

1. Historia del proyecto.

En el momento de escribir esta introducción, ha finalizado el curso 2000/2001, con lo que finaliza el 4º año de desarrollo. En total han pasado por el mismo 44 alumnos y se han escrito miles de líneas de código.

En un proyecto de este tipo, no hay una progresión lineal en cuanto a la evolución del mismo, sino que suelen producirse un ciclo de desarrollo en espiral, en el que los avances en un momento determinado hacen que se vuelva a dar una nueva vuelta en la que se han refinado requisitos, replanteado objetivos y aumentado el alcance del mismo. Alguna de estas vueltas han replanteado hasta la base misma del proyecto.

En las siguientes líneas se hace un extracto histórico del proyecto, necesariamente breve, en el que se indican los hitos fundamentales producidos en cada generación.

1.1. Curso 1997/1998

Punto de arranque del proyecto. Se plantea la necesidad de un emulador para realizar prácticas sobre el Indalo, y por tanto este será el eje este año. Participan en el proyecto 6 personas y como tareas el diseñar el emulador de la CPU y un ensamblador.

Una decisión que se toma en esta fase temprana es la elección del lenguaje a utilizar, y por mayoría de los componentes del grupo se decide el uso de Java que presenta como principales ventajas su capacidad real de generar aplicaciones multiplataforma (algunos componentes del proyecto sólo usan Linux en sus ordenadores particulares) y su sólida adaptación a la programación orientada a objetos. El principal inconveniente es el de su menor rendimiento en cuanto a velocidad de ejecución respecto a otros lenguajes. Se considera que el tiempo atenúa este inconveniente debido al aumento constante de la capacidad de proceso de los nuevos ordenadores.

Por tanto se diseña y construye un ensamblador (Alfredo Zurdo nos sorprende a todos cuando realiza un ensamblador funcional en un fin de semana, dedicando el sábado al aprendizaje del Java) y por otro se aborda el emulador.

En el diseño del emulador, surgen dos filosofías, a las que llamamos encadenamiento hacia adelante y encadenamiento hacia atrás. Básicamente la CPU está compuesta por componentes menores como registros, puertas triestado, etc. los cambios de estado pueden hacerse notificando a un elemento que ha cambiado alguna entrada, actuar en consecuencia, y si cambia alguna salida notificar a todos los componentes conectados a ella la nueva situación. A esto lo denominamos encadenamiento hacia delante.

Encadenamiento hacia atrás consiste en que si en un momento determinado queremos conocer el estado de una conexión de un circuito, este consulta sus entradas, que para saber su estado piden la situación de las salidas a las que están conectadas y así sucesivamente. Esta última implementación no llega a funcionar correctamente, por lo que terminamos por abandonar esta línea.

Al final del curso disponíamos de un prototipo, que era parcialmente funcional en alguno de sus componentes, y que permitió asentar las bases para el crecimiento posterior.

1.2. Curso 1998/1999

Durante este curso colaboraron 9 personas en el proyecto, y supuso importantes avances en diferentes frentes. Se comienza con el diseño de un entorno de trabajo con varios paneles, en este punto el *Indalo* es todavía el eje de la aplicación. Por tanto todo se centra en abrir ficheros fuente, ensamblarlos y ejecutar los programas sobre la CPU.

Sobre el entorno de trabajo se desarrolla un panel sobre el que se colocan componentes y líneas de conexión, por lo que a la funcionalidad de los componentes se le añade la imagen que los representa en el emulador.

Por otro lado se mejora el ensamblador y se crea el desensamblador. También se trabaja para completar la CPU, generando una imagen vectorizada del contenido de la misma, en la que cambian de color los componentes que cambian de estado. En cuanto al interior de la CPU se sigue trabajando sobre ella, desarrollándose de forma completa el decodificador, que constituye la parte más compleja de la misma.

Por diferentes fallos en el entorno de ventanas de la aplicación, y algunos en la CPU, el proyecto no quedó lo suficientemente refinado como para ponerlo en uso en los laboratorios.

