(202210-IELE2210-SED) PROYECTO Frogger. Grupo 09 Sec1

Contents

ESPECIFICACIONES

Descripción del producto

[Collapse]

PRODUCTO DE CALIDAD: El estudiante identifica las especificaciones y restricciones (enunciadas y no enunciadas del producto) para comprender y garantizar e requerimiento solicitado.

- a. La descripción del producto está redactada en palabras propias de los estudiantes, organizada lógica y claramente.
- b. Las especificaciones y restricciones responden totalmente al producto solicitado, demostrando originalidad y aportes propios. Se propone un macroalgoritmo de operación/funcionamiento del producto que responde totalmente al producto solicitado.
- c. La búsqueda e identificación de características similares de proyectos base utilizados como referencia es clara justificando su utilidad para el desarrollo del proyecto.
- d. El lenguaje disciplinar es preciso y adecuado, las frases son gramaticalmente correctas y no hay errores ortográficos.

El producto es el juego llamado “frogger”, el cual consiste en guiar ranas hasta un punto especifico que es considerado la casa de la rana, para lograrlo el jugador mueve la rana por medio de pulsar botones con dirección de derecha, izquierda, arriba y abajo. Estos botones tienen la intención de que el jugador pueda mover la rana y de esta manera logre evitar chocar con diferentes tipos de obstáculos. Para este proyecto, se va a usar una matriz de leds de 8X8 por lo que las luces encendidas son los obstáculos que el jugador debe esquivar. El frogger o la rana en este caso será también una luz encendida la cual empieza en la primera fila inferior de la matriz, y al pulsar los botones mencionados anteriormente pueda moverse en todas las direcciones, excepto cuando ya está en la primera columna o ultima, de igual manera si se encuentra en la fila inferior, en estos casos si se presiona el botón es cada respectiva dirección la rana no se moverá, ni se perderá en el juego. Además, los obstáculos se moverán y cambiarán su velocidad en cada nivel, haciendo que cada nivel sea más difícil. El juego tendrá 4 niveles, cada uno más difícil que el anterior. Para saber dónde están las casas, habrá leds en la fila superior los cuales estarán apagados, y allí es donde deben llegar los froggers. En el

juego, el usuario sabrá cuando gano y cuando perdió, así como se le informara cuando cambia de nivel. Para todo lo anterior, se tienen unas señales, las cuales son el start, reset, derecha, izquierda, arriba y abajo. Start es para iniciar el juego, reset para vuelve a iniciar el juego y la derecha, izquierda, arriba y abajo son para controlar al frogger.

El juego inicia con el numero 1, luego se presiona start para pasar al nivel. En términos de estados, se esta en 'check' hasta que se pulse un botón para mover la rana, se hace verificación constante si el jugador perdió. Si pierdo se mostrará una cara triste, se tendrá que presionar start para volver a las condiciones iniciales del nivel actual. También se está verificando constantemente si se ganó o se anido. Si se anido, se bloquea el nido y se reinicia el jugador, pero se mantiene el nivel intacto, excepto por la fila de los nidos. Si se llenaron ambos nidos, se gana el nivel y se pasa a la pantalla de carga del siguiente, se tendrá que oprimir start para iniciar ese nivel y así con todos los niveles hasta superar el nivel 4 que mostrara una cara feliz.

Referencias de proyectos de referencia.

1. *Frogger*. Minijuegos. [1] (https://www.minijuegos.com/juego/frogger)

Cronograma de Actividades

[Collapse]

PRODUCTO DE CALIDAD: Presenta un cronograma de actividades y la distribución de tareas entre los integrantes del grupo. Se deben consignar los logros y problemas (RESUMIDOS) presentados día a día en el desarrollo del proyecto. Se debe presentar como una tabla muy resumida.

Parrafo / Tabla: Cronograma de actividades y la distribución de tareas.

Primera entrega	Segunda entrega	Tercera entrega	Entrega Final
1.1 Descripcion del producto.	1.4 Arquitectura del sistema (Circuitos):Diagrama de bloques, señales e Interconexiones.	2.1 Memorias de Calculo.	3.1 Descripcion de lenguaje HW y SW.
1.2 Diagrama de caja negra.	1.5 Arquitectura del Sistema (Harware).	2.2 Definiciones de vectores de prueba y simulaciones parciales de respaldo.	3.2 Funcionalidad global del sistema (Fotos, videos, prototipo).
1.3 Macroalgoritmo general de solucion.		2.3 Indicadores de utilizacion de recursos y rendimiento de los dispositivos utilizados.	3.3 Funcionalidad modular (Videos, fotos, demostracion de moulos del sistema). 4.1 Resultados y lecciones aprendidas. 5.1 Herramientas de Software. 6 Bibliografia.

Para trabajar en el proyecto frogger, los integrantes acordamos reunirnos los dias jueves en la universidad, y los fines de semana de manera virtual, para así avanzar en el proyecto.

Diagrama de Caja Negra

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de realizar un diagrama de caja negra para identificar señales de entrada/salida del producto.

- a. El diagrama de caja negra muestra todas las señales de entrada y salida con sus correspondientes nombres estructurados (In/Out) y tamaños (bit/bus).
- b. El diagrama de caja negra relaciona dicho componente con el diagrama de caracterización (test-bench).

