EA773 - Experimento 5

Wu Shin - Ting
DCA - FEEC - Unicamp

19 de Novembro de 2009

O projeto consiste em implementar uma calculadora com memória com uso de barramento de 8 *bits*. Neste documento são discutidos os pontos críticos do projeto. Para ilustrar são apresentados blocos de circuitos de 4 *bits*.

1 Visão Geral da Organização da Calculadora

Foi explicada uma possível organização dos componentes da calculadora: barramento, chaves (de entrada), circuito combinacional de UA, registradores para armazenar os operandos X e Y, registrador para armazenar o resultado de uma operação aritmética. O sistema é provido de duas botoeiras para carregar o operando X e o código de operação, respectivamente, e 4 *displays* de 7 segmentos para mostrar os resultados em decimal. Figura 1 mostra o esquemático de um circuito com funcionalidades similares, porém com operações limitadas para 4 *bits* e o "conversor binário–decimal" é "reduzido" a um "conversor binário–binário" a fim de mostrar uma estratégia de controlar o momento de conversão, sem a intervenção do usuário por meio de uma botoeira adicional.

Vimos na aula que os acessos ao barramento podem ser controlados pelas portas *tri-states*, que apresentam a característica de se "desconectarem" eletricamente do barramento através de um sinal de controle OUT. Figura 2 mostra um banco de portas it tri-states utilizado na saída de todos os registradores para o barramento (Figura 3). No meu projeto utilizei os *latches*, para que fiquem compatíveis com a especificação das outras turmas, mas os *flip-flops* também funcionariam. Basta lembrarem que os primeiros são sensíveis a nível e os segundo à borda.

Vimos também na aula que, para carregar o operando X, podemos utilizar diretamente o pulso gerado pela botoeira dedicada à carga deste operando bx, e para carregar o código de operação C.O., temos que elaborar uma estratégia de desdobrar o pulso em dois pulsos p_1 e p_2 para que tenhamos sinais de controle distintos na linha de tempo a fim de executar dois conjuntos de operações sequenciais: (1) carregar efetivamente o operando, via

