Midterm Exam ECE 448
Spring 2020
Tuesday, March 3
15 points

Instructions:
Zip all your deliverables into an archive <last_name>.zip and submit it through Blackboard no later than Tuesday, March 3, 11:40 AM EST.

Textual Description

The EXAM circuit takes as an input a stream of 8-bit ASCII characters representing a document written in English. The end of the document is indicated with NULL (a character with an ASCII code 0x00).
The circuit calculates and reports the number of occurrences of each letter of the Latin alphabet. For the purpose of this calculation, the lower-case and upper-case letters, e.g., 'a' and 'A', are treated the same way. The document is assumed to contain no more than $2^k - 1$ characters.

The circuit is specified below using its:

- Table of input/output ports
- Pseudocode
- Block diagram of the Datapath
- ASM chart of the Controller.

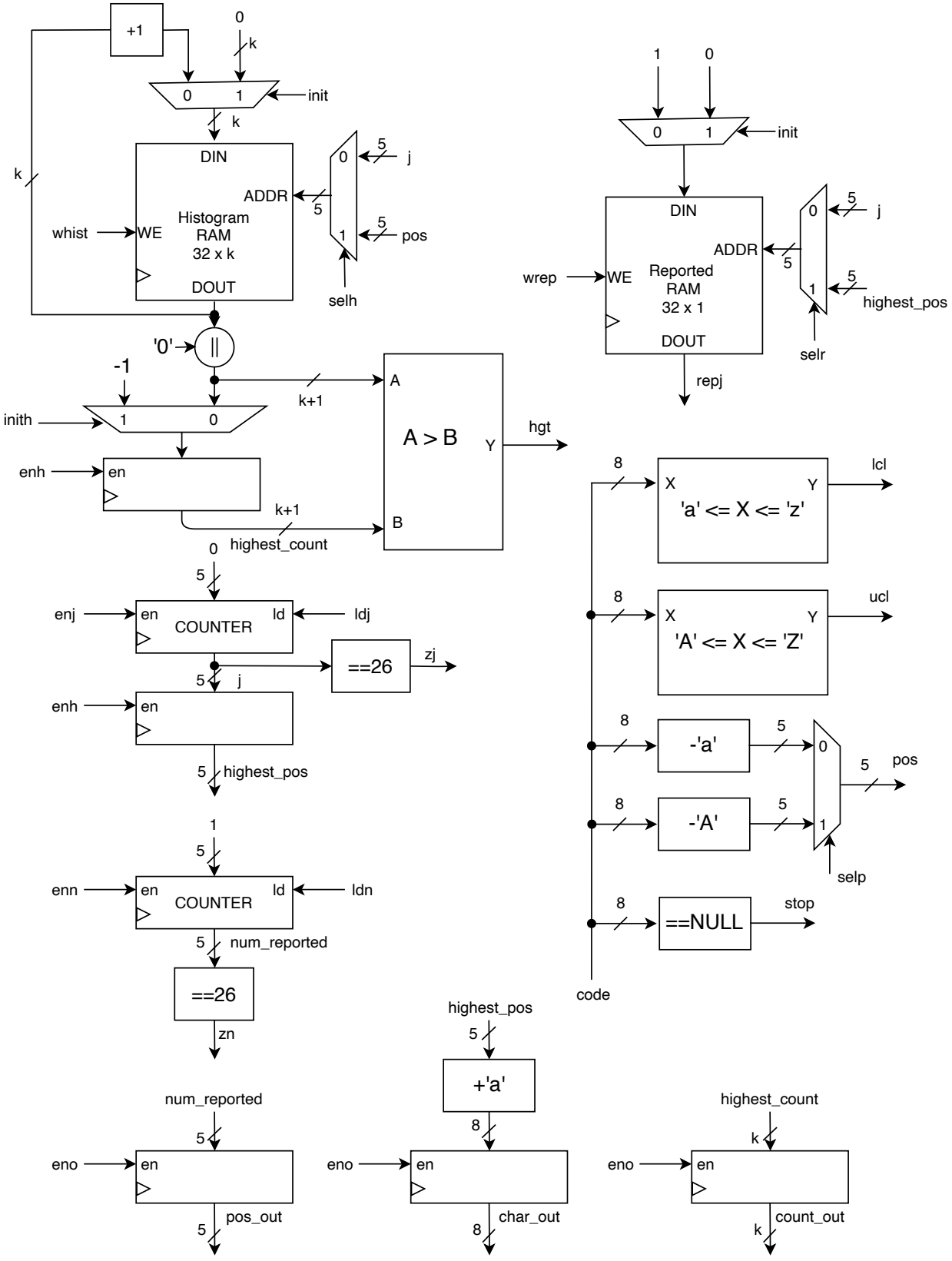
Table of Input/Output Ports

Port	Width	Meaning
clk	1	System clock
rst	1	System reset. Active high.
code	8	Input data bus.
start	1	Control input indicating the start of data processing. The first ASCII code can be read one clock cycle later.