Parrafo 1: Descripción.

Imagen 1: Imagen de diagrama de caja negra.

- Señales de entrada del sistema (INPUTS)
  - *Señal Start:* Inicia el programa.
  - *Señal Reset:* Vuelve a las condiciones iniciales.
  - *Señal Clock:* Reloj del sistema.
  - *Señal Up:* Desplazamiento una casilla arriba.
  - *Señal Down:* Desplazamiento una casilla abajo .
  - *Señal Right:* Desplazamiento una casilla a la derecha .
  - *Señal Leftt:* Desplazamiento una casilla a la izquierda.
- Señales de Salida (OUTPUTS)
  - *Señal Clock:* Relojn del sistema.
  - *Señal Data:* Descripción señal Data.
  - *Señal CS:* Descripción señal CS

Macro-algoritmo general de solución

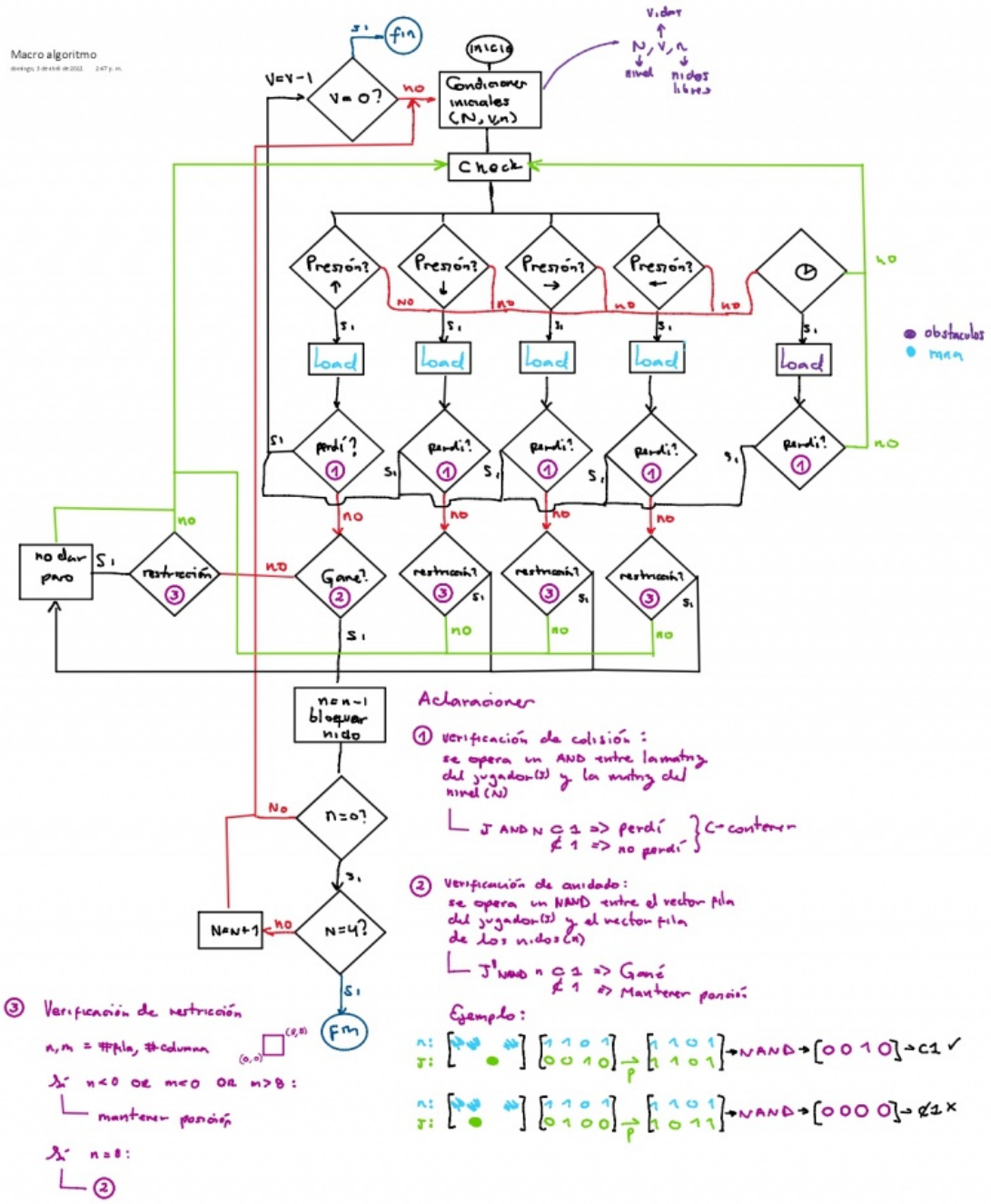
[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de proponer macroalgoritmos descomponiendo el problema en un conjunto de pasos para detallar la funcionalidad esperada del producto.

- a. El macro-algoritmo de solución describe correctamente la funcionalidad del producto y se representa adecuadamente con una explicación detallada en donde cada paso es menos complejo que el producto solicitado.
- b. En cada paso del macro-algoritmo se detalla correctamente un pseudo-algoritmo que describe una posible implementación.

Parrafo 1: Descripción.

Imagen 1: Imagen de diagrama flujo / pasos del macroalgoritmo general de solución.



