Universidade Federal da Paraíba - UFPB

Engenharia da computação

Centro de Informática

Concepção Estruturada de Circuitos Integrados

Estrutura e implementação do ADDAC_4 (Fase 1)

Professor: Antônio Carlos Cavalcanti

Aluno: Johan Kevin Estevão de Freitas

Matrícula: 20170171741

2020

Resumo

Neste relatório, serão apresentados a modelagem estrutural o processo de organização e implementação de um ADDAC_4, mostrando assim a organização dos arquivos, a linguagem e o processo de implementação dos blocos: Inversor, Somador, Multiplexador e Acumulador. Assim como a geração de resultados em relação a entrada do circuito completo.

Introdução

A organização dos arquivos responsáveis pela simulação e testes da **fase 1** é mostrado na figura abaixo ai como pedido na especificação da atividade.

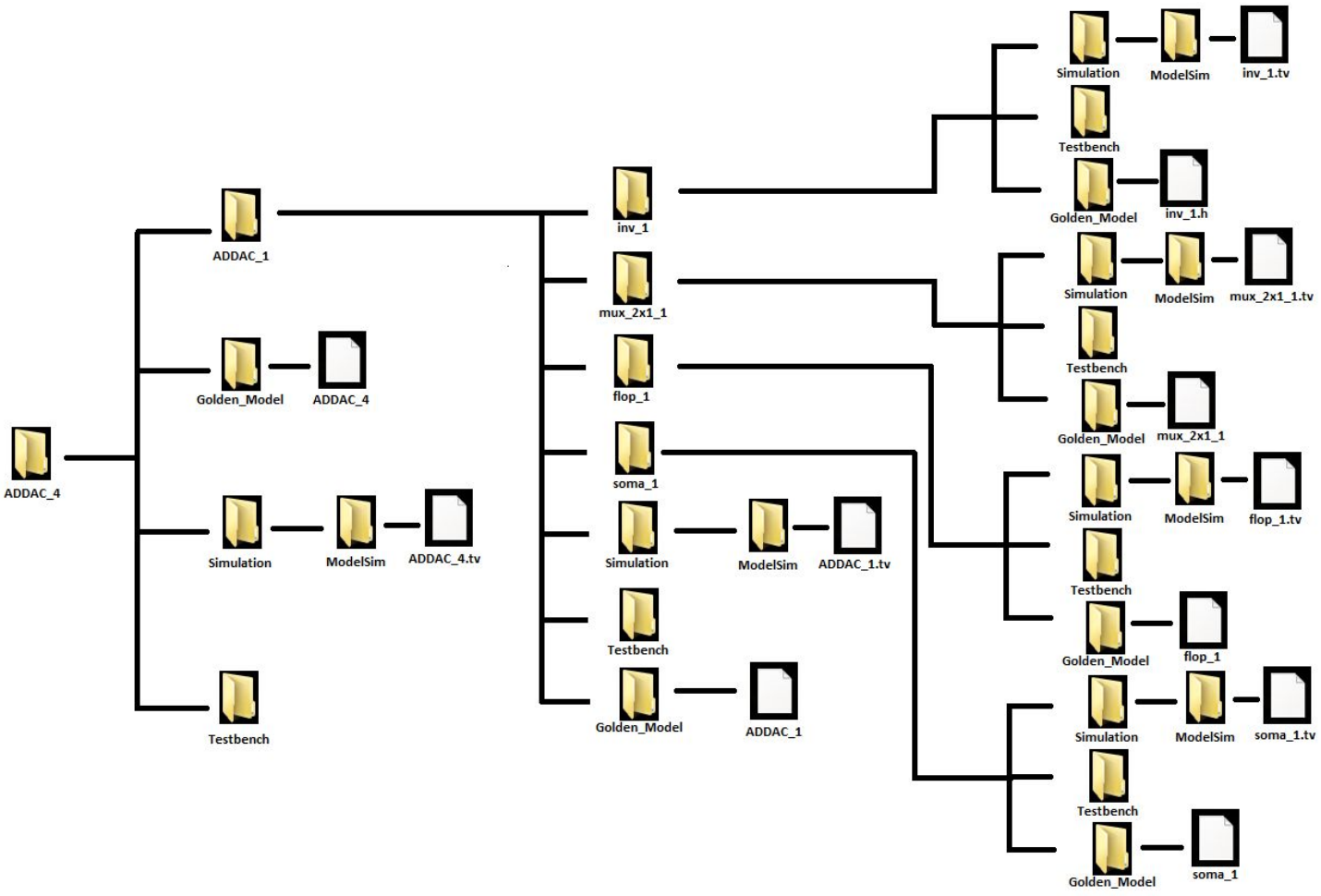

Figura 01 - Organização dos arquivos de implementação.