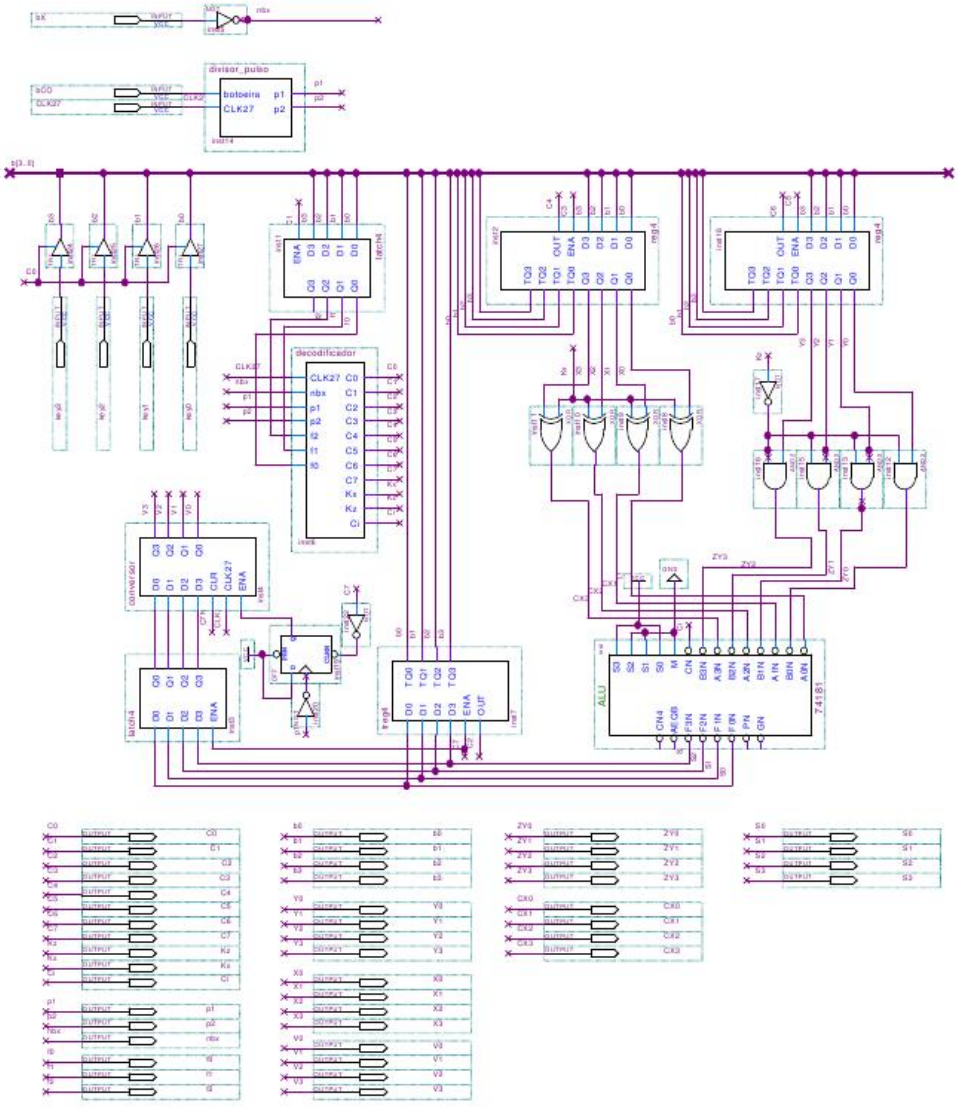


Figura 1: Esquemático do circuito.

barramento, no registrador para ser decodificado em sinais de controle apropriados de forma a habilitar apropriadamente os módulos combinacionais ou sequenciais; (2) transferir os dados, via barramento, entre os registradores (X, Y ou acumulador do resultado). Uma solução para desdobrar o pulso em dois pulsos menores é mostrada na Figura 4. Observe que foi utilizado um registrador de deslocamento de 6 períodos do "relógio do sistema".

O circuito foi projetado para o seguinte repertório de instruções, cujos códigos de operação são:

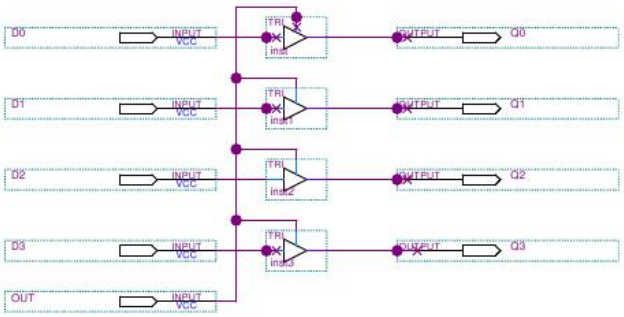
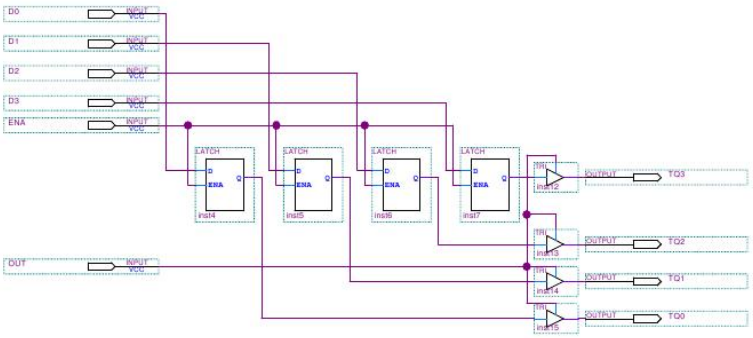
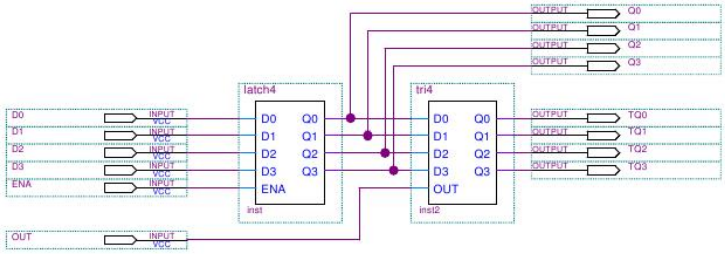


Figura 2: Portas de *tri-states*.