Parrafo 1: Descripción.

Imagen 1: Imagen de diagrama flujo / pasos del pseudo-algoritmo para una posible implementación de cada paso del macroalgoritmo.

Arq del Sistema (PARTE CIRCUITAL): Diagrama de Bloques, Señales e Inter-conexiones

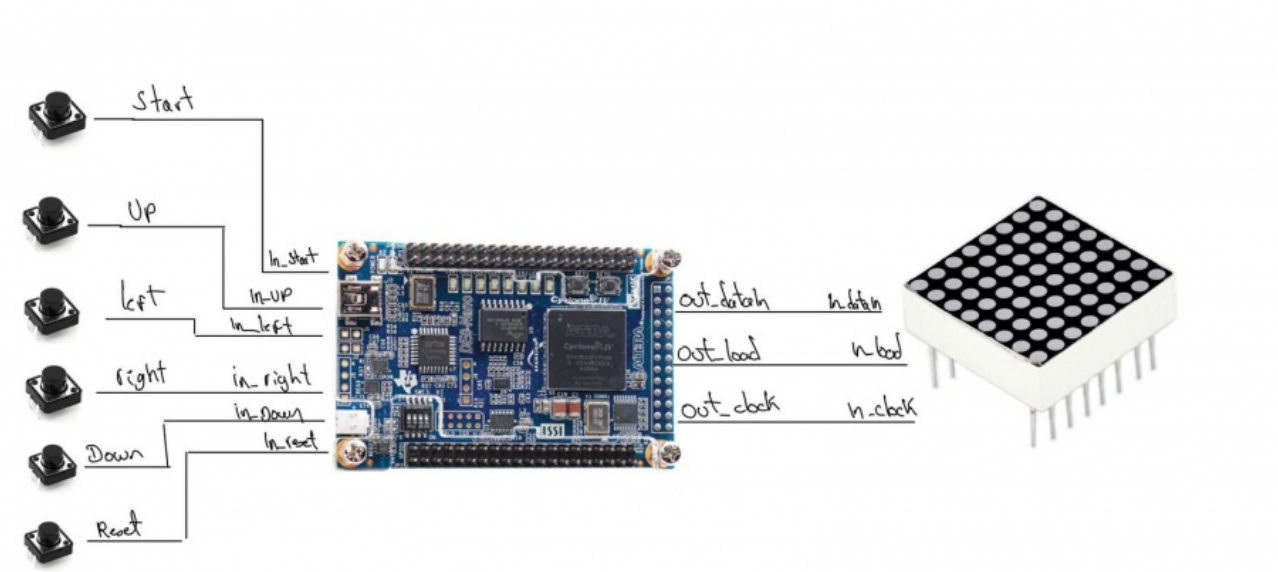
[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de identificar los componentes circuitales constitutivos del producto y su forma de comunicación para comprender y evidenciar la estructura del requerimiento solicitado en términos circuitales.

- a. Se muestran todas las señales de entrada, salida e internas con sus correspondientes nombres estructurados (In/Out) y tamaños (Bit/Bus) para todos los componentes que forman el sistema.
- b. Se presenta una descripción (qué es y cómo funciona) de cada uno de los componentes físicos del producto solicitado, describiendo las conexiones físicas a realizar.

Parrafo 1: Descripción.

Imagen 1: Imagen de diagrama circuital.



Parrafo 2: Descripción de cada componente.

Arq del Sistema (PARTE HARDWARE): Diagrama de Componentes, Señales e Inter-conexiones