ADDAC

O Sistema lógico do ADDAC realiza quatro funções: cópia, soma, subtração e inversão, através da acumulação de 4 bits. Para isso são utilizados conexões entre 2 multiplexadores, 1 inversor, 1 somador completo e 1 acumulador, como ilustra a Figura 02.

Na primeira fase do projeto de implementação do ADDAC trabalharemos com a construção de blocos básicos de 1 bit, montaremos estruturalmente o ADDAC_1bit instanciando os blocos básicos e no final, também estruturalmente, juntamos 4 blocos de 1 bit para montar o projeto final do ADDAC com 4 bits.

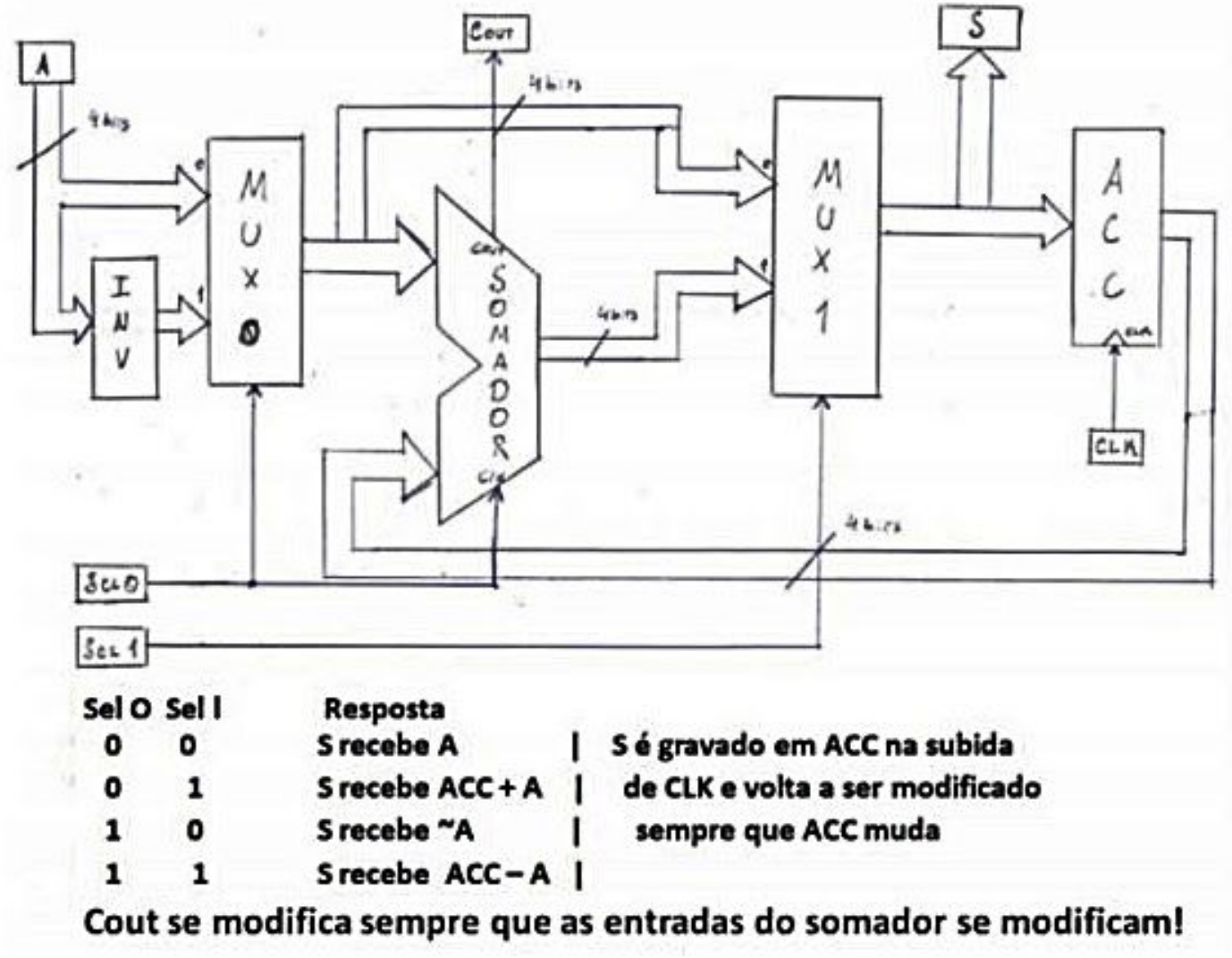

Figura 02 - Estrutura lógica do ADDAC_4.

