**HW/SW codeign: Design of complex systems incorporating software executing on a processor and dedicated hardware processing components (referred to as reprocessors).**

**Overview of Design Flow (one method):**

1. Initial design (SW, HW, SW+HW)
   - couple/synthesize
   - HW/SW design
     - Simulate/estimate
   - Next step? yes/no
   - Final HW/SW
     - go to lower-level system blocks

**HW/SW Partitioning:** Partition an application (typically SW) into a HW part and a SW part.
- Several possible design implementations

1. Loosely coupled coprocessor

   - simple communication: shared memory
   - coprocessor has direct access to memory
   - communication overhead between IP and coprocessor
2. **Tightly-Coupled Coprocessor**

- Direct communication between \( \text{up} \) and coprocessor
- \( \text{up} \) has access to data through \( \text{coproc.} \)

3. **Instruction Set Extendible Processor**
   (custom ASIC)

- No communication, coprocessors tightly integrated with \( \text{up} \) as instructions
- Only has access to data in register file
- Instructions may affect processor performance (e.g., frequency, pipeline stall)
Example: beer application

1. Profile: Total Instructions: 9,112,159
   Main Loop: 4,182,000 (99.5% of exec. time)
   CPI: 1243

2. List to partition to HW:
   Choose innermost loop (common approach)

3. Partition to HW (PSMO = Faith Stilk Module with Dale)

   Focus on algorithm (not memory and comm)

   \[ \text{PSMO:} \]
   - same semantics as FSM
     but can include disk
     in the form of local
     registers and multi-bit
     registers
   - transition conditions
     can be complex logical
     expressions
   - must distinguish between
     current value and
     value being assigned
     (register and multi-bit
     output not updated
     until next rising clock
     edge)
   - can use newly and \( \text{new} \)
Memory Interface:

- Access Port Types: Read Port

  ![Diagram of Read Port]

- Write Port:

  ![Diagram of Write Port]

- Read/Write Port:

  ![Diagram of Read/Write Port]

- Access Time: single-cycle, multi-cycle, read vs. write times
- Access Modes: Direct access takes more time compared to sequential access
- Related to bus protocol but can be independent

Modify F5P5 to use memory Interface:

![Diagram of modified F5P5]

Communications: What do we need to communicate?

- 5 bits address
- A base address
- Start signal (need to transfer control of bus to new coprocessor)
4. Modify SW to communicate with HW

```c
for (i=0; i < loops; i++)
    for (x=0; x < 8; x++)
        hw (i=x; x=x+1; x=x+1) {
            hw_spintr = s_base+008
            hw_dptr = d_base+008
            hw_startptr = 1;
            wait_for_int(1);
        }
```

3. Estimate Performance

- Time_sw = #instructions * CPI
- Time_hw/sw = Time_sw - Time_sw(loop) + Time_hw * Time_comm
- Time_HW = Execs * Cycles_exec
- Time_comm = Execs * (Cycles_init + Cycles_sync)
\[ \text{Time}_{sw} = 4,152,458 \times 1.2493 = 5,161,133 \]

\[ \text{Time}_{nw} = \text{Time}_{sw} - \text{Time}_{sw}(\text{loop}) \times \text{Time}_{nw} + \text{Time}_{conn} \]

\[ = 5,161,133 - (4,152,458 \times 1.2493) \times \text{Time}_{nw} + \text{Time}_{conn} \]

\[ (5,161,133) \]

\[ = 2,5057 \times \text{Time}_{nw} \times \text{Time}_{conn} \]

\[ \text{Time}_{nw} = 16,000 \times (1 + 4\% - 1) \]

\[ = 16,000 \times 10 \]

\[ = 160,000 \]

\[ \text{Time}_{conn} = 16,000 \times (3 + 0) \text{ Kushi Cycles already included in Time}_{nw} \]

\[ = 48,000 \]

\[ = 2,5057 \times 160,000 + 48,000 \]

\[ = 233,057 \]

\[ \text{Speedup}_{nw} = \frac{\text{Time}_{sw}}{\text{Time}_{nw}} = \frac{5,161,133}{233,057} = 22.1 \times \]

5. Does this meet our goals?