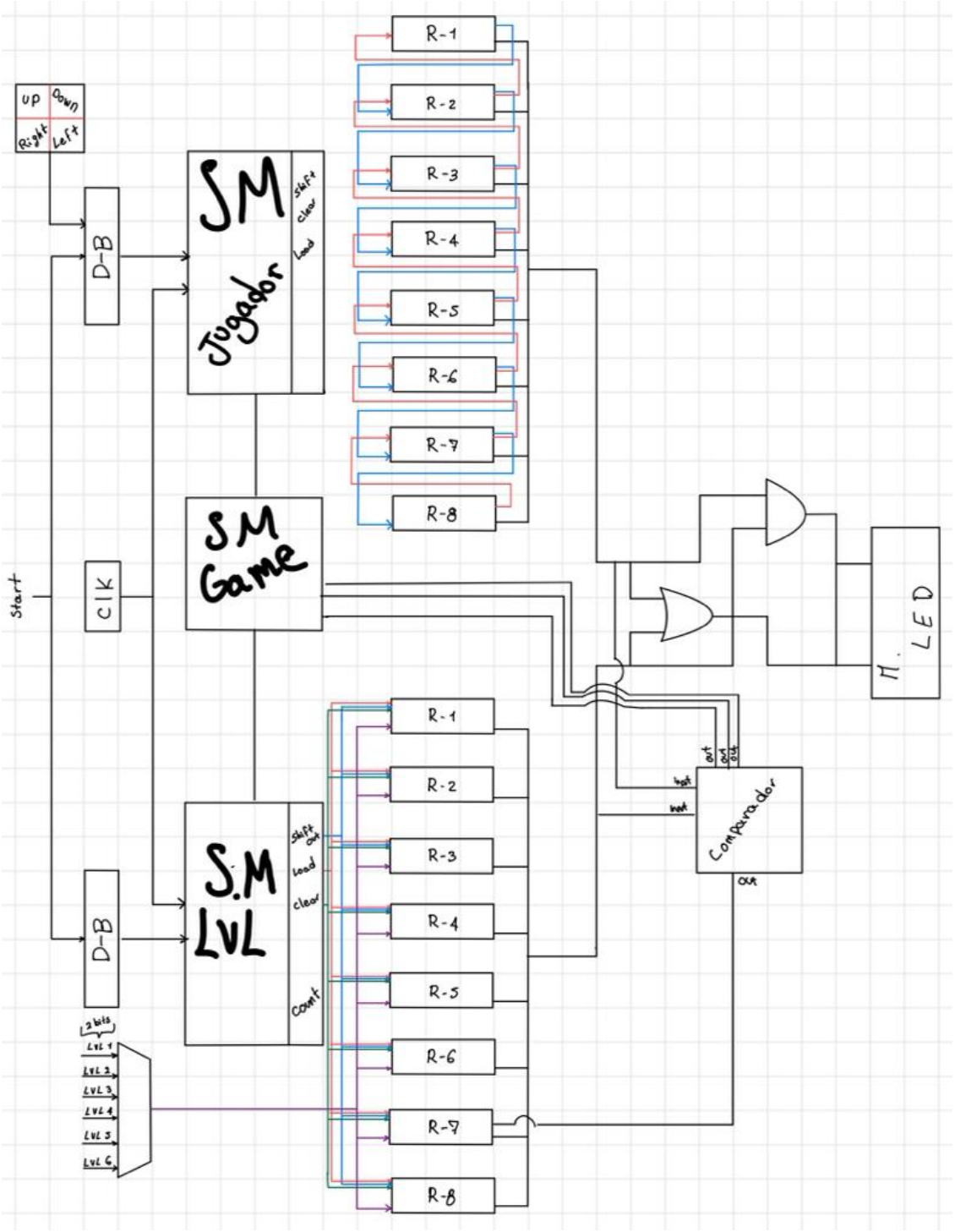
[Collapse]

PRODUCTO DE CALIDAD: : El estudiante es capaz de proponer una arquitectura estructurada usando componentes funcionales (combinacionales y secuenciales) y su forma de comunicación para comprender y evidenciar la estructura y funcionalidad del requerimiento solicitado en términos de componentes.

- a. Se muestran todas las señales de entrada/salida e internas con sus correspondientes nombres estructurados (In/Out) y tamaños (Bit/Bus) para todos los componentes de sistema.
- b. La arquitectura corresponde a una solución eficiente en cuanto a recursos, número de componentes y elementos y algoritmo de solución. Los componentes internos son menos complejos que los de mayor jerarquía.
- c. Se diferencian los componentes combinacionales y secuenciales al igual que las señales de reloj y reset que deben ser compartidas por todos los bloques secuenciales. La arquitectura presenta un módulo de control principal (máquina de estados finita).
- d. Se presenta una descripción (qué es y cómo funciona) de cada uno de los componentes constitutivos del producto solicitado, describiendo sus señales.

Parrafo 1: Descripción.

Imagen 1: Imagen de diagrama de componentes, señales e inter-conexiones.



COMPONENTE (COMBINACIONAL O SECUENCIAL) X

*regPointType:*En este registro, se carga la posicion del jugador, hace shift para que este se mueva para los lados y load en los registros para subir y bajar la posicion del jugador. Entradas:SC_RegPOINTTYPE_CLOCK_50,

SC_RegPOINTTYPE_RESET_InHigh,

SC_RegPOINTTYPE_clear_InLow,

SC_RegPOINTTYPE_load0_InLow,

SC_RegPOINTTYPE_load1_InLow,

SC_RegPOINTTYPE_shiftselection_In,

SC_RegPOINTTYPE_data0_InBUS,

SC_RegPOINTTYPE_data1_InBUS

Salidas: SC_RegPOINTTYPE_data_OutBUS .

RegBACKGTYPE: En este registro, se cargan los obstaculos, se mueven, se actualizan los niveles y en el registro superior se anidan los froggers al llegar. Entradas:SC_RegBACKGTYPE_CLOCK_50, SC_RegBACKGTYPE_RESET_InHigh, SC_RegBACKGTYPE_clear_InLow, SC_RegBACKGTYPE_load_InLow, SC_RegBACKGTYPE_shiftselection_In, SC_RegBACKGTYPE_data_InBUS, SC_RegBACKGTYPE_SET_NN SC_RegBACKGTYPE_NN Salidas: SC_RegBACKGTYPE_data_OutBUS .

bottom side comparator: En este componente se implemento la logica de los shift con sus restircciones, en este caso para caundo la rana este en uno de los bordes laterales de la matriz y se pulse el boton en direccion al borde, la rana no traspase y salga al otro lado sino que se qude donde esta.