Implementação

Os passos para implementar a concepção estrutural do ADDAC_4 (Fase 1) será a elaboração e implementação do modelo de ouro (Golden Model) em qualquer linguagem de programação e também a simulação ModelSim analisando os resultados gerados em arquivos .tv

O Modelo de Referência de Ouro, do inglês Golden Reference Model, trata-se de um modelo de referência em alto nível de um dado circuito capaz de descrever seu comportamento, isto é, suas saídas, em função das suas entradas. Esse modelo é usado para produzir um vetor de casos de teste que exaustivamente compara se o resultado produzido pelo circuito está de acordo com o vetor de testes gerado.

Modelos de Ouro dos blocos lógicos Inversor, Multiplexador, Somador, Acumulador, implementados em linguagem C:

inv_1.h

```
#ifndef inv_1_H_INCLUDED
#define inv_1_H_INCLUDED

#include <stdbool.h>

int inversor(int a){

  FILE *arquivo;
  int aux=!a;

  arquivo = fopen("inv_1/Simulation/ModelSim/inv_1.tv", "a");
  //fprintf(arquivo, "//Inversor\n//Entrada_Saida\n");
  fprintf(arquivo, "%d_%d\n", a, aux);
  return aux;
  fclose(arquivo);
}
#endif
```

Figura 03 -Modelo de Referência do inversor - inv_1.h.

mux_2x1_1.h

```
#ifndef mux_2x1_1_H_INCLUDED
#define mux_2x1_1_H_INCLUDED

int mux(int a, int b, int Sel){

  FILE *arquivo;
  arquivo = fopen("mux_2x1_1/Simulation/ModelSim/mux_2x1_1.tv", "a");

  //fprintf(arquivo, "//MUX\n//Sel_Saida\n");
  fprintf(arquivo, "%d_%d_%d_", Sel,a,b);

  if(!Sel){
        fprintf(arquivo, "%d\n", a);
        return a;
  }
  else{
        fprintf(arquivo, "%d\n", b);
        return b;
  }
  fclose(arquivo);
}
#endif
```

Figura 04 - Modelo de Referência do Multiplexador - mux_2x1_1.h.

soma_1.h

```
#ifndef soma_1_H_INCLUDED
#define soma_1_H_INCLUDED
#include "carry_out_function.h"
#include <string.h>

#define BIT 2

int somador_completo(int a, int b, int carry_in){

  FILE *arquivo;
  arquivo = fopen("soma_1/Simulation/ModelSim/soma_1.tv", "a");
  //fprintf(arquivo,"//Somador\n//A_B_Cin_Cout_Soma\n");
  fprintf(arquivo, "%d_%d_%d_", a,b,carry_in);
  int aux = 0, carry_out = 0;

  int soma = a+b+carry_in;
  //char soma_bin[BIT];
  //int carry_out = carry_out_function(soma);
  //itoa(soma, soma_bin, 2);//transforma a soma p binario

  if(soma==0){
    fprintf(arquivo, "0_0\n");
    aux=0;
    carry_out = 0;
  }
  else if (soma==1){
    fprintf(arquivo, "0_1\n");
    aux=1;
    carry_out = 0;
  }
```

Figura 05 -Modelo de Referência do Somador - soma_1.h.

```
  else if(soma==2){
    fprintf(arquivo, "1_0\n");
    aux=0;
    carry_out = 1;
  }
  else if (soma==3){
    fprintf(arquivo, "1_1\n");
    aux=1;
    carry_out = 1;
  }
  else{
    aux= -5;
  }

  fclose(arquivo);
  return aux;
}
#endif
```

Figura 06 -Modelo de Referência do Somador parte 2.

carry_out_function.h

```
#ifndef CARRY_OUT_FUNCTION_H_INCLUDED
#define CARRY_OUT_FUNCTION_H_INCLUDED

int carry_out_function(int soma){
    if(soma>1){
        return 1;
    }
    else return 0;
}
#endif
```

Figura 07 -Estrutura Auxiliar do Modelo de Referência do carry out - carry_outfunction.h.

flop_1.h

```
#ifndef Flop_1_H_INCLUDED
#define Flop_1_H_INCLUDED

int acc(int clk_a, int clk, int a, int acumulado){
    FILE *arquivo;
    int y =0;
    arquivo = fopen("flop_1/Simulation/ModelSim/Flop_1.txt", "a");
    fprintf(arquivo, "%d_%d_%d_%d_", clk_a, clk, a, acumulado);

  //fprintf(arquivo, "//acc\n//clk_a clk saida\n");

    if(clk_a==1 & clk==0)//borda de descida
      y=a;
    else
        y=acumulado;//valor anterior

    fprintf(arquivo, "%d\n", y);
    return y;

    fclose(arquivo);

}
#endif
```

Figura 08 - Modelo de Referência do Acumulador - Flop.h.

