RTL Design Method

### Slide 1: RTL Design Method Examples

**Digital Design 5.3**

**RTL Design Method**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Capture a high-level state machine.</strong></td>
<td>Describe the system’s desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is “high-level” because the transition conditions and the state actions are more than just Boolean operations on bit inputs and outputs.</td>
</tr>
<tr>
<td>2. <strong>Create a datapath.</strong></td>
<td>Create a datapath to carry out the data operations of the high-level state machine.</td>
</tr>
<tr>
<td>3. <strong>Connect the datapath to a controller.</strong></td>
<td>Connect the datapath to a controller block. Connect external Boolean inputs and outputs to the controller block.</td>
</tr>
<tr>
<td>4. <strong>Derive the controller’s FSM.</strong></td>
<td>Convert the high-level state machine to a finite-state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath.</td>
</tr>
</tbody>
</table>

### Slide 2: RTL Example: Video Compression – Sum of Absolute Differences

**Video Compression**

- **Sum of Absolute Differences**

**Video is a series of frames (e.g., 30 per second)**
- **Most frames similar to previous frame**
  - **Compression idea:** just send difference from previous frame

---

**Slide 3: Video Compression - Digitized Frames**

- Digitized frame 1
- Digitized frame 2
- Frame 1
- Frame 2

**Difference of 2 from 1**

- 0.01 Mbyte

**Just send difference**
RTL Design
RTL Example: Video Compression – Sum of Absolute Differences

- Need to quickly determine whether two frames are similar enough to just send difference for second frame
  - Compare corresponding 16x16 "blocks"
  - Treat 16x16 block as 256-byte array
  - Compute the absolute value of the difference of each array item
  - Sum those differences – if above a threshold, send complete frame for second frame; if below, can use difference method (using another technique, not described)

- Steps:
  - S0: wait for go
  - S1: initialize sum and index
  - S2: check if done (i>=256)
  - S3: add difference to sum, increment index
  - S4: done, write to output sad_reg

- Datapath:
  - Inputs: A, B (256 byte memory); go (bit)
  - Outputs: sad (32 bits)
  - Local registers: sum, sad_reg (32 bits); i (9 bits)
  - When go=1, sums the differences of element pairs in arrays A and B, outputs that sum
  - Step 2: Create datapath
RTL Design

RTL Example: Video Compression – Sum of Absolute Differences

- Step 3: Connect to controller
- Step 4: Replace high-level state machine by FSM

```
S0
S1
S2
S3
S4

sum=sum+abs(A[i]-B[i])
```

\[ i<256 \]
\[ i=i+1 \]
\[ \text{sad}_\text{reg}=\sum \]

Controller

```
\text{sum} \leftarrow 0
\text{i} \leftarrow 0
```

```
\text{goto S0, i<256}
```

```
\text{goto S1, i<256}
```

```
\text{goto S2, i<256}
```

```
\text{goto S3, i<256}
```

```
\text{goto S4, i<256}
```

```
\text{AB}_\text{addr}
\text{A_data}
\text{B_data}
```

```
\text{sum}_\text{ld}=1; \text{AB}_\text{rd}=1
\text{sad}_\text{reg}_\text{ld}=1
```

```
\text{i_inc}=1
\text{i_clr}=1
```

```
\text{R} = 10
\text{R} = 11
\text{R} = 12
\text{R} = 17
\text{R} = 20
```

```
\text{Inputs:} \text{D} (8-bits), \text{Q} (8-bits)
\text{Local Registers:} \text{R} (8-bits)
```

```
\text{RTL Design}
\text{Pitfalls and Good Practice}

- Considering the high-level state machine shown to the right, what is the final value of D in state F?
  1. 10
  2. 11
  3. 12
  4. 17
  5. 20
```

```
\text{RTL Design}
\text{Pitfalls and Good Practice}

- Common pitfalls: Assuming register is update in the state it’s written
  - All registers updates in each state will happen simultaneously
    - On the next rising clock edge
  - Consider the FSM to the right:
    - What is the final value of Q?
    - What is the final state?
    - Answer:
      - Value of Q unknown
      - Final state is C (not D)
    - Why?
      - State A: \( R=99 \) and \( Q=R \) happen simultaneously
      - State B: \( R \) not updated with \( R+1 \) until next clock cycle, simultaneously with state register being updated

```

```
\text{RTL Design}
\text{Pitfalls and Good Practice}

- Comparing software and custom circuit SAD
  - Circuit: Two states (S2 & S3) for each \( i \), 256 \( i \)’s \( \rightarrow \) 512 clock cycles
  - Software: Loop (for \( i=1 \) to 256), but for each \( i \), must move memory to local registers, subtract, compute absolute value, add to sum, increment \( i \) – say about 6 cycles per array item \( \rightarrow 256*6 = 1536 \) cycles
  - Circuit is about 3 times (300%) faster
  - Later, we’ll see how to build SAD circuit that is even faster
```

```
\text{RTL Design}
\text{Pitfalls and Good Practice}

- Comparing software and custom circuit SAD
  - Circuit: Two states (S2 & S3) for each \( i \), 256 \( i \)’s \( \rightarrow \) 512 clock cycles
  - Software: Loop (for \( i=1 \) to 256), but for each \( i \), must move memory to local registers, subtract, compute absolute value, add to sum, increment \( i \) – say about 6 cycles per array item \( \rightarrow 256*6 = 1536 \) cycles
  - Circuit is about 3 times (300%) faster
  - Later, we’ll see how to build SAD circuit that is even faster
```
### RTL Design

**Pitfalls and Good Practice**

- **Considering the high-level state machine shown to the right, what is the final value of D in state F?**
  1. 10
  2. 11
  3. 12
  4. 17
  5. 20

- **Common pitfall:** Reading outputs
  - Outputs should only be written
  - Solution: Introduce additional register, which can be written and read

- **Good practice:** Register all data outputs
  - In fig (a), output P would show spurious values as addition computes
  - Furthermore, longest register-to-register path, which determines clock period, is not known until that output is connected to another component
  - In fig (b), spurious outputs reduced, and longest register-to-register path is clear

---

**Diagram**

- **Inputs:** A, B (8 bits)
- **Outputs:** P (8 bits)
- **Local register:** R (8 bits)

---

**Diagram**

- **Inputs:** A, B (8 bits)
- **Outputs:** P (8 bits)
- **Local register:** R (8 bits)