Examen de Codiseño Hardware/Software **Primera oportunidad. 1 de junio de 2020**
Alumno: Alonso Rodríguez Iglesias

Cada pregunta vale 1 punto sobre un total de 4 puntos del examen

1. Define qué es un ASIP, incidiendo en los detalles que los diferencian de otras opciones tecnológicas.

2. Dado un programa escrito en SystemC con precisión de ciclo de reloj. Indica cual es el nivel de granularidad en la simulación de un método que sólo sea sensible a la subida del flanco de reloj. Indica cual sería el nivel de granularidad si ese método fuese sensible a cambios en más señales. Relaciona tu respuesta con la parte del proyecto sobre el procesador MIPS que has hecho este curso.

3. Indica, para las 7 operaciones siguientes, los conjuntos de operaciones que se pueden realizar de forma concurrente. Por ejemplo: {g, e, i}. Hazlo de forma que ningún conjunto contenga a otro.

   A continuación, indica los conjuntos que se pueden realizar de forma paralela si las unidades funcionales disponibles son 2 sumadores/restadores y un multiplicador/divisor.

```
e = c * d;        f = m * n;        g = a - b;        h = c + b;

i = h + f;        j = g / e;        k = i - j;
```

4. Este es el código ensamblador que tu grupo presentó para el proyecto. Supón que este programa fuese un ejemplo significativo de todos los programas que el procesador tuviese que ejecutar. Se te pide que conviertas el procesador MIPS que hemos utilizado en el proyecto, en un ASIP. Para ello, indica qué características del procesador añadirías o eliminarías que permitieran que el ASIP fuese más rápido o más barato y eficiente.

```
ori $s6, $0, 0x3000
ori $t8, $0, 1
sw $t8, 12($s6)
ori $a0, $0, 0x1000
ori $a1, $0, 0x2000

while_true:
inicioEspera:

lw $t0, 0($s6)
beq $t0, $0, inicioEspera

esperaPosterior:

lw $t0, 8($s6)
beq $t0, $0, esperaPosterior
xor $s0, $s0, $s0

for_i:

ori $s1, $s0, $0
xor $s2, $s2, $s2
xor $s3, $s3, $s3

for_n_1:

sll $s2, $s2, 1
andi $t0, $s1, 1
or  $s2, $s2, $t0
srl $s1, $s1, 1
addi $s3, $s3, 1
addi $t0, $s3, -8
bne $t0, $0, for_n_1
sll $s4, $s2, 1
addi $s1, $s0, 1
xor  $s2, $s2, $s2
xor  $s3, $s3, $s3

for_n_2:

sll $s2, $s2, 1
andi $t0, $s1, 1
or  $s2, $s2, $t0
srl $s1, $s1, 1
addi $s3, $s3, 1
addi $t0, $s3, -8
bne $t0, $0, for_n_2
sll $s5, $s2, 1
sll $s4, $s4, 2
sll $s5, $s5, 2
add $t0, $a0, $s4
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t0, $a0, $s5
lw $t3, 0($t0)
lw $t4, 4($t0)
add $t5, $t1, $t3
sub $t6, $t1, $t3
add $t7, $t2, $t4
sub $t8, $t2, $t4
sll $t0, $s0, 3
add $t0, $t0, $a1
sw $t5, 0($t0)
sw $t7, 4($t0)
sw $t6, 8($t0)
sw $t8, 12($t0)
addi $s0, $s0, 2
addi $t0, $s0, -256
bne $t0, $0, for_i
sw $t8, 4($s6)
sw $t8, 12($s6)
ori $t0, $t0, 0x800
xor  $a0, $a0, $t0
xor  $a1, $a1, $t0
j while_true
```