Simulação e análise dos “Arquivos.tv”

No escopo de cada código .h é executado o processo de escrita em um arquivo salvo na extensão “.tv” que futuramente irá ser utilizada como o arquivo de verificação para o comportamento do circuito implementado em verilog. Para obter todas as possibilidades de saída para cada bloco na função principal deste programa (main.c) foi realizada a variação dos parâmetros de todas as funções implementadas em .h. Nas figuras abaixo serão mostrados os vetores de testes escritos nos arquivos .tv.

inv_1.tv

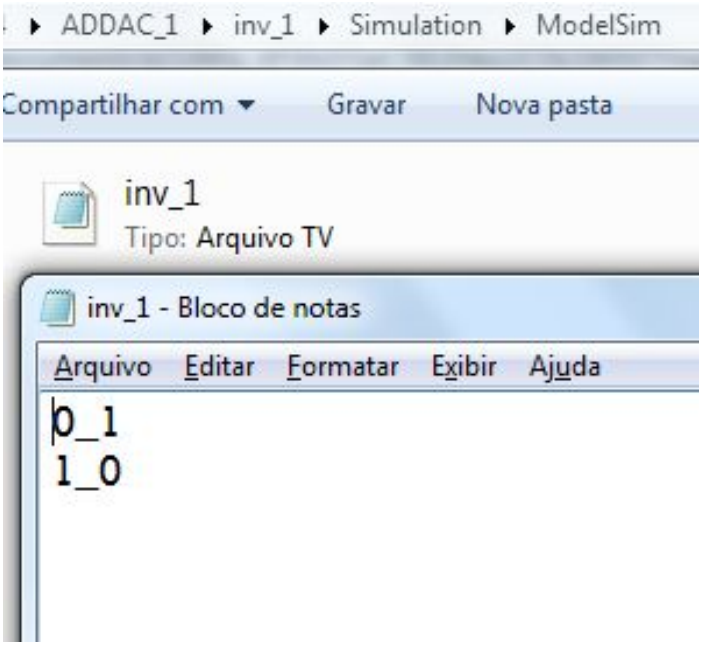

Figura 9 - Arquivo.tv do Inversor - inv_1.tv.

mux_2x1_1.tv

O modelo de referência do arquivo mux.tv foi escrito com o bit equivalente a chave de seleção seguido das duas variáveis de entrada do mux e finalmente a variável de saída do bloco lógico. Dessa forma, sempre que o primeiro bit for equivalente a 0 a saída (o último bit) tem que ser igual ao bit equivalente a entrada A, ou seja, o segundo bit. Assim, a saída é igual ao segundo bit. Porém, se a chave de seleção estiver em 1 a saída é igual ao B.

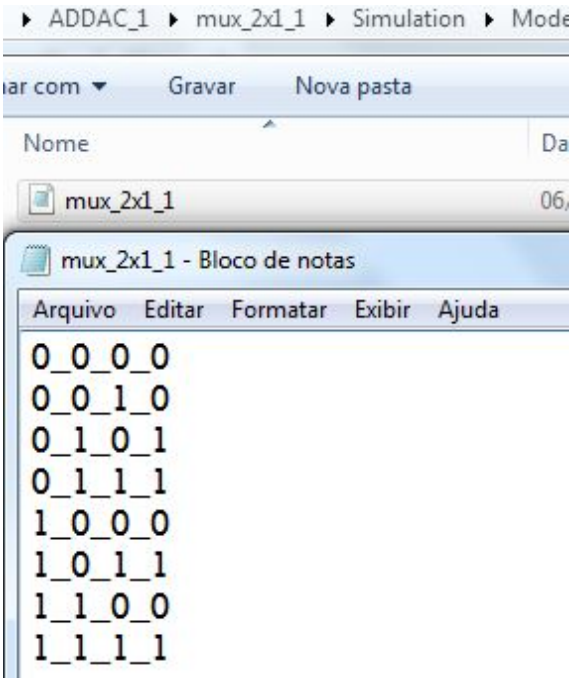

Figura 10 - Arquivo.tv do Multiplexador - mux_2x1_1.tv.

