

## Trabalho 1 – Projeto e simulação de um acumulador de 8 bits com sequenciador de dados em memória

Este trabalho tem como objetivos o projeto e a simulação de circuitos lógicos recorrendo à ferramenta DIGITAL (<https://github.com/hneemann/Digital>). A abordagem de projeto seguida baseia-se nos conceitos de hierarquia e modularidade, técnicas fundamentais para estruturar e validar o projeto de circuitos digitais.

No Sigarra (pasta relativa ao trabalho 1) está disponível um arquivo com componentes e dados necessários para o trabalho. Deve extrair esses conteúdos para a pasta em que ficará toda a informação dos circuitos a projetar.

As secções seguintes descrevem as tarefas do trabalho, devendo realizá-las pela ordem apresentada. Em cada circuito utilize a designação indicada, assim como os nomes das respetivas entradas e saídas. Consulte detalhes de utilização da ferramenta no documento Simulação de Circuitos Lógicos com DIGITAL.

### 1 Somador completo

Um somador completo (*full-adder*, FA) é um circuito com três entradas e duas saídas que realiza a adição de três números de 1 bit (entradas), produzindo na saída 2 bits que representam o valor resultante. As três entradas,  $A$ ,  $B$  e  $Ci$ , representam, respetivamente, os dois valores a somar e o *carry-in*. As duas saídas,  $Co$  e  $S$ , representam, respetivamente, o bit mais significativo do resultado (*carry-out*) e o bit menos significativo (soma).

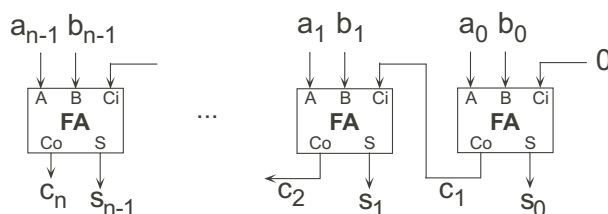
Definindo estas funções numa tabela de verdade é possível obter as suas expressões e respetivo circuito com a ferramenta DIGITAL. Para tal proceda da seguinte forma:

1. No menu, selecione **Analysis** e a opção **Synthesize**. A tabela gerada possui, por omissão, três entradas e uma saída. No menu **Columns** da tabela escolha **Add Output Column** para acrescentar uma saída. Altere o nome dos sinais, clicando sobre cada um com o botão direito do rato, de modo a corresponderem aos pretendidos. Finalmente preencha as colunas de  $Co$  e  $S$  com os valores resultantes de  $A + B + Ci$ . Se quiser pode gravar a tabela num ficheiro (**File** → **Save**) para uso posterior.
2. No menu **Create** da tabela escolha **Circuit** para gerar o circuito do somador completo. Guarde-o com o nome FA (menu principal, **File** → **Save As**).
3. Verifique o funcionamento do circuito simulando-o. Iniciar a simulação selecionando **Simulation** → **Start Simulation** (ou o botão ▶). Atuar as entradas e verificar se as saídas correspondem aos valores esperados (definidos na tabela de verdade).

Q1: Obtenha as expressões de  $S(A, B, Ci)$  e  $Co(A, B, Ci)$  selecionando **Analysis** no menu **Analysis**. Mostre que  $S$  é equivalente a  $S = (A \oplus B) \oplus Ci$ .

## 2 Somador de 8 bits

O circuito anterior é um componente fundamental para a construção de circuitos aritméticos. O diagrama abaixo mostra como implementar um somador de dois números com  $n$  bits, obtido por interligação de  $n$  componentes do tipo somador completo. O operando  $A$  é representado pelos bits  $a_{n-1}, \dots, a_1, a_0$ , o operando  $B$  é representado pelos bits  $b_{n-1}, \dots, b_1, b_0$ , e a soma  $S$  é dada pelos bits  $s_{n-1}, \dots, s_1, s_0$ .