pos_out	5	Position of a letter in the ranking. 1 is used for a letter with the largest number of occurrences; 26 for a letter with the smallest number of occurrences.
char_out	8	ASCII code of the reported character in its lower-case form.
count_out	k	Number of occurrences of the currently reported character char_out.
new	1	Status signal active for one clock cycle after a new position in the ranking is reported.

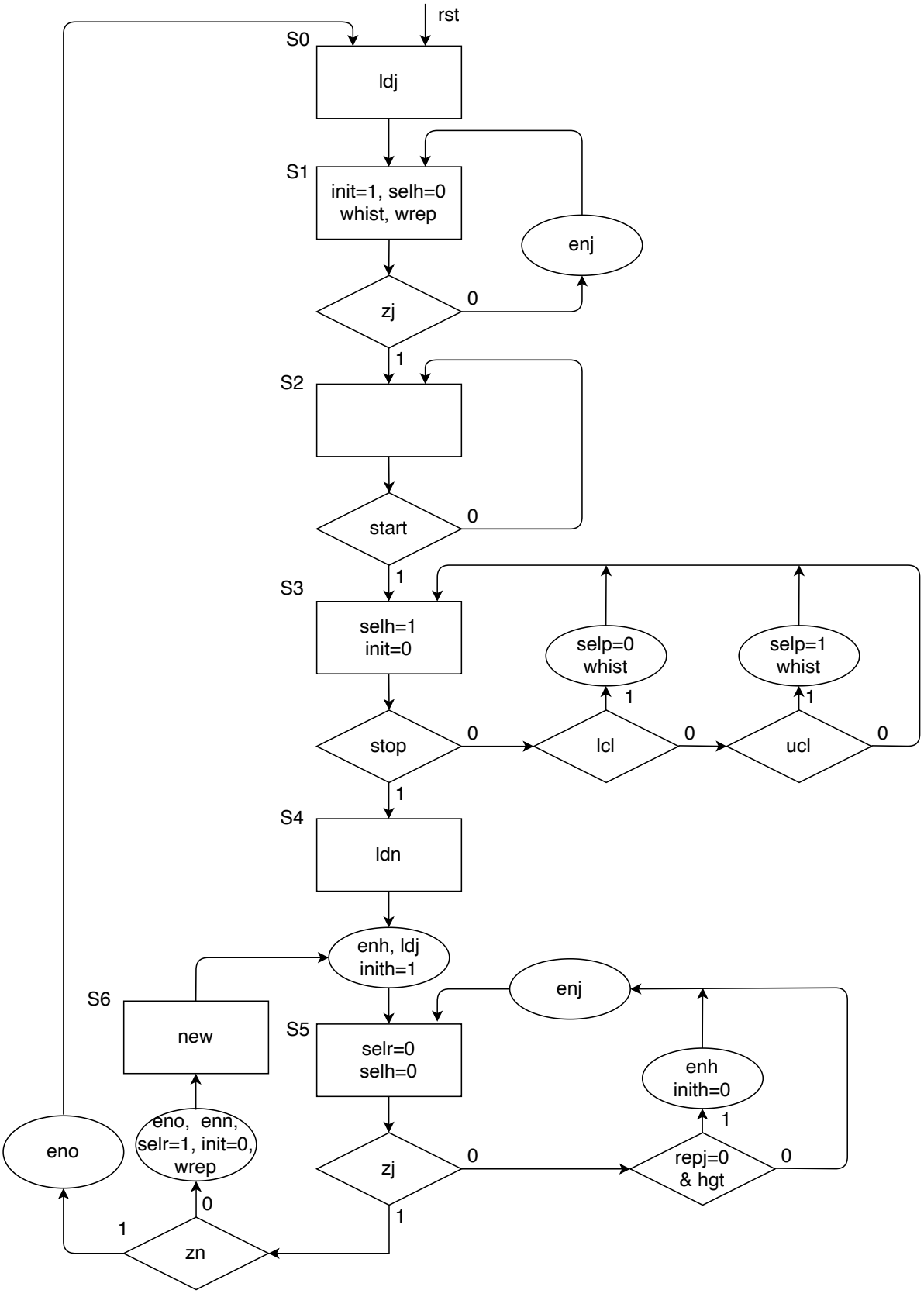
Pseudocode

```
begin:

/* Initializations */

for j=0 to 25
   histogram[j] = 0
   reported[j] = 0
end for

wait for start = '1'

/* Processing input */

while (code != NULL) do
  if ((code >= 'A') and (code <= 'Z')) then
     pos = code - 'A'
     histogram[pos]++
  elsif ((code >= 'a') and (code <= 'z')) then
     pos = code - 'a'
     histogram[pos]++
  endif
end while

/* Post-processing and reading results */

num_reported = 1

while (num_reported < 27)
   highest_count = -1
   for j=0 to 25 do
       if (reported[j]=0) and (histogram[j] > highest_count) then
            highest_count = histogram[j]
            highest_pos = j
       end if
    end for
    pos_out = num_reported
    char_out = highest_pos + 'a'
    count_out = highest_count
    reported[highest_pos]=1
    num_reported++
end while

go to begin
```