(a) Acumulador para o barramento



(b) Registradores X e Y para o barramento

Figura 3: Saída dos registradores para o barramento.

Instrução	Código de Operação ($f_2 f_1 f_0$)
ADD	000
EXSUB	010
CMX	110
CS	101
CYA	011
CAY	001
NOP	100

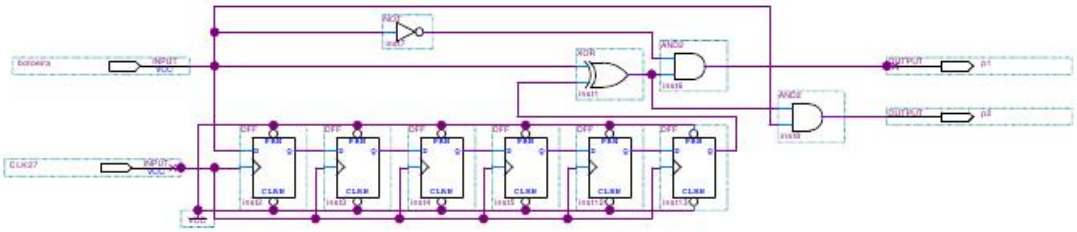


Figura 4: Divisor de pulso.

Aplicando as técnicas de síntese de expressões lógicas, chega-se às seguintes funções de chaveamento em termos dos sinais de entrada $nbx = \neg bx$, os dois pulsos p_1 e p_2 desdobrados do pulso da botoeira que carrega o código de operação, e dos códigos de operação $f_2 f_1 f_0$:

$$
\begin{aligned}
C_0 &= nbx + p_1 \\
C_1 &= p_1 \\
C_2 &= (\neg f_2 \neg f_0 + f_2 f_1 + f_2 f_0) p_2 \\
C_3 &= nbx + \neg f_2 f_1 f_0 p_2 \\
C_4 &= \neg f_2 \neg f_1 f_0 p_2 \\
C_5 &= (\neg f_2 \neg f_0 + f_2 \neg f_0 + \neg f_1 f_0) p_2 \\
C_6 &= \neg f_2 f_1 f_0 p_2 \\
C_7 &= (\neg f_2 \neg f_0 + f_2 f_1 + f_2 f_0) p_1
\end{aligned}
$$

2 Dúvidas Comuns e Sugestão de Solução

Além do desdobramento do pulso da botoeira em dois pulsos, observei alguns problemas comuns nos projetos:

1. Como é o controle da transmissão de dados pelo barramento? Os pares de sinais (C_0, C_7), (C_2, C_5), (C_4, C_5), (C_6, C_3), quando ocorrem simultaneamente, devem gerar, **teoricamente**, respostas iguais nos componentes. Na prática, pode-se ocorrer o fenômeno de **corrida** de respostas, em que um componente sempre responda mais rápido que o outro. Imagine, por exemplo, que C_5 desabilite o registrador Y depois de C_4 colocar a porta tri-state em estado Z. O barramento pode ficar, por uma fração de tempo, em um estado "indefinido" e "destruir" o conteúdo esperado do registrador Y! A solução seria garantir que os registradores receptores sejam desabilitados antes do chaveamento das portas *tri-states*. Um circuito para introduzir atrasos seria o registrador de deslocamento que vimos na Experiência 3, como ilustra o circuito do módulo decodificador apresentado na Figura 5. Observe que

o circuito gera pulsos de 3 períodos. Tendo pulsos de 6 e 3 períodos, podemos utilizá-los para habilitar, respectivamente, os registradores acionadores e os registradores receptores. Isso assegura que os registradores receptores não sejam habilitados quando o barramento estiver em estado indefinido.

