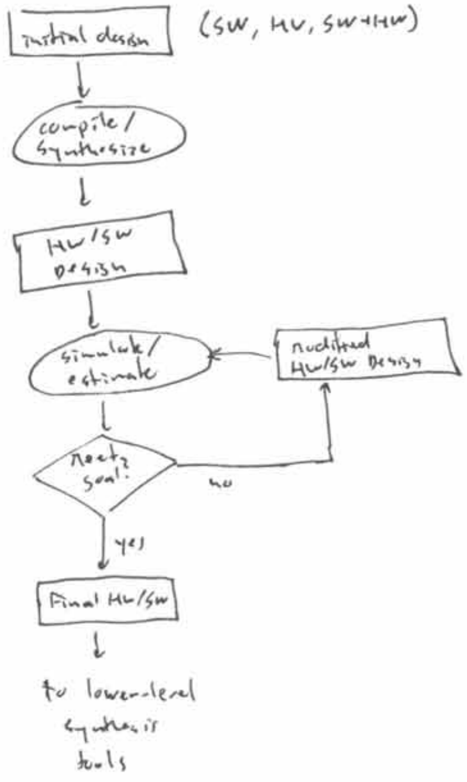
HW/SW codesign: Design of complex systems incorporating software executing on a processor and dedicated hardware processing components (referred to as coprocessors)

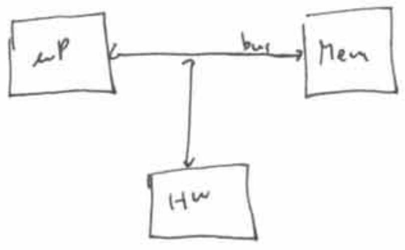
Overview of Design Flow (one method):



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

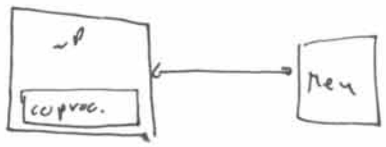
1. Loosely coupled coprocessor

+ simple communication: shared memory
+ coprocessor has direct access to memory
- communication overhead between µP and coprocessor

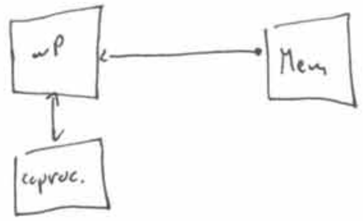


2. Tightly-Coupled Coprocessor

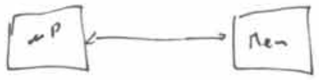
+ direct communication between uP and coprocessor
- only has access to data through uP



or



3. Instruction Set Extensible Processor
(custom ASIP)



+ no communication, coprocessors directly integrated w/in uP as instructions

- only has access to data in register file
- instructions may affect processor performance (e.g. frequency, pipeline stages)

Example: brev application

1. Profile: Total Instructions: 4,152,159

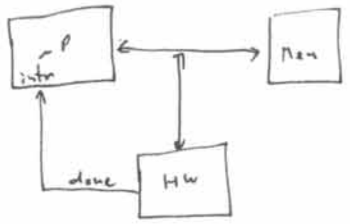
   Main Loop: 4,132,000 (99.5% of exec. time)

   CPI: 1.243

2. What to partition to HW:

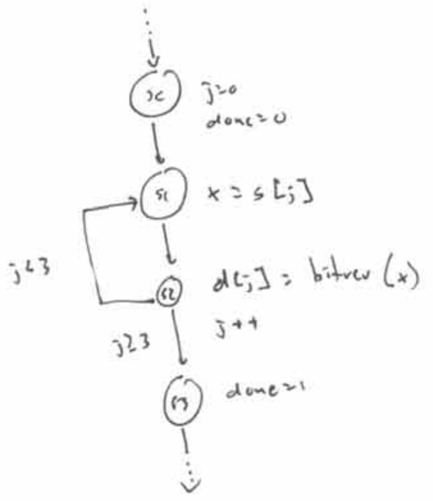
   choose innermost loop (common approach)

3. Partition to HW (FSMD - Finite State Machine with Data)



- Processor and HW have same operating frequency
- single cycle memory access
- one read and one write port (independent)

Focus on algorithm (not memory and comm.)