Block diagram of the Datapath



ASM Chart


rst
S0
ldj
S1
init=1, selh=0
whist, wrep
enj
zj
0
1
S2
start
0
1
S3
selh=1
init=0
selp=0
whist
selp=1
whist
1
1
stop
0
lcl
0
ucl
0
1
S4
ldn
enh, ldj
inith=1
enj
S6
new
S5
selr=0
selh=0
enh
inith=0
1
eno
eno, enn,
selr=1, init=0,
wrep
zj
0
repj=0
& hgt
0
0
1
1
zn

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Tasks:

Perform the following tasks:

1. Write a synthesizable VHDL code representing the circuit.
2. Write a testbench verifying the operation of your circuit for k=16.
   Assume that the document consists of the following characters followed by NULL:
   "If you tell the truth you do not have to remember anything"
   Hint: In your testbench, define the constant `Text` of the type string, set to the given above sequence of characters, e.g.,
   ```
   constant Text : string := "If you tell the truth you do not have to remember anything";
   ```
   Convert subsequent characters of this text, `Text(i)`, to the corresponding ASCII codes using the following code:
   ```
   code <= STD_LOGIC_VECTOR(TO_UNSIGNED(CHARACTER'POS(Text(i)), 8));
   ```
3. Perform functional simulation of your circuit and use it to debug your VHDL code. Take a printout of the waveform showing the entire operation.
4. Synthesize your circuit.
5. Implement your circuit using
   a. FPGA family: Artix-7
   b. Part name: XC7A35TCPG236-1
   c. Speed Grade: -1
6. Based on the implementation reports, determine the number of CLB slices, LUTs, flip-flops, and pins used by the circuit.

Expected results:

pos_out	char_out	count_out	pos_out	char_out	count_out
1	‘t’	7	14	‘d’	1
2	‘e’	6	15	‘f’	1
3	‘o’	5	16	‘g’	1
4	‘h’	4	17	‘v’	1
5	‘n’	3	18	‘c’	0
6	‘r’	3	19	‘j’	0
7	‘u’	3	20	‘k’	0
8	‘y’	3	21	‘p’	0
9	‘a’	2	22	‘q’	0
10	‘i’	2	23	‘s’	0
11	‘l’	2	24	‘w’	0
12	‘m’	2	25	‘x’	0
13	‘b’	1	26	‘z’	0

Deliverables:

1. VHDL code of your entire circuit.
2. VHDL code of your testbench.
3. Timing waveforms from the functional simulation demonstrating the outputs generated by your circuit.
4. FPGA resource utilization (as defined in Task 6 above).