IIT JODHPUR

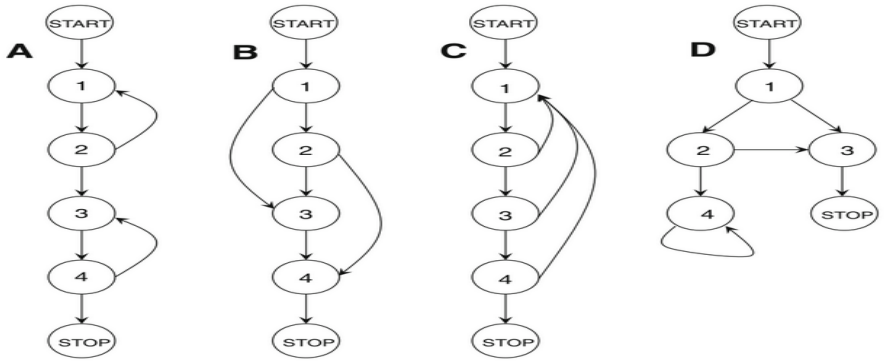
Major Examination: EEL7210 Hardware Software Co-design [OPEN-BOOK] (Apr'25)

Guidelines (Total time: 180 minutes, Maximum Marks: 40):

- Please read the question paper very carefully and answer-to-the-point ONLY.
- **NO clarification is required in any question.** In case of any doubt, assume whatever you wish to and state that in your answer. Step-wise marks would be awarded wherever applicable.

---

1. Consider the following 4 different control flow graphs (CFG). Design the software (code) in C/C++ that implement these respective CFGs and measure your design (i.e., code) performance? [8]



2. Bresenham's line algorithm is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is commonly used to draw line primitives in a bitmap image (e.g. on a computer screen), as it uses only integer addition, subtraction, and bit shifting, all of which are very cheap operations.

```
1  void bresen(unsigned tx, unsigned ty) {
2    unsigned char  x, y;    // pixel coordinates
3    unsigned char  e;       // error accumulator
4    unsigned char  es, ed; // error inc for straight/diag
5    unsigned char  xs, xd; // x inc for straight/diag
6    unsigned char  ys, yd; // y inc for straight/diag
7    unsigned i;
8
9    x  = 0; y = 0;
10   e  = 0;
11   ed = (tx > ty) ? (ty - tx) : (tx - ty);
12   es = (tx > ty) ? ty        : tx;
13   xd = 1;
14   xs = (tx > ty) ? 1 : 0;
15   yd = 1;
16   ys = (tx > ty) ? 0 : 1;
17
18   for (i=0; i<64; i++) {  // plot 64 pixels
19     // plot(x, y);
20     x = (e & 0x80) ? x + xs : x + xd;
21     y = (e & 0x80) ? y + ys : y + yd;
22     e = (e & 0x80) ? e + es : e + ed;
23   }
24 }
```

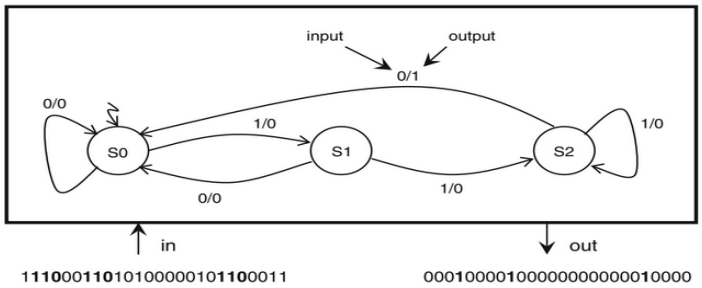
   Design the hardware equivalent of Bresenham's line algorithm for faster processing? Analyze and optimize your design for overall latency and resource consumption? [5+3]

3. In a System-on-Chip (SoC) design for an embedded image processing application, you have the following components (interconnected using AXI to handle communication) :

