IIT JODHPUR

Minor Examination: EEL7860 High-level Synthesis [OPEN-BOOK] (May'25)

Guidelines (Total time: 120 minutes, Maximum Marks: 25):

- 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.
- Usage of LLM/Internet is NOT allowed.

---

1. In the figure shown below, two probable implementations of the function C = AB + C are shown where the loop iterations are treated in different ways (unfolding/unrolling, pipelining etc.). By inspecting the design code shown below, suggest the advantages/disadvantages of (b) over (a)? With appropriate diagrams/representations, explain the synthesis outputs in the respective cases? [6 Marks]

```
1 for (int n = 0; n < N; ++n)
2   for (int m = 0; m < M; ++m) {
3     double acc = C[n][m];
4     #pragma PIPELINE
5     for (int k = 0; k < K; ++k)
6       acc += A[n][k] * B[k][m];
7     C[n][m] = acc; }
```

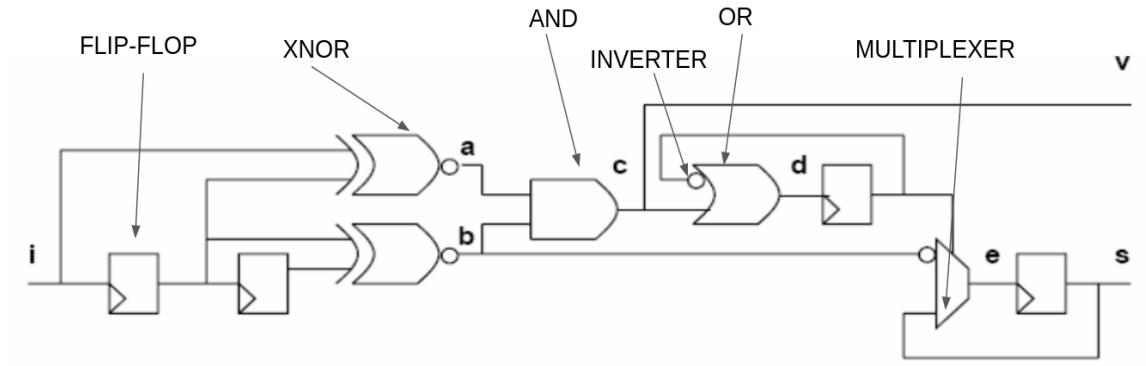
(a) Naive implementation of general matrix multiplication

```
1  for (int n = 0; n < N; ++n)
2    for (int m = 0; m < M/T; ++m) {
3      double acc[T]; // Tiles of size T
4      for (int k = 0; k < K; ++k)
5        double a = A[n][k]; // M/T reads
6        #pragma PIPELINE
7        for (int t = 0; t < T; ++t) {
8          double prev = (k == 0) ?
9              C[n][m*T+t] : acc[t];
10         acc[t] = prev + a * B[k][m*T+t]; }
11     for (int t = 0; t < T; ++t) // Write
12       C[n][m*T+t] = acc[t]; }   // out
```

(b) Tiled iteration (looping) space, same location written every T cycles

Figure 1: Multiply-Addition Implementation Methods

2. Below is an implementation of datapath of a design that takes an input (**i**) and generates two outputs (start, **s** and valid, **v**). What are the opportunities to optimize the latency of this design? [4 Marks]

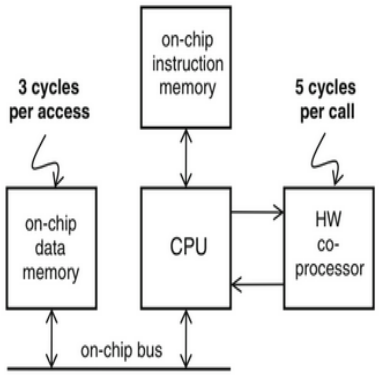


3. Write a C program to check if a number is odd/even? Now, perform the high-level synthesis of this program to obtain an operator graph with any scheduling mechanism that you are aware of? [5 Marks]

4. 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. Design the hardware equivalent of Bresenham's line algorithm for faster processing? Analyze and optimize your design for overall latency and resource consumption? [3+2]

```
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 }
```

5. The system-on-chip (SoC) 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 the 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]