$$
\begin{aligned}
C_0 &= nbx + p_1 \\
C_1 &= p_1 \\
C_2 &= (\neg f_2 \neg f_0 + f_2 f_1 + f_2 f_0) p_2 \\
C_3 &= nbx + \neg f_2 f_1 f_0 p_{2,\frac{1}{2}} \\
C_4 &= \neg f_2 \neg f_1 f_0 p_2 \\
C_5 &= (\neg f_2 \neg f_0 + f_2 \neg f_0 + \neg f_1 f_0) p_{2,\frac{1}{2}} \\
C_6 &= \neg f_2 f_1 f_0 p_2 \\
C_7 &= (\neg f_2 \neg f_0 + f_2 f_1 + f_2 f_0) p_{1,\frac{1}{2}}
\end{aligned}
$$

2. Como se consegue perceber os estados instáveis de um registrador? Vale enfatizar aqui que os estados instáveis de um registrador, variando na ordem de nanosegundos, não são perceptíveis a olho nu. Somente com uma boa ferramenta de simulação é possível visualizar tais estados. E isso facilita a identificação dos problemas e o foco das soluções. No ambiente Quartus II, a ferramenta Simulator pode ser utilizada para esta finalidade. Podemos colocar saídas em qualquer ponto do circuito, como ilustra a Figura 1, e editar as formas de onda **compatíveis** com o comportamento dos componentes implementados em *kits* do Altera. Algumas características são:

   - botoeiras são ativo-baixo, ou seja, elas são normalmente em nível 1 e ao serem acionadas, vão para nível lógico 0;
   - relógio do sistema de 27MHz gera sinais na frequência de 27MHz, em torno de 37ns por período.

   Vale ressaltar que se tentarmos simular o nosso circuito com uma frequência maior do que ela suporta, as formas de onda podem aparecer como **reticulados diagonais** que representam "estados indefinidos".

   Figura 6 mostra as formas de onda da simulação do circuito apresentado na Figura 1. Observe que através das formas de onda podemos avaliar o comportamento de cada sinal em uma escala da ordem de nanosegundos!