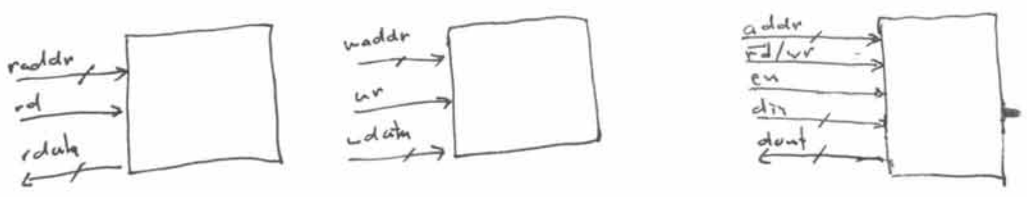


FSMD:

- same semantics as FSM but can include data in the form of local registers and multibit IO
- transition conditions can be complex logical expressions
- must distinguish between current value and value being assigned (registers and multibit output not updated until next rising clock edge)
- can mix Mealy and Moore

## Memory Interface:

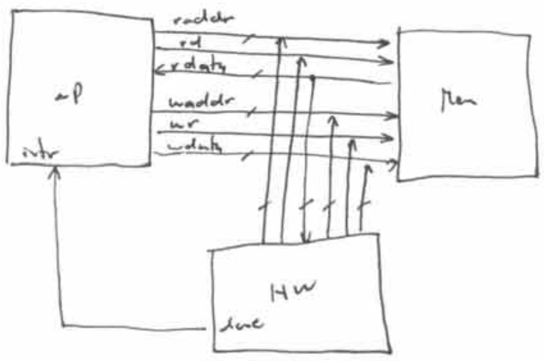
- Access Port Types: Read Port          Write Port:          Read/write Port:

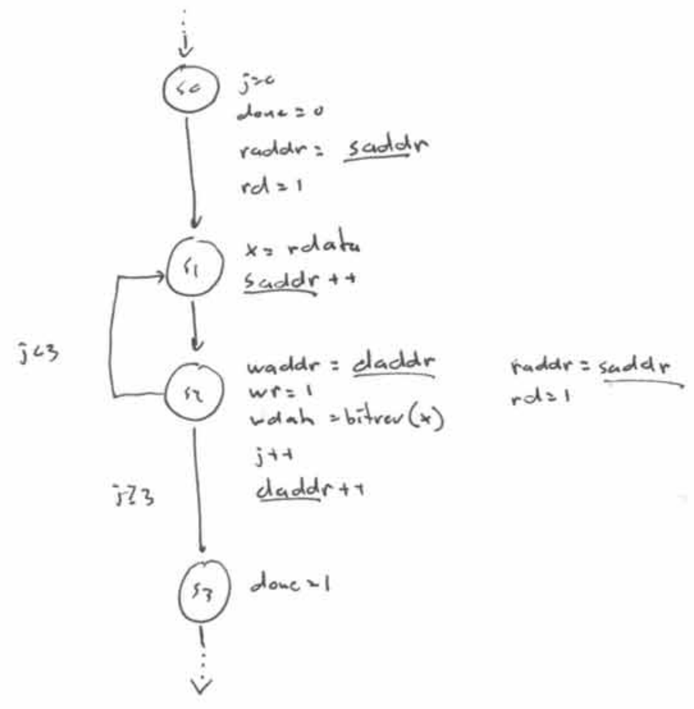


- Access Time: single-cycle, multi-cycle, read vs. write times
- Burst modes: First access takes more time compared to sequential accesses