Este circuito é designado por *ripple-carry adder* e produz o resultado da adição de forma semelhante à que é seguida quando se efetua a operação manualmente: a saída  $Co$  de um somador completo representa o bit de transporte para o próximo andar, devendo ser ligada à entrada  $Ci$  do somador completo seguinte. Note que a entrada  $Ci$  do andar menos significativo deverá ser ligada a zero e que a saída  $Co$  mais significativa representa o transporte resultante da soma dos bits mais significativos.

Tendo isto em consideração pretende-se implementar um somador de 8 bits, procedendo da seguinte forma:

1. Desenhe o circuito somador de 8 bits, **Add8**, por interligação de 8 componentes **FA**. O somador completo, **FA**, está disponível em **Components** → **Custom**<sup>1</sup>. Os operandos a somar,  $A$  e  $B$ , são definidos pelas entradas  $A7, A6, A5, A4, A3, A2, A1, A0$  e  $B7, B6, B5, B4, B3, B2, B1, B0$ , respetivamente. Considere ainda a entrada de transporte  $Ci$ . A soma é traduzida pelas saídas  $S7, S6, S5, S4, S3, S2, S1$  e  $S0$ .
2. Verifique o funcionamento do circuito simulando-o. Observe que pode saber o valor lógico de qualquer sinal posicionando o cursor do rato sobre esse sinal.

Q2: Assumindo  $A=10010110$  e  $B=00110011$  indique o valor observado na saída  $S$ . Que valor resulta em  $S$  se  $A=01111110$  e  $B=00000010$ ?

Q3: Considere que numa simulação do circuito **Add8**  $A6=B6=1$ ,  $A7=1$  e  $B7=0$ . Qual o *carry* gerado nos componentes **FA** que produzem, respetivamente,  $S5$  e  $S6$ ?


<sup>1</sup>É neste grupo que encontrará os componentes que vai desenvolvendo.

### 3 Somador/subtrator de 8 bits

A subtração  $A - B$  pode ser realizada como  $A + (-B)$ , onde  $-B$  é o complemento para 2 de  $B$ . Note que o complemento para 2 de um número pode ser obtido pelo complemento lógico de cada bit ao qual se soma 1.

Pretende-se obter um circuito, com base no componente **Add8**, para realizar adições ou subtrações, consoante o valor lógico colocado numa entrada adicional **AS** que permitirá escolher a operação a realizar. Se **AS** for 0, as saídas deverão apresentar o resultado da adição,  $A + B$ ; se **AS** for 1, o valor apresentado nas saídas deverá ser o da subtração,  $A - B$ .

O valor de **AS** determina o valor a aplicar nas entradas **B7**, **B6**, **B5**, **B4**, **B3**, **B2**, **B1**, **B0** do componente **Add8**: se **AS=0**  $B$  não é alterado; se **AS=1**  $B$  é negado. Uma porta lógica útil nesta operação é o OU-exclusivo (XOR).

1. Mostre que o valor a aplicar em cada entrada  $B_i$  ( $i = 0, \dots, 7$ ) do componente **Add8** é dado por  $AS \oplus B_i$ . 
2. O valor a aplicar à entrada **Ci** de **Add8** depende da operação a realizar: na adição é 0 e na subtração é 1. Verifique que este é o valor de **AS**.
3. Inicie o desenho de um novo circuito, **AddSub8**, capaz de realizar adições e subtrações de 8 bits, usando o componente **Add8** e as portas lógicas adicionais necessárias à manipulação do operando  $B$ .
4. Considere três entradas:  $A$  e  $B$ , ambas com 8 bits, e **AS**. O resultado é colocado na saída **R**, também com 8 bits. Utilize barramentos para efetuar as ligações necessárias.
5. Verifique a operação do circuito, simulando a realização de várias adições e subtrações. Ao analisar o resultado tenha em consideração que está-se a considerar complemento para 2.

