 | S1 S0 = 11                   | S1 S0 = 11     |
|       | Compare(Data, 0)       | Up/Down Ctr w/PL    | compare-to-0 | (no enable signals needed: compare always computed) |
| S5    | Output:=Ocount*        | Tri-State Buffer    | enable | (no separate OE, re-uses “Done” as tri-state buffer enable) | Done (=1) [control output] |

*[note: we only list datapath operations here]*
Here is the final ("B/V"-style) ASM for the controller. All datapath operations have been replaced by control signals. Also, external control inputs (e.g. Done) are also added.
Step #7. Translate Control ASM to Symbolic State Diagram (FSM Specification)

The control ASM of Step #6 can be directly translated into a Moore state diagram, following the method presented in “B/V” book and class lectures.

Note: Remember that the control ASM implies certain inputs/outputs that are not explicit! (to avoid clutter).

So, you need to restore those signals, making them explicit in the symbolic state diagram:

Inputs: On each arc in the Moore state diagram, the values of all inputs must be listed: Start, Data ≠ 0, Data_{LSB}.

Outputs: Within each Moore state, the values of all outputs must be listed (see the microarchitecture diagram): S1, S0, E, Load, Done.

Step #8. Synthesize FSM

Follow steps in class lectures and reading (including B/V book). For one final version of the implementation, see Gajski, ch. 8.3, fig. 8.8(a).
RTL Design Flow: Summary of Steps

0. Verbal Description of Algorithm (behavioral)
1. Write Pseudo-Code of Algorithm (behavioral)
2. Write RTL Specification: Generalized ASM (RTL)
3. Allocate (Select) Datapath Blocks (and set hardwired inputs + other optimizations)
4. Identify Status Signals
5. Draw Final Microarchitecture (RTL)
7. Generate Symbolic State Diagram = Control FSM Specification
8. Synthesize Controller (FSM)

Note: A Mealy version of this example is also presented in Gajski, ch. 8.3.