- related to bus protocol but can be independent

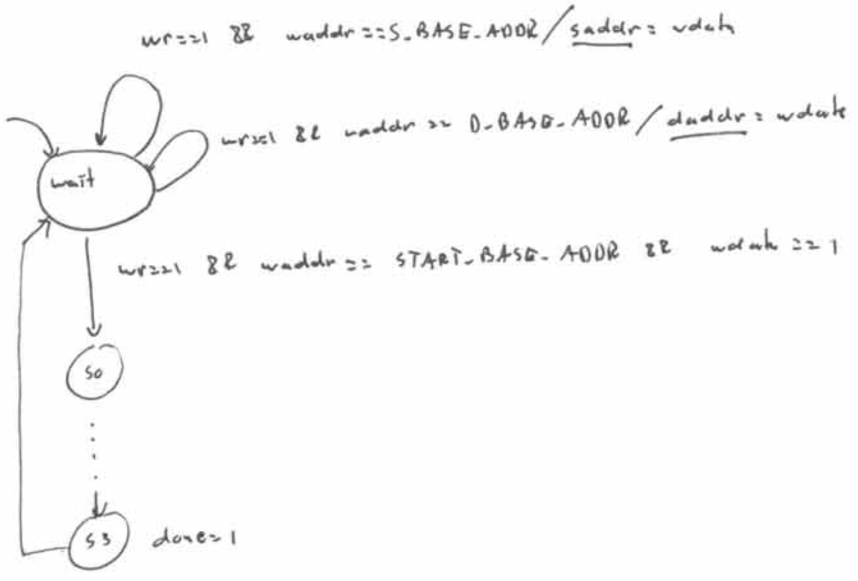
Modify FSMD to use memory interface:





communication: what do we need to communicate?

- s base addr
- d base addr
- start signal (need to transfer control of bus to HW coprocessor)


wr==1 && waddr==S_BASE_ADDR / saddr = wdata
wr==1 && waddr == D_BASE_ADDR / daddr = wdata
wait
wr==1 && waddr == START_BASE_ADDR && wdata == 1
S0
S3
done=1

4. Modify SW to communicate with HW

```
for (i=0; i < Loops; i++)
    for (s=src, d=dst; *s; s+=4, d+=4)
        for (j=0; j<4; j++) {
            *hw_sptr= s;
            *hw_dptr = d;
            *hw_startptr = 1;
            waitForIntr();
        }
```

```
hw_sptr      = S_BASE_ADDR
hw_dptr      = D_BASE_ADDR
hw_startptr  = START_BASE_ADDR
```

3. Estimate Performance

$\text{Time}_{sw} = \#\text{instructions} * \text{CPI}$

$\text{Time}_{HW/SW} = \text{Time}_{sw} - \text{Time}_{sw(Loop)} + \text{Time}_{HW} + \text{Time}_{comm}$

$\text{Time}_{HW} = \text{Execs.} * \text{Cycles}_{Exec.}$

$\text{Time}_{comm} = \text{Execs.} * (\text{Cycles}_{init} + \text{Cycles}_{sync.})$

$\text{Time}_{sw} = 4{,}152{,}159 * 1.2432 \quad 5{,}161{,}133$

$\text{Time}_{HW/SW} = \text{Time}_{sw} - \text{Time}_{sw(loop)} + \text{Time}_{HW} + \text{Time}_{comm}$

$= 5{,}161{,}133 - (4{,}132{,}000 * 1.243) + \text{Time}_{HW} + \text{Time}_{comm}$

$(5{,}136{,}076)$

$= 25{,}057 + \text{Time}_{HW} + \text{Time}_{comm}$

$\text{Time}_{HW} = 16{,}000 * (1 + 4*2 + 1)$

$= 16{,}000 * 10$

$= 160000$

$= 25{,}057 + 160{,}000 + \text{Time}_{comm}$

$\text{Time}_{comm} = 16{,}000 * (3 + 0)$

$= 48{,}000$

Note: $\text{Cycles}_{sync}$ already included in $\text{Time}_{HW}$

$= 25{,}057 + 160{,}000 + 48{,}000$

$= 233{,}057$

$$\text{Speedup}_{HW/SW} = \frac{\text{Time}_{sw}}{\text{Time}_{HW}} = \frac{5{,}161{,}133}{233{,}057} = \boxed{22.1\times}$$

5. Does this meet our goals?