IIT JODHPUR

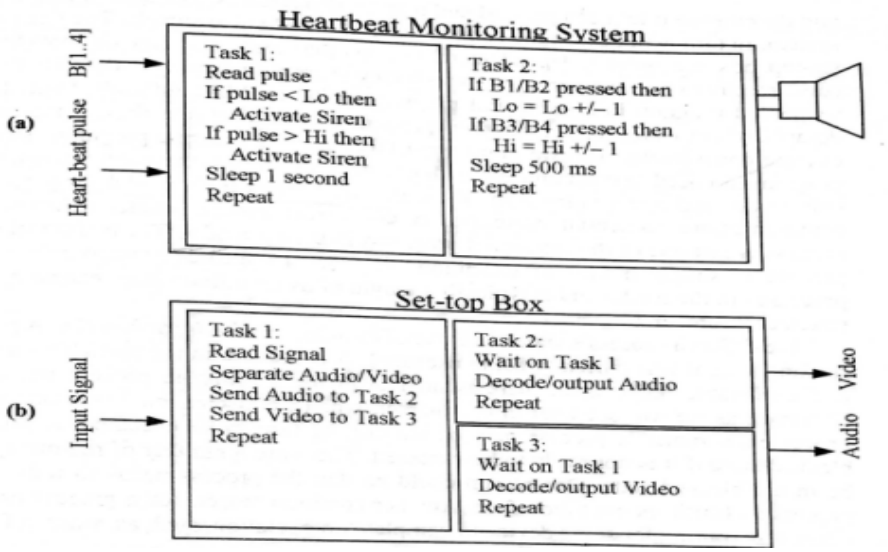
Major Examination: EEL7860 High-level Synthesis [OPEN-BOOK] (June'25)

Guidelines (Total time: 180 minutes, Maximum Marks: 45):

- 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. High-level synthesis approach is considered to be a good methodology for designing embedded systems. Below are two examples of real-life embedded systems. The procedure of the operation of the systems are also mentioned. Develop the representation of design in C/C++ Or SystemC language. Now convert the design in the hardware form through the step of allocation, binding and scheduling? [8 + 8 marks]



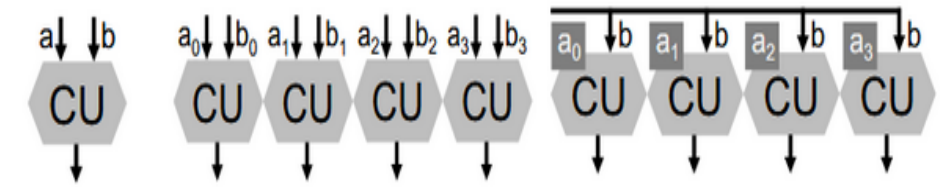
2. Below code is a fragment of the design description of a digital signal processing (DSP) sub-module. Analyze the impact of pragmas during the HLS stage for the below code fragment? [Ignore the pragma unroll=fold variant of the directive i.e., consider pragma unroll as 0/partial/all ONLY] [7 marks]

```
1  int data[16]/*pragma array=REG|RAM*/;
2
3  int ave16(int data_new){
4  int sum=0, x
5  /* pragma unroll=0|partial|all|fold*/
6  for(x=15;x>0;x--)
7       data[x]=data[x-1];
8  data[0]= data_new;
9  /* pragma unroll=0|partial|all|fold*/
10 for(x=0;x < 16; x++)
11      sum =sum+data[x];
12 return(sum/16);
13 }
```

3. Suppose the goal of a design is to perform some encryption-related task. One of the step in this process is to do multiplication with prime numbers. However, the prime number identification has to be done in an on-chip manner i.e., it has to be computed in the hardware. Therefore, you can assume that a random stream of numbers is input to your system and your design should perform the detection of prime numbers from this stream. With the help of HLS-based approach, develop the complete hardware design (datapath + controller) description for performing the prime number identification following a control flow graph (CFG) and data flow graph (DFG) computation approach? [8 marks]

4. You are employed in a deep-tech startup named as "BharatSystemDesign". This startup designs a hardware-based obfuscation scheme for credit card security in the following manner. Given the credit card number "CN" is a 9-bit number, it transforms "CN" into "RCN" as follows: if the decimal value of "CN" is odd, cube it else square it. How can we design the authenticator hardware system (through the HLS approach if design is represented in C++ or SystemC) in order to achieve almost real-time performance? Evaluate your design in terms of latency and try to optimize it? [7 marks]

5. We can achieve scalable parallelism in HLS without relying on external memory band-width by exploiting data reuse, distributing input elements to multiple computational units replicated "vertically" through unrolling. Viewed from the paradigm of cached architectures, the opportunity for this transformation arises from temporal locality in loops. Vertical unrolling draws on bandwidth from on-chip fast memory by storing more elements temporally, combining them with new data streamed in from external memory to increase parallelism, allowing more computational units to run in parallel at the expense of buffer space. In comparison, horizontal unrolling requires us to widen the data path that passes through the processing elements (consider the below illustration in this regard- CU stands for computational unit).



(a) Before. (b) Horizontal unrolling. (c) Vertical unrolling.

Taking the relevant assumptions about the values of N,M (matrix size), tiling factor (T) and folding factor (P) and any other parameter, analyze the benefits of vertical pipelining over horizontal pipelining for the below implementation of matrix multiplication? [7 marks]

```
1 for (int n = 0; n < N / P; ++n) { // Divided by unrolling factor P
2   for (int m = 0; m < M / T; ++m) { // Tiling
3     double acc[T][P]; // Is now 2D
4     // ...initialize acc from C...
5     for (int k = 0; k < K; ++k) {
6       double a_buffer[P]; // Buffer multiple elements to combine with
7       #pragma PIPELINE    // incoming values of B in parallel
8       for (int p = 0; p < P; ++p)
9         a_buffer[p] = A[n*P + p][k];
10      #pragma PIPELINE
11      for (int t = 0; t < T; ++t) // Stream tile of B
12        #pragma UNROLL
13        for (int p = 0; p < P; ++p) // P-fold vertical unrolling
14          acc[p] += a_buffer[p] * B[k][m*T+t];
15    }
16    /* ...write back 2D tile of C... */
17  }
18 }
```