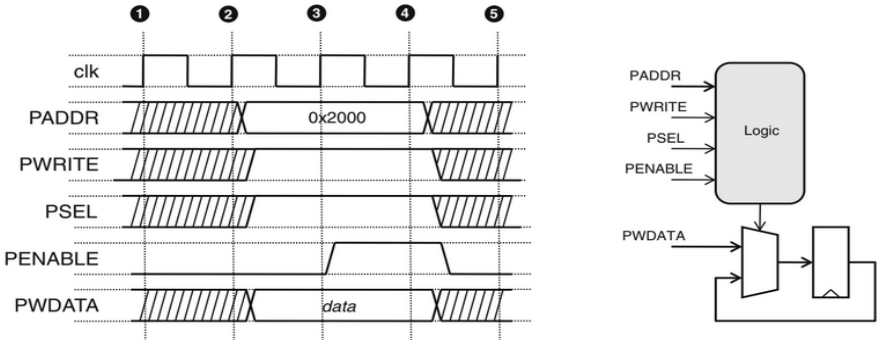
   - An ARM Cortex-A53 processor that acts as the AXI master for general control tasks.
   - A DMA controller, also an AXI master, that streams image data directly from memory to peripherals.
   - A DDR4 memory controller and an image processing accelerator, both acting as AXI slaves.

   It is quite obvious that the processor and DMA often need to access the DDR memory at the same time, and the accelerator also receives large chunks of data from memory via the DMA. Explain how the AXI Interconnect manages simultaneous memory access requests from both the processor and the DMA controller to the DDR4 memory. With proper calculations (wherever possible), provide an explanation as to what mechanisms are involved in ensuring efficient data throughput, and how does the interconnect handle arbitration and potential data bottlenecks? [5]

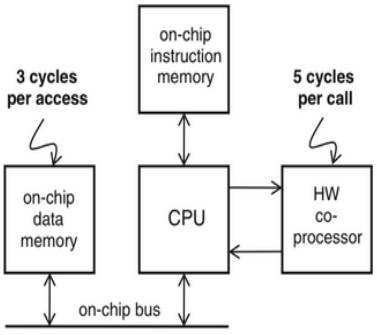
4. You are responsible for designing the authenticator hardware for an access control mechanism (ACM) in an IoT-based factory automation system. An intern working in the company comes up with the below FSM for the authenticator hardware. The intern forgot to tell you the 3-bit password before leaving the company. Given the below Mealy FSM, guess the 3-bit password for the ACM? How? (Hint: Whenever this password is provided, system output is 1, i.e., ACM provides access to the user.) Also, design the authenticator hardware for ensuring nearly real-time system performance? [3+4]



5. Below timing diagram illustrates a write operation on the AMBA peripheral bus, AMBA APB. A memory-mapped register is a register which is able to intercept bus transfers from a specific address. In this case, we wish to create logic which will write PWDATA into a register when a write to address 0x2000 occurs. Develop a logic expression for the logic module as shown below? [5]



6. The system-on-chip in the below Figure combines a coprocessor, a processor, and on-chip data-memory and instruction-memory. The processor will copy each element of an array a[ ] to the coprocessor, each time storing the result as an element of an array r[ ]. The C program that achieves this is shown on the right of the below Figure. All the operations in this architecture take zero time to execute, apart from the following two operations: accessing the on-chip data memory takes three cycles, and processing a data item on the coprocessor takes five cycles.




```
int a[1000], r[1000];
void transform() {
  int i;
  int *ra = a;
  int *rr = r;
  for (i=0; i<1000; i++)
    *rr++ = call_coproc(*ra++);
}
```

- (a) Find the resulting execution time of the C program that is shown above. [2]
- (b) Show how you can rewrite the C program so that the resulting execution time becomes smaller than 8,000 cycles. As a hint, assume that you can rewrite the y = call_coproc(x); function call as two separate calls. The first call is send_coproc(x);, and it issues an argument from software to the coprocessor. The second call is y = get_coproc();, and it takes a result from the coprocessor. Compare the execution time of the rewritten C program compared to the answer in (a)? [3+2]