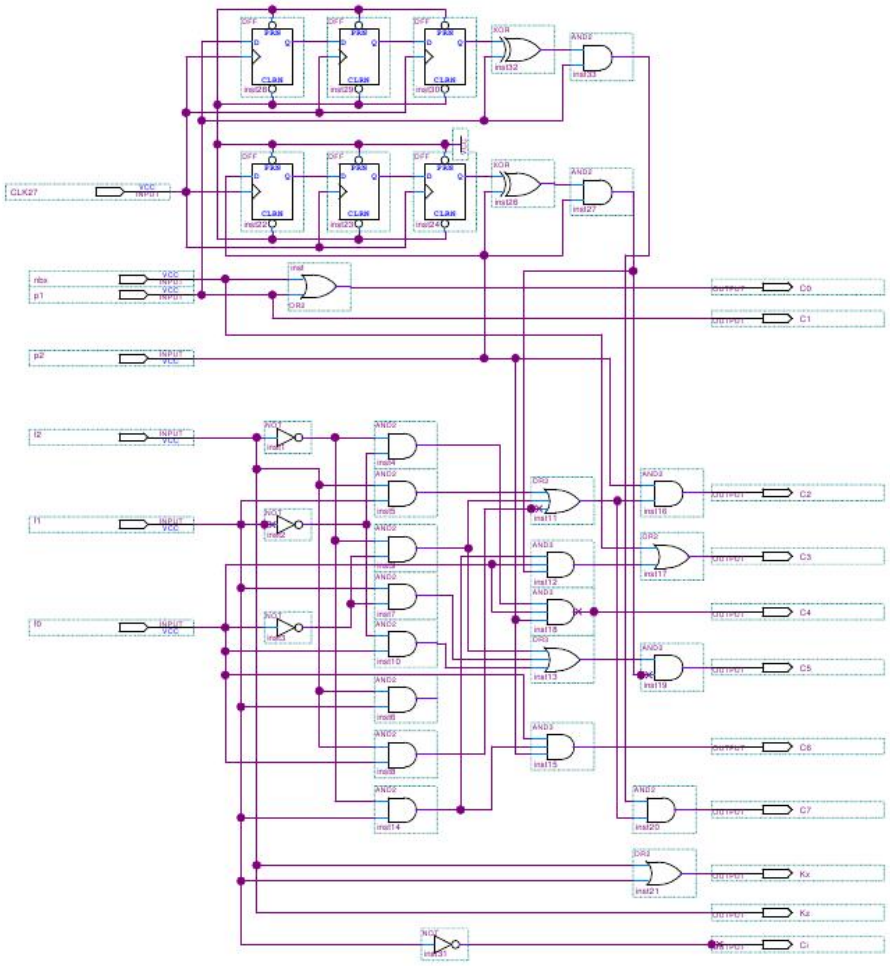


Figura 5: Decodificador.