1.3. Curso 1999/2000

Se suman 14 personas al proyecto. Este es el año del empujón definitivo. El trabajo se divide en varios frentes. Por un lado Daniel Miegimolles, asume labores de coordinación y de desarrollo del entorno de ventanas. Por otro lado se aborda el desarrollo del depurador, en el que se podrán ejecutar las instrucciones paso a paso o a máxima velocidad, pudiéndose visualizar en todo momento cual es la instrucción en curso y el estado de registros y flags.

También se sigue trabajando en la CPU, para eliminar errores y mejorar su funcionamiento. En esta parte es especialmente destacable la labor de Antonio Carvajal, que redefine gran parte de la estructura del código sobre el que se sustenta la CPU. Todo ello desembocó en el programa *Indalo 2000*, que ya si que era funcional y que entró en explotación en ese mismo curso. Por fin disponemos del emulador para primero, en el que pueden hacer prácticas los alumnos de la asignatura de Laboratorio de Electrónica Digital.

Hubo un último grupo de trabajo que siguió desarrollando el tema de gráficos, haciendo especial incapié en el trazado automático de líneas entre dos puntos, rodeando los obstáculos que se puedan presentar.

1.4. Curso 2000/2001

Durante este curso, y recogiendo toda la experiencia acumulada, se redefinen las bases del proyecto para mejorar diferentes aspectos que limitaban el desarrollo de la aplicación. Por un lado y aprovechando la experiencia de Antonio Carvajal que se reengancha este año al proyecto. Se vuelven a diseñar las clases base, dando un nuevo enfoque a la aplicación.

Básicamente se trata de acercar el emulador a la realidad. Por un lado la CPU deja de ser el eje del diseño. La CPU pasa a ser un componente más, muy complejo pero uno más. Quizás este fue una de las limitaciones que constreñía al proyecto. Por otro lado surge la dificultad que supone en algunos circuitos las conexiones de retroalimentación, es decir que la salida de una etapa se conecta con entradas de etapas

anteriores. Para que esto funcione de forma correcta es necesario tener en cuenta los tiempos de respuesta que deben tener cada uno de los componentes.

También se reconsideran las diferentes posibilidades para las entradas y salidas de los circuitos, introduciendo algunas modificaciones para las entradas al aire y las salidas triestado.

Se avanza también en la parte gráfica analizando diversos programas de emulación para ver métodos intuitivos y cómodos para el trazado de líneas de conexión, se opta por una alternativa, que finalmente es implementada. También se aborda la definición de la representación de los circuitos, permitiendo el uso de bitmaps, dotando de sensibilidad al ratón en las patillas de conexión y capacidad de seleccionarias de diferentes formas.

También se procede a diseñar algunos componentes, comenzando desde las puertas estándar y continuando por circuitos simples tales como multiplexores, decodificadores, contadores, biestables, ... Finalmente se añaden algunos circuitos más complejos, tales como el Controlador Programable de interrupciones 8259, la memoria RAM estática 2114 haciendo especial incapié en respetar de forma escrupulosa las temporizaciones expresadas en su cronograma y también se modifica el Indalo para generar la versión que presenta tiempos de respuesta.

1.5. Curso 2001/2002

La expectativas para este curso, son las de volver a crear una versión ejecutable del programa. Para ello se debe trabajar en varios frentes:

Por un lado, terminar la parte gráfica de trazado de líneas, que ahora está solo parcialmente resuelto desde el punto de vista del interface con el usuario, pero no desde el de comportamiento interno. A nivel de interface no está resuelta la selección de segmentos y nodos, para su modificación o borrado. Otro tema que se podría apuntar es el de la posibilidad de deshacer con uno o varios niveles.

Por otro lado hay que crear/modificar el entorno de la aplicación, redefiniendo los menús, opciones, etc.

Finalmente hay que modificar el comportamiento de los componentes ya diseñados de acuerdo a las últimas (y espero que realmente sean últimas) modificaciones que se han realizado en las clases básicas.