MUX: Este componente es el que guarda las condiciones iniciales de cada registro o fila del nivel y cada que pasamos de nivel se cargan las nuevas condiciones para el siguiente nivel hasta ganar. Entradas: CC_MUX_select_InBUS, CC_MUX_data_InBUS_0, CC_MUX_data_InBUS_1, CC_MUX_data_InBUS_2, CC_MUX_data_InBUS_3, CC_MUX_data_InBUS_4, CC_MUX_data_InBUS_5, CC_MUX_data_InBUS_6, CC_MUX_data_InBUS_7, CC_MUX_data_InBUS_8, CC_MUX_data_InBUS_9 Salidas: CC_MUX_z_Out

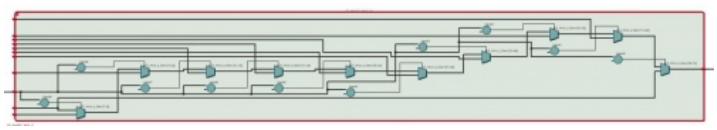
Comparador: Tenemos varios comparadores los cuales tienen funciones especificas, tenemos uno que revisa en todo momento la ultima fila y si esta esta con todos los bits en 1 significa que ganamos y pasamos al suiguiente nivel, tenemos uno que revisa si perdimos o si hubo colisiones que en todo momento esta revisando todas las filas y con un and revisa si nos chocamos y al final tenemos un comparador de registros el cual usa los anteriores e impolementa la logica para seguir con el juego en casops especificos como si avanza de nivel , anido , gano o perdio. Entradas: CC_BACKREG_0, CC_BACKREG_1, CC_BACKREG_2, CC_BACKREG_3, CC_BACKREG_4, CC_BACKREG_5, CC_BACKREG_6, CC_BACKREG_7, CC_POINTREG_0, CC_POINTREG_1, CC_POINTREG_2, CC_POINTREG_3, CC_POINTREG_4, CC_POINTREG_5, CC_POINTREG_6, CC_POINTREG_7 Salidas: CC_REGISTERCOMPARATOR_WinF_OutLow, CC_REGISTERCOMPARATOR_WinL_OutLow, CC_REGISTERCOMPARATOR_Lose_OutLow, CC_REGISTERCOMPARATOR_NN_Outlow

STATEMACHINEBACKG: Aquí es donde se hace todo lo que pasa en el nivel. Se controla la velocidad a la que se mueven los obstáculos, la posición de estos , carga cuando se debe cambiar el nivel, cuando se reinicia vuelve al primer nivel, también controla el anidado del ultimo registro, básicamente se controla todo lo que pase con los registros del nivel. Entradas: SC_STATEMACHINEBACKG_CLOCK_50, SC_STATEMACHINEBACKG_RESET_InHigh, SC_STATEMACHINEBACKG_startButton_InLow, SC_STATEMACHINEBACKG_T0_InLow, SC_STATEMACHINEBACKG_RESET_FromGame, SC_STATEMACHINEBACKG_WINF, SC_STATEMACHINEBACKG_WINL, STATE_Signal Salidas:SC_STATEMACHINEBACKG_clear_OutLow, SC_STATEMACHINEBACKG_load_OutLow, SC_STATEMACHINEBACKG_shiftselection_Out, SC_STATEMACHINEBACKG_upcount_out.

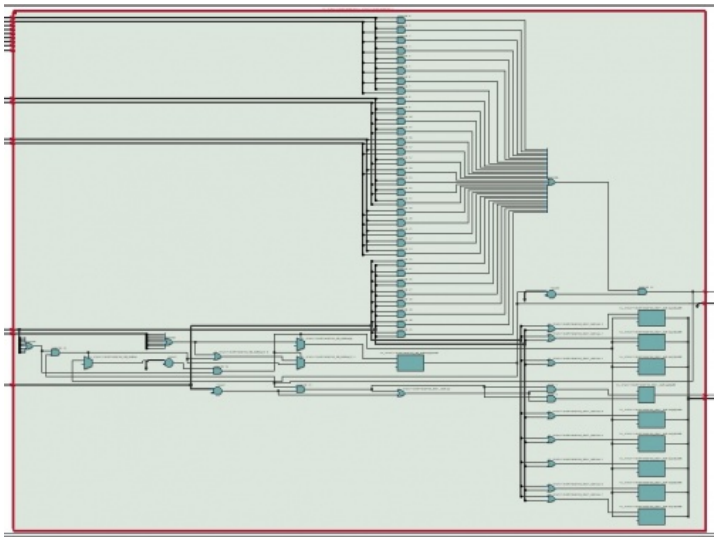
STATEMACHINEPOINT: En esta state machine se maneja todo lo relacionado con el jugador. Aqui, se controlan los shifts que se cargan en los registros y hacen que se mueva el jugador hacia los lados, tambien hay una restriccion para cuando este a los bordes no se salte hacia el otro ladosino que se queda en esa posicion. aqui tambien se maneja el reset cuando se pierde en el nivel y esta la señal para cuando se gana. Entradas:SC_STATEMACHINEPOINT_CLOCK_50, SC_STATEMACHINEPOINT_RESET_InHigh, SC_STATEMACHINEPOINT_startButton_InLow, SC_STATEMACHINEPOINT_upButton_InLow, SC_STATEMACHINEPOINT_downButton_InLow, SC_STATEMACHINEPOINT_leftButton_InLow, SC_STATEMACHINEPOINT_rightButton_InLow, SC_STATEMACHINEPOINT_bottomsidecomparator_InLow, SC_STATEMACHINEPOINT_RESET_FromGame, SC_STATEMACHINEPOINT_WINF, STATE_Signal Salidas:SC_STATEMACHINEPOINT_clear_OutLow, SC_STATEMACHINEPOINT_load0_OutLow, SC_STATEMACHINEPOINT_load1_OutLow, SC_STATEMACHINEPOINT_shiftselection_Out