**Q4:** Coloque nas entradas  $A$  e  $B$  do circuito os valores 10000000 e 11111111 (representação em complemento para 2), respetivamente. A soma indicada na saída, 01111111, está errada. O circuito está mal construído ou atribui outro significado a esta situação?

### 4 Somador/subtrator de 8 bits com deteção de excesso

O componente **AddSub8** permite adicionar, ou subtrair, números representados em complemento para 2. Pretende-se obter um novo circuito capaz de detetar a ocorrência de excesso (*overflow*), ao realizar uma adição ou uma subtração, assinalando-a na saída **OVF**. Caso ocorra excesso **OVF** é 1 e no caso contrário **OVF** é 0.

Tal como aprendeu, a ocorrência de excesso depende do sinal dos operandos e do resultado, assim como da operação realizada (adição ou subtração). Defina numa tabela de verdade a função

$OVF(AS, sA, sB, sR)$ , em que  $AS$  indica a operação realizada e,  $sA$ ,  $sB$  e  $sR$  correspondem ao sinal dos operandos,  $A$  e  $B$ , e do resultado  $R$ , respetivamente.

1. Comece por acrescentar ao projeto um novo circuito designado por **det-OVF**.
2. Utilize a ferramenta de análise e geração de circuitos para definir a tabela de verdade da função  $OVF(AS, sA, sB, sR)$  e depois obter um circuito que realiza esta função (procedimento idêntico ao seguido no passo 1 da secção 1).
3. Desenhe um novo circuito, **AddSub8-ovf**, constituído pelos componentes **AddSub8** e **det-OVF**. As entradas e as saídas são as mesmas do circuito **AddSub8**, às quais se junta a saída **OVF**.
4. Simule o circuito e verifique se a deteção de excesso funciona corretamente.

Q5: Atribua às entradas **A** e **B** os valores 10011010 e 11101101, respetivamente, e analise os valores resultantes em **R** e **OVF**. Repita este procedimento para **A**=10000000 e **B**=11111111.

Q6: Este circuito produz o resultado **R** da adição/subtração e indica em **OVF** se ocorre excesso. Qual a utilidade da saída **OVF** para um circuito que use este componente **AddSub8-ovf**?

## 5 Acumulador de 8 bits

Uma aplicação comum dos somadores é na realização de circuitos capazes de somar uma sequência de valores. Guardando a soma num registo é possível usar este valor como entrada do somador. Aplicando na outra entrada do somador um novo valor a somar ao anterior obtém-se a atualização da saída que assim corresponde à acumulação dos valores que sucessivamente são aplicados à entrada. Um circuito destes designa-se por acumulador, sendo constituído por um somador e um registo. A atualização do valor armazenado no registo é feita à frequência do sinal de relógio que assim sequencia a operação de acumulação.

Os passos seguintes descrevem a obtenção de um acumulador de 8 bits.

1. Acrescente ao projeto um novo circuito designado por **Acc8**.
2. Inicie o desenho do circuito juntando-lhe os componentes **AddSub8-ovf** e **Reg8** (disponível em **Components** → **Custom**).
3. As entradas do circuito são **A**, **en**, **rst** e **clk**. **A**, com 8 bits, é o valor a acumular, **en** é o sinal de habilitação (*enable*) do registo, **rst** é o sinal de inicialização do registo e **clk** o respetivo sinal de relógio. À entrada **AS** do componente **AddSub8** deve ser ligado o valor adequado para este realizar a acumulação.

As saídas são **R** e **OVF**. **R**, com 8 bits, é o resultado da acumulação, e **OVF** indica se ocorre excesso.

4. Simule o funcionamento do circuito, verificando não só a acumulação de uma sequência de valores, mas também a atuação das entradas de enable e reset.

Q7: Este circuito utiliza o componente **AddSub8-ovf** com a entrada **AS=0**. Que simplificações poderiam fazer-se?

