Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Introdução aos Sistemas Lógicos

Trabalho Prático 1: Processamento de Dados de Sensores Binários para o Tesla Autopilot

Henrique Gomes dos Santos Medeiros
2021084986

Belo Horizonte
2024

Introdução

Este documento apresenta a implementação de um circuito combinatório em Verilog destinado ao processamento de dados provenientes dos sensores do Tesla Autopilot. Esses sensores emitem leituras binárias de 4 bits representando a distância até obstáculos, e o sistema precisa calcular o resto da divisão por 5 dessas leituras.

O objetivo do circuito é permitir que o Tesla Autopilot priorize a análise de dados mais críticos para a tomada de decisões em tempo real, garantindo uma resposta ágil e precisa. A abordagem utiliza lógica combinatória, explorando a relação direta entre as entradas dos sensores e o resultado processado.

Especificação do circuito

- O circuito deve ter uma entrada de dados binários de 4 bits, representando as leituras dos sensores.
- Ele deve ter uma saída de 3 bits, que é o resto da divisão por 5 da entrada.
- A divisão por 0 não será considerada neste problema.

Decisões de implementação

Pela similaridade do Verilog com C e C++, linguagens que já estou habituado a trabalhar, não foi difícil compreender a sintaxe básica para executar a simples tarefa de retornar o resto da divisão por 5.

Não havia necessidade de muitos registradores para armazenar valores, então utilizei apenas **reg sensor_input** para tal, e alterei seu valor ao longo do código de testbench.sv para realização dos test cases. Foi utilizado **wire** como tipo padrão para entradas e saídas, e utilizado **assign** como mecanismo padrão para atribuição de valores para valorar a saída.

Um testbench foi criado para fornecer entradas predefinidas e monitorar as saídas do circuito, e os casos de teste como entradas foram determinados conforme pedia o enunciado. A simulação foi realizada na ferramenta EDA Playground, gerando formas de onda e logs para validar o comportamento do circuito.

Estruturei o código com boa organização e comentários claros, para facilitar a leitura e compreensão. Nomes intuitivos foram usados para variáveis e módulos, seguindo boas práticas gerais de programação.

Optei por uma unidade de tempo de 1 nanosegundo (1ns). Simulações com sensores e sistemas embarcados geralmente utilizam tempos curtos para refletir as operações rápidas de hardware. A precisão foi definida como 1 picosegundo (1ps) pois isso permite representar eventos em tempos muito curtos com alta resolução, essencial para capturar transições precisas de sinais no circuito. Sensores de sistemas avançados como o Tesla Autopilot requerem alta precisão para garantir reações em tempo real. Mesmo que o circuito seja combinatório, o simulador deve refletir as condições de tempo próximas das reais.

Forma canônica de cada saída usando notação Σm ou ΠM

∑m(5, 10, 15)	**A'BC'D + AB'CD' + ABCD**
∑m(1, 6, 11)	**A'B'C'D + A'BCD' + AB'CD**
∑m(2, 7, 12)	**A'B'CD' + A'BCD + ABC'D'**
∑m(3, 8, 13)	**A'B'CD + AB'C'D' + ABC'D**
∑m(4, 9, 14)	**A'BC'D' + AB'C'D + ABCD'**

Forma simplificada de cada saída como um Produto de Somas (PoS)

Resto 0: ΠM(1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14)
(A+B+C+D') (A+B+C'+D) (A+B+C'+D') (A+B'+C+D) (A+B'+C'+D) (A+B'+C'+D') (A'+B+C+D) (A'+B+C+D') (A'+B+C'+D') (A'+B'+C+D) (A'+B'+C+D') (A'+B'+C'+D)

Resto 1: ΠM(2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15)
(A+B+C'+D) (A+B+C'+D') (A+B'+C+D) (A+B'+C+D') (A+B'+C'+D') (A'+B+C+D) (A'+B+C+D') (A'+B+C'+D) (A'+B'+C+D) (A'+B'+C+D') (A'+B'+C'+D) (A'+B'+C'+D')

Resto 2: ΠM(1, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 15)
(A+B+C+D') (A+B+C'+D') (A+B'+C+D) (A+B'+C+D') (A+B'+C'+D) (A'+B+C+D) (A'+B+C+D') (A'+B+C'+D) (A'+B+C'+D') (A'+B'+C+D') (A'+B'+C'+D) (A'+B'+C'+D')