STATEMACHINEGAME: En esta state machine se maneja todo lo relacionado a la dinamica del juego. Aqui se reciben las señales de ganar, perder y anidar y según lo que haya pasado, se envian señales hacia las otras maquinas de estados para reiniciar el nivel o cambiar de nivel o reiniciar le punto. Entradas: SC_STATEMACHINEGAME_CLOCK_50, SC_STATEMACHINEGAME_RESET_InHigh, SC_STATEMACHINEGAME_startButton_InLow, SC_STATEMACHINEGAME_WinF_InLow, SC_STATEMACHINEGAME_WinL_InLow, SC_STATEMACHINEGAME_Lose_InLow, Salidas: SC_STATEMACHINEGAME_Level_Out, SC_STATEMACHINEGAME_RESET_FromGame_Point, SC_STATEMACHINEGAME_SET_FrogGame, SC_STATEMACHINEGAME_Change_BACKG, STATE_Signal

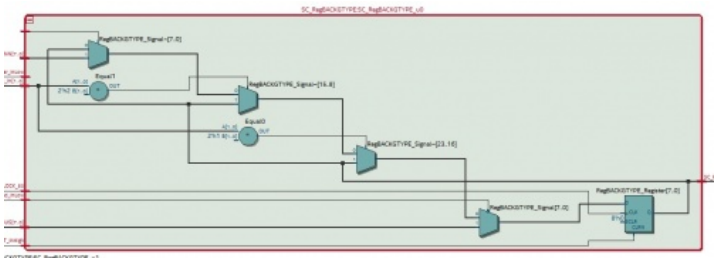
Caja negra del Multiplexor:



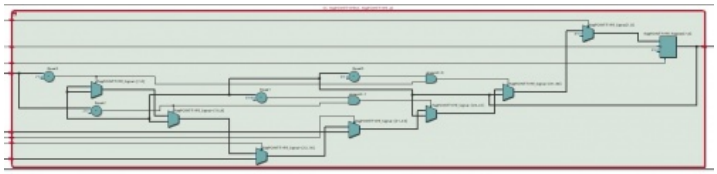
Caja negra del comparador:



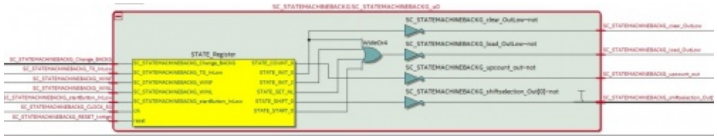
Caja negra del registro del nivel



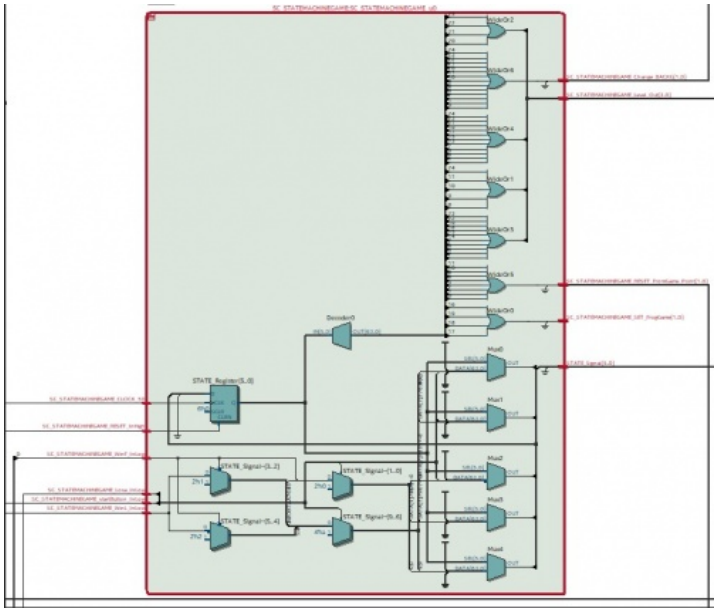
Caja negra del registro del jugador



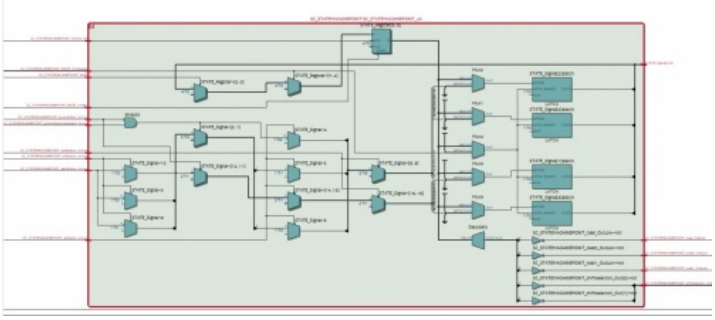
Caja negra de la maquina de estados del nivel



Caja negra de la maquina de estados de la dinamica del juego



Caja negra de la maquina de estados del jugador



Arq del Sistema: (PARTE SOFTWARE): Funciones

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de proponer una arquitectura estructurada usando funciones/métodos y su forma de comunicación para comprender la funcionalidad del requerimiento solicitado en términos de funciones/métodos.