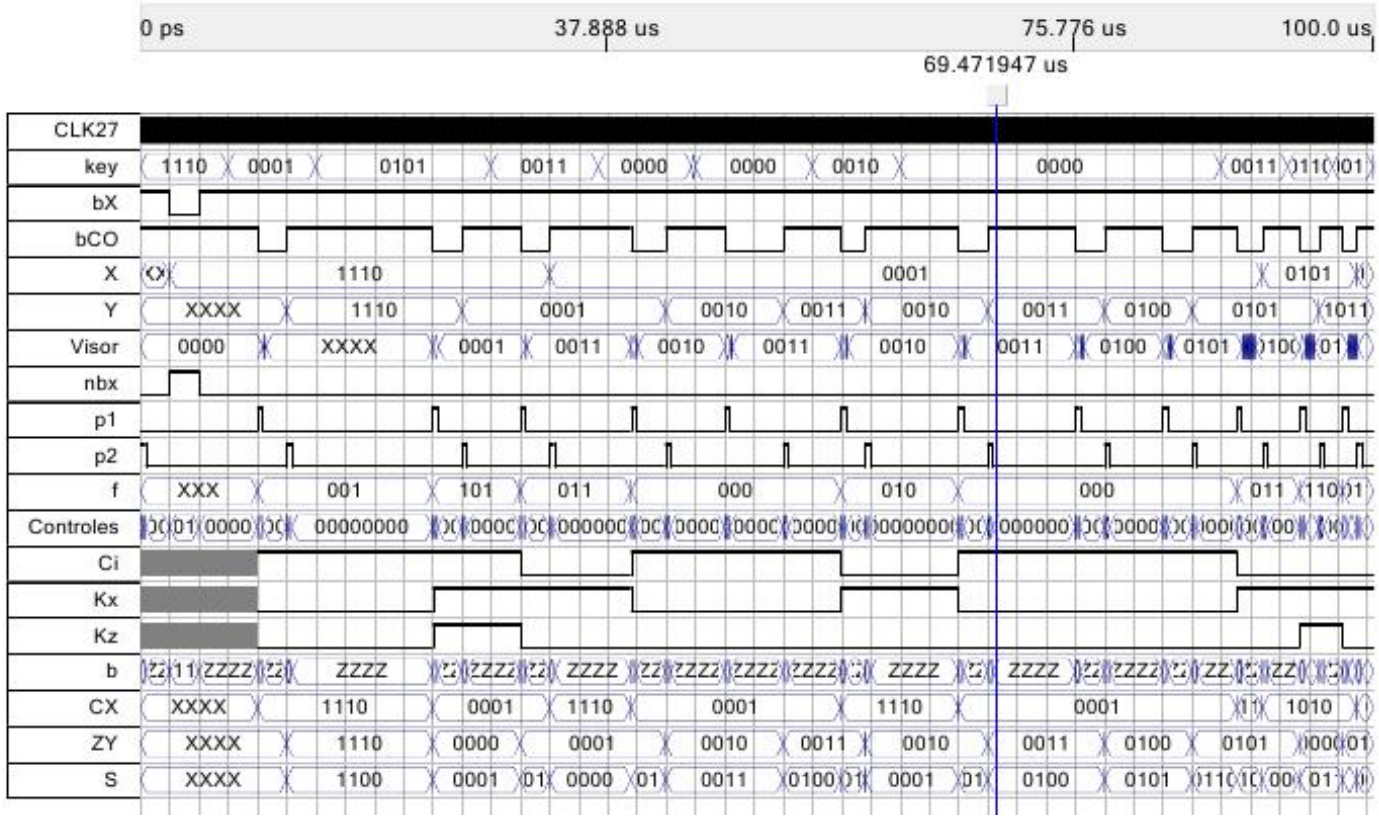


Figura 6: Simulação do circuito.

3. Quando e como habilitar o conversor de binário para decimal? O conversor deve apresentar resultados das operações lógico-aritméticas. Portanto, ele deve mostrar o conteúdo do registrador Y ou do registrador colocado na saída do somador, ou seja, do acumulador. Uma solução seria introduzir um *flip-flop* para controlar o **estado** do conversor: habilitado ou desabilitado. Toda vez que a botoeira de código de operação for acionada, o estado passa a ser "desabilitado". E, quando o acumulador tiver resultado da operação estabilizado, o conversor é novamente habilitado. Quando se estabiliza o resultado de uma operação? O circuito foi projetado para ser no final do pulso de C_7. Observe que no circuito do módulo apresentado na Figura 7 o contador pode estar desabilitado, mesmo que o sinal ENA seja ativado. Isso é necessário, pois a contagem é somente feita até que o conversor atinja o valor que se deseje visualizar nos *displays*. Vale ressaltar aqui que não está incluído no módulo o circuito combinacional de "extrair a magnitude do resultado" antes da conversão. Este circuito deve ser colocado entre o somador e a entrada do conversor e o pulso C_7 é suficiente para abranger estas operações também.

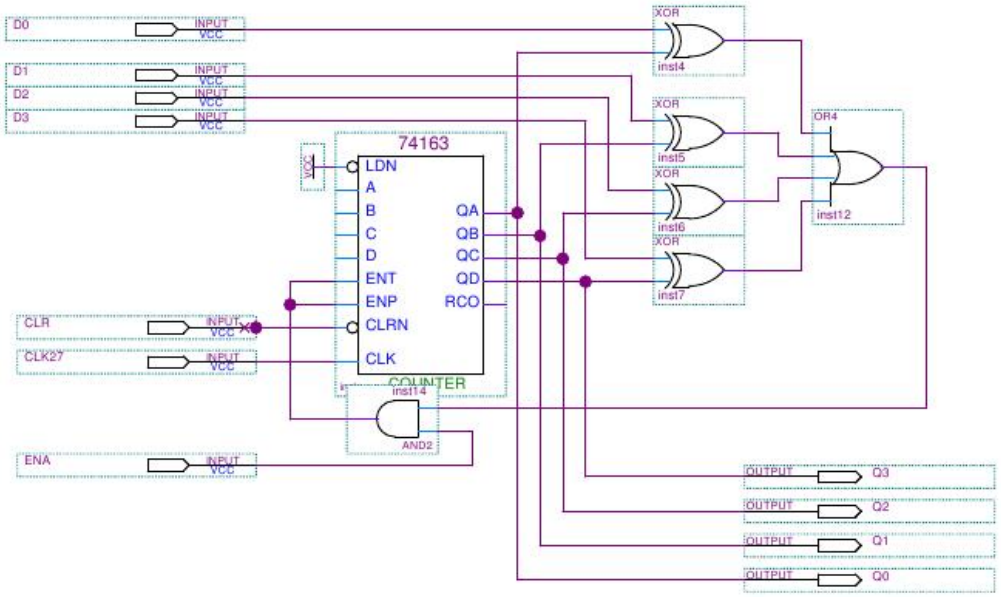


Figura 7: Decodificador.