Resto 3: ΠM(1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15)

(A+B+C+D') (A+B+C'+D) (A+B'+C+D) (A+B'+C+D') (A+B'+C'+D) (A+B'+C'+D') (A'+B+C+D') (A'+B+C'+D) (A'+B+C'+D') (A'+B'+C+D) (A'+B'+C'+D) (A'+B'+C'+D')

Resto 4: ΠM(1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 15)
(A+B+C+D') (A+B+C'+D) (A+B+C'+D') (A+B'+C+D') (A+B'+C'+D) (A+B'+C'+D') (A'+B+C+D) (A'+B+C'+D) (A'+B+C'+D') (A'+B'+C+D) (A'+B'+C+D') (A'+B'+C'+D')

Mapas de Karnaugh

As saídas possíveis para o programa são respostas de 0 a 4. Assim, chamaremos cada uma de Output 0 até Output 4. Consideraremos como entradas os valores de 1 a 15 (lembrete: 0 desconsiderado nesse problema).

NOTA: existem duas interpretações possíveis para o que chamamos de saídas do programa: uma interpretação se refere aos números de 0 a 4, e outra se refere aos 3 bits de saída do programa. De outro modo, ou são 5 mapas de Karnaugh, ou são 3. Optei pela primeira versão, e deixei essa nota porque pensei nisso depois e fiz uma escolha de acordo com o que me parece mais certo.

Output 0 (5, 10, 15)

AB\CD	00	01	11	10
00	X	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

Output 1 (1, 6, 11)

AB\CD	00	01	11	10
00	X	1	0	0
01	0	0	0	1
11	0	0	0	0
10	0	0	1	0

Output 2 (2, 7, 12)

AB\CD	00	01	11	10
00	X	0	0	1
01	0	0	1	0
11	1	0	0	0
10	0	0	0	0

Output 3 (3, 8, 13)

AB\CD	00	01	11	10
00	X	0	1	0
01	0	0	0	0
11	0	1	0	0
10	1	0	0	0

Output 4 (4, 9, 14)

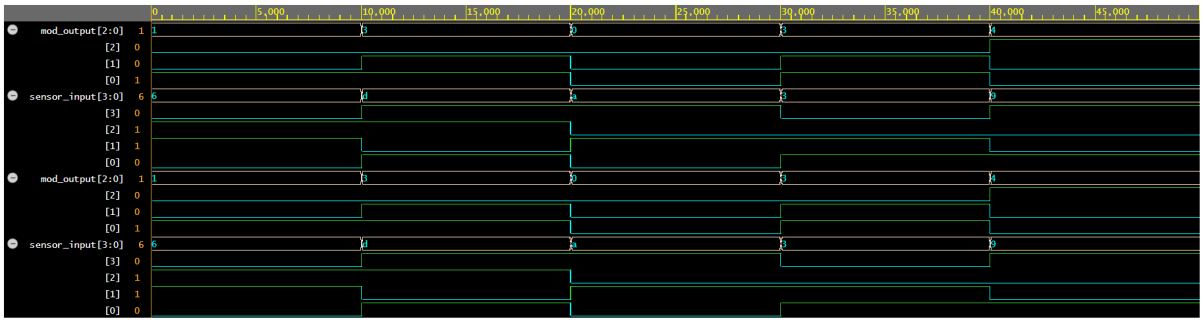
AB\CD	00	01	11	10
00	X	0	0	0
01	1	0	0	0
11	0	0	0	1
10	0	1	0	0

Tabela verdade

Entrada (sensor_input**)**	**Decimal**	**Saída (**mod_output**)**
0000	X	X
0001	1	001
0010	2	010

0011	3	011
0100	4	100
0101	5	000
0110	6	001
0111	7	010
1000	8	011
1001	9	100
1010	10	000
1011	11	001
1100	12	010
1101	13	011
1110	14	100
1111	15	000

Visualização das formas de onda do caso de teste



Referências

Verilog – Wikipédia, a enciclopédia livre
https://www.cin.ufpe.br/~eaa3/Arquivos/Verilog/Tutorial%20Verilog.pdf
Aulas, notas e materiais fornecidos pelo curso.