- a. Se muestran todas las variables y constantes de entrada, salida e internas con sus correspondientes nombres estructurados (In/Out) y tamaños (Int/float, etc) para todas las funciones y métodos que forman el sistema.
- b. La arquitectura corresponde a una solución eficiente en cuanto a recursos y algoritmo de solución. Las funciones y métodos internos tienen menor complejidad que las funciones y métodos de mayor jerarquía.
- c. Se muestra una estructura de ejecución de las funciones/métodos (polling, interrupción, scheduling, etc).
- d. Se presenta una descripción (qué es y cómo funciona) de cada una de las funciones y métodos constitutivos del producto solicitado, describiendo sus variables.

Parrafo 1: Descripción.

Imagen 1: Imagen diagrama de funciones o métodos empleados.

Función X

El codigo del frogger esta en el github

REPORTES TÉCNICOS

Memorias de Cálculo

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de realizar cálculos de parámetros cuantitativos de módulos funcionales para dar una solución matemática y científica a los parámetros que lo requieran.

- a. Se presentan memorias de cálculo completas del proyecto, que incluyen justificaciones de su pertinencia para todos los productos que las requieran (ejemplo: cálculos de variables temporales, de espacio de memoria, entre otros).

Los cálculos realizados fueron pocos, debido a que no se hizo uso de contadores para realizar los pasos de nivel. <p><p>En especifico la operación de bloquear el nido cuando la rana llega a un nido vacío, para ello se hizo la siguiente operación: (CC_BACKREG_7 | CC_POINTREG_7) que representa una operación de OR sobre el jugador y los nidos. Este valor se envía al registro 7 para ser almacenado.<p> <p>Otra operación importante es la siguiente: (|(CC_BACKREG_1 & CC_POINTREG_1 | CC_BACKREG_3 & CC_POINTREG_3 | CC_BACKREG_5 & CC_POINTREG_5 | CC_BACKREG_7 & CC_POINTREG_7) && CC_REGISTERCOMPARATOR_NN_Outlow == 1'b0). Esta comparación define si se perdió o no, se revisan solo los registros con obstaculos y se revisa que no se este anidando.

La ultima operación importante es la revisión de anidado es la siguiente ((|CC_POINTREG_7)). En la cual se mira que efectivamente el punto este en la ultima fila <p>

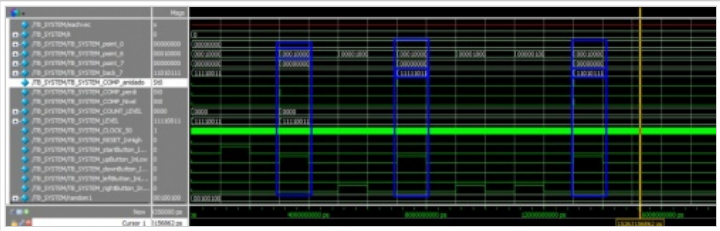
Definición de vectores de prueba y simulaciones de respaldo

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de proponer una estrategia de identificación de vectores de prueba para validar la funcionalidad del producto.

- a. Los vectores de prueba se seleccionan describiendo en un párrafo una estrategia explícita y claramente definida y estos vectores permiten comprobar totalmente la funcionalidad.
- b. Se presentan resultados de simulación para el producto solicitado, explicando tres o más casos de funcionamiento sobre el diagrama de simulación. La simulación contiene marcadores en la gráfica que señalan situaciones específicas del prototipo.

<p> *Parrafo 1:*Se va a revisar anide, perdi o gane. En la imagen se pueden identificar tres rectangulos azules los cuales separan los estados anteriormente mencionados. Primer rectangulo Perdi: podemos identificar que en este rectangulo se perdio ya que se da un pulso hacia arriba y se deja quieto para que choque contra los obstaculos, depues de identificar esto podemos ver al lado derecho de TB_SYSTEM_point_6 que vuelve a aparecer el frogger en la misma posicion inicial.



None

Segundo rectangulo anide: Se puede identificar que anida por medio de que el frogger sube a la posicion 1111x011 donde la x es el frogger, por lo que vuelve a aparecer el frogger, sin embargo en la ultima fila se mantiene la misma fila por lo que se concluye que anido.

Tercer rectangulo gane: Se puede observar que con el cuadrado anterior ya se habia anidado una vez por lo que se vuelve a anidar en el punto que falta y se obtiene un cambio en el nivel ya que los lugares de anidar cambiaron.

Identificar los vectores de prueba relevantes de bloques/funciones invididuales y de todo el sistema.

<p> *Parrafo 2:* Descripción

En general los vectores de prueba eran casos especiales para verificar si se perdia, si se anidaba, si se ganaba y en ese proceso ver como cambiaba el mapa y la posición del jugador. Esos valores eran 1 y 0 para los botones.

Imagen de simulaciones

Indicadores de utilización de recursos y rendimiento de los dispositivos utilizados

<thead> </thead> <tbody> </tbody>

TIpo de codigo	**porcentaje**
hardware	90%
software	9%
dispositivo	1%

IMPLEMENTACIÓN

Descripción en lenguajes HW y SW (Códigos fuente)

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de describir lo solicitado en lenguajes de HDL para evidenciar la utilización de los elementos de la herramienta de diseño y del lenguaje y contruir su solución en tecnología de dispositivos de lógica programable.