soma_1.tv

O modelo de referência do arquivo somador.tv foi escrito da seguinte forma: dado o bit referente a chave de seleção, em seguida são adicionadas as entradas A, B e o Carry in seguidas de underscore e por fim, é ilustrado o bit de estouro ou carry out e a saída equivalente ao resultado da operação realizada pelo somador. Para o último caso de análise, em que todas as entradas estão em nível lógico alto, ambas variáveis da saída também são 1.

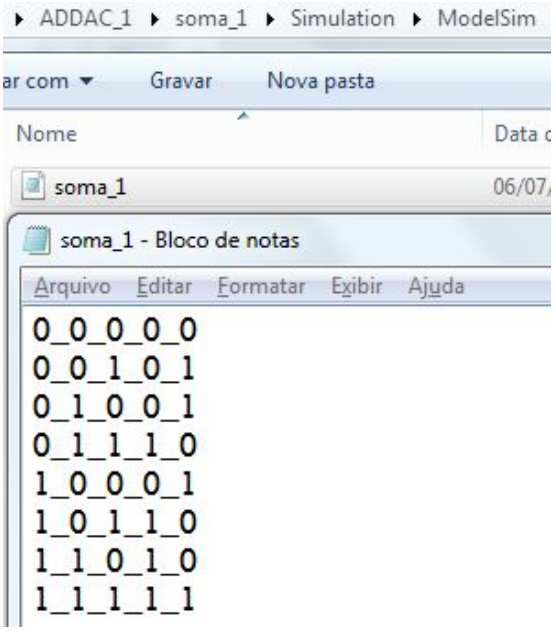

Figura 11 - Arquivo.tv do Somador - soma_1.tv.

flop_1.tv

O acumulador tem por função receber o sinal de entrada e só atualizar sua saída quando o clock estiver na borda de descida. Para este golden model foram utilizadas duas variáveis auxiliares que permitem controlar a borda do clock. A borda de descida é atingida quando o clock anterior está em 1 e o clk (atual) está em 0. Desse modo, apenas quando o acumulador recebe o clk_a como 0 e clk como 1 ele possui a sua saída igual a entrada. A variável ACC se refere ao valor que está na entrada do acumulador porém a saída do bloco lógico só será igual a ele nas linhas 6 e 7, visto que o clock está na borda de descida.

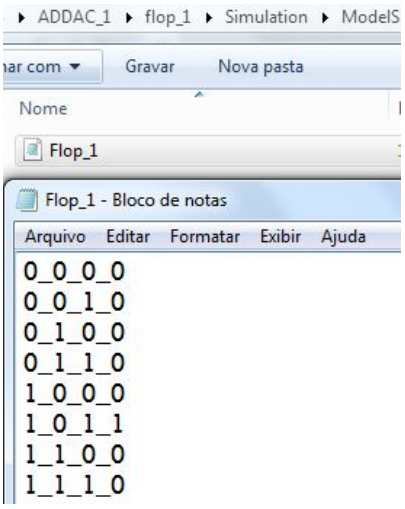

Figura 12 - Arquivo.tv do Acumulador - Flop_1.tv.

ADDAC_1.tv

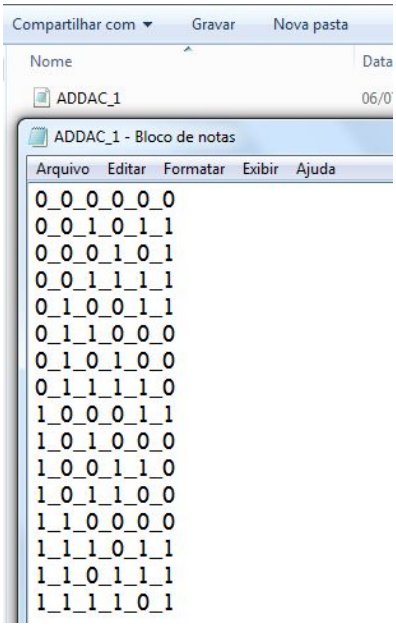

Figura 13 - Arquivo.tv do ADDAC_1 - ADDAC_1.tv.

O modelo de referência foi elaborado da seguinte forma: Sel_0, Sel_1, entrada, CLK, acumulado e saída.