Q8: Como poderá alterar o circuito de modo a que após ter ocorrido excesso (**OVF=1**) a saída **R** não volte a ser alterada mesmo que se queira somar mais valores?  
Implemente e simule a solução encontrada.

## 6 Acumulador com sequenciador de dados

Como alternativa a indicar um a um os valores a somar durante uma simulação, pode construir-se um circuito capaz de sequenciar o fornecimento destes valores. Nesta etapa do projeto vai construir-se um circuito com uma memória, onde previamente se armazenam valores. O acesso aos valores é obtido com um contador que gera sequencialmente os endereços de memória onde os valores estão armazenados. Desta forma, para somar-se uma sequência de valores com o circuito acumulador apenas é necessário provocar as transições do sinal de relógio por forma a que em cada ciclo seja lido um valor de memória e somado ao valor acumulado até ao momento.

1. Crie um novo circuito, designando-o por **Acc8-seq**, utilizando os componentes **ROM** (bloco de memória apenas de leitura) e **Counter** (contador binário), disponíveis em **Components** → **Memory**, e **Acc8** (acessível em **Components** → **Custom**).
2. Para dimensionar a memória clique o botão direito do rato sobre **ROM** e indique 8 bits para os dados e 4 bits para os endereços. Desta forma a memória tem capacidade para armazenar  $2^4=16$  valores de 8 bits. Deverá fazer o mesmo para o contador escolhendo 4 bits para a saída, uma vez que é este componente que vai gerar a sequência de endereços da memória de onde são lidos os valores a acumular em **Acc8**.
3. Ligue a saída **out** do contador à entrada **A** (endereço) da memória e a saída **D** (dados) da memória à entrada **A** do acumulador.
4. As entradas do circuito (**Acc8-seq**) são **en**, **rst** e **clk**, com o significado já descrito a propósito do circuito **Acc8**, e as saídas são **R** e **OVF**. Efetue todas as ligações necessárias ao fluxo de dados no circuito e respetivas operações de controlo (habilitação e inicialização).
5. Defina o conteúdo da memória clicando o botão direito do rato sobre **ROM** e escolha **Edit**. Após editar o conteúdo pode gravá-lo.
6. Faça a simulação do circuito por forma a verificar a correta operação do circuito.

Q9: Como modificar este circuito para que eventuais valores negativos armazenados na memória não sejam somados pelo acumulador?


Q10: Como modificar este circuito para impedir a leitura de mais valores da memória ROM após nela ser encontrado o valor 01010101?

## 7 Acumulador com sequenciador de dados num sistema de memória

Nesta última etapa do projeto vai considerar-se um circuito, resultante da evolução do circuito anterior, que permite sequenciar dados a partir de um sistema de memória composto por duas memórias. Pretende-se com este sistema, de pequena dimensão, exemplificar o acesso aos dados das memórias a partir de qualquer endereço válido.

1. Abra em janelas separadas os circuitos **smem** e **Acc8-smem** (fazem parte do conteúdo extraído do arquivo no início do trabalho).
2. Analize o circuito **smem** e responda às questões seguintes.

Q11: Determine a capacidade das memórias ROM1 e ROM2 em bytes.

Q12: Apresente o intervalo de endereços a que as memórias respondem, e indique se a descodificação de endereços de cada memória é total ou parcial. 

3. Complete o circuito **Acc8-smem** com o componente em falta (o seu acumulador **Acc8**). Edite o conteúdo das memórias ROM1 e ROM2 ou carregue-o a partir dos ficheiros ROM1.hex e ROM2.hex. Simule atentamente a operação do circuito **smem** considerando os dados contidos nas memórias.

Q13: Mostre como operar o circuito para obter em cada ciclo de relógio, na saída de **smem**, o conteúdo da memória correspondente aos endereços 00, 01, 02, 03, 80, 81, 82, A0 e A1.

Q14: Explique a utilidade do AND que gera o sinal de habilitação do acumulador (**Acc8**).