- a. La descripción en lenguajes hardware (complejidad del código, diferencia combinacional y secuencial) y software (complejidad del código, diferencia tipos de variables funciones métodos) es correcta y corresponde al producto solicitado.
- b. Se incluye documentación completa para estructurar y/o entender el código claramente (indentación y sintaxis de los lenguajes), nombrando correcta y adecuadamente todas las señales, variables y demás elementos relevantes.

Los archivos de código estan en el repositorio de Github presente en el final de la wiki

Funcionalidad modular

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de identificar una agrupación de componentes para evidenciar una funcionalidad parcial del producto para concretar avances efectivos.

- a. Se definen módulos (que sumados corresponden a todo el prototipo) sobre los cuales se propusieron pruebas independientes y todas funcionaron adecuadamente y completamente según las especificaciones.

Vídeos y fotos de demostración de módulos del sistema

Parrafo 1: Los videos estan en el github.

Video/Foto 2

Funcionalidad global del sistema

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de integrar todos los componentes en un producto funcional que cumple con toda las especificaciones y restricciones para evidenciar una funcionalidad global de la solución.

- a. El prototipo funciona adecuadamente y completamente según las especificaciones del producto solicitado.

Vídeos y fotos de demostración del prototipo final

Parrafo 1: Para esta prueba el video se encuentra en el GitHub

Video/Foto 2

REFLEXIÓN

Resultados y lecciones aprendidas

[Collapse]

PRODUCTO DE CALIDAD: El estudiante es capaz de fundamentar/explicar el proceso de diseño realizado evidenciando las mejoras o problemas de la solución propuesta para demostrar la comprensión y dominio de las actividades propias del diseño.

- a. Se proponen nuevas especificaciones y aplicaciones del trabajo realizado (ejemplo: mayores niveles de complejidad y usos en otros contextos)..
- b. En caso de no lograr el item de funcionamiento glogal; identifica y argumenta las razones principales del no funcionamiento.
- c. El lenguaje disciplinar es preciso y adecuado, haciendo uso de frases gramaticalmente correctas, sin errores ortográficos.

<p> *Párrafo 1:* Al comienzo del proyecto planteamos muchas ideas de cosas diferentes para que el frogger fuera distinto y unico, pero la verdad no sabiamos como implementar estas cosas algunas de estas eran power ups que dieran ventajas al jugador, tambien queriamos implementar que el nivel se fuera moviendo hacia abajo a medida que el jugador subia. Finalmente solo implementamos que el frogger no cruzara de lado a lado sino que en caso de estar en la izquierda y pulsar el boton izquierda se quedara en el mismo lugar.

Párrafo 2: Se encontraron varias dificultades a la hora de realizar el proyecto como la dificultad frente a la nueva aplicacion, ademas del lenguaje verilog el cual no es muy simple, aprender las diferentes funciones, parametros, entre otros.

Párrafo 3: Logramos obtener el resultado que queriamos, a la hora de realizar el video final no nos dimos cuenta de que el video no se podia ver bien y nos daba un error a la hora de reproducirlo, sin embargo por medio de la simulacion logramos obtener el resultado final.

Trabajo Colaborativo

[Collapse]

PRODUCTO DE CALIDAD:Los estudiantes son capaces de cooperar y contribuir en el proceso de diseño para sopesar las utilidades y alcance del trabajo en equipo, en la obtención de metas comunes. a. Participación: todos los miembros del grupo contribuyeron por igual y asistieron a las reuniones programadas y cumplieron sus tareas designadas.

Párrafo 1: Por medio de acuerdos en el grupo logramos sacar adelante el proyecto, ya que nos dividimos el trabajo, acordabamos horarios para reunirnos, en caso de que no pudieran se avisaba para ver que hacer, ademas de asistir a los laboratorios con preguntas y errores para que sebastian y los monitores nos ayudaran a resolverlos.

Párrafo 2: Por medio de que somos de diferentes semestres en algunos casos no podiamos asisitir a todas las reuniones acordadas o los horarios acordados.

Párrafo 3: Recomendaria tener mas disponibilidad con las FPGAs, en la mayoria de las clases queriamos probar algo pero varios grupos la necesiataban por lo que no podiamos probar varias cosas en poco tiempo.

MATERIALES

Dispositivos Hardware

- FPGA (altera)
  - Hoja de datos [2] (https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/ds_cyc.pdf)

Herramientas Software

- Verilog
  - [3] (https://www.intel.la/content/www/xl/es/support/programmable/support-resources/design-examples/horizontal/ver-hier.html)
- Quartus
  - [4] (https://www.intel.la/content/www/xl/es/software/programmable/quartus-prime/overview.html)

Código del proyecto

- GitHub.
  - Enlace Github [5] (https://github.com/Ersebreck/SED_Frogger)

BIBLIOGRAFIA

- Titulo - Autor (año) (Descargar (http://www.libros.com/ref1.pdf))
- Titulo - Autor (año) (Descargar (http://www.libros.com/ref2.pdf))
- Titulo - Autor (año) (Descargar (http://www.libros.com/ref3.pdf))