Centro Federal de Educação Tecnológica de Minas Gerais

CEFET-MG

Engenharia de Computação
Arquitetura e Organização de Computadores II
Prática II

Aluno 1: Ana Carolina
Aluno 2: Bernardo Meneghini

Introdução

Nesta prática, apresentaremos um processador simples descrita na linguagem Verilog, os módulos necessários para a implementação do mesmo, com as respectivas simulações.

Desenvolvimento

Tomamos como base, o circuito descrito pela Imagem 1, que pode ser encontrada no arquivo auxiliar em inglês disponibilizado no Moodle. Escolhemos instruções de 16 bits, e, portanto, registrados de 16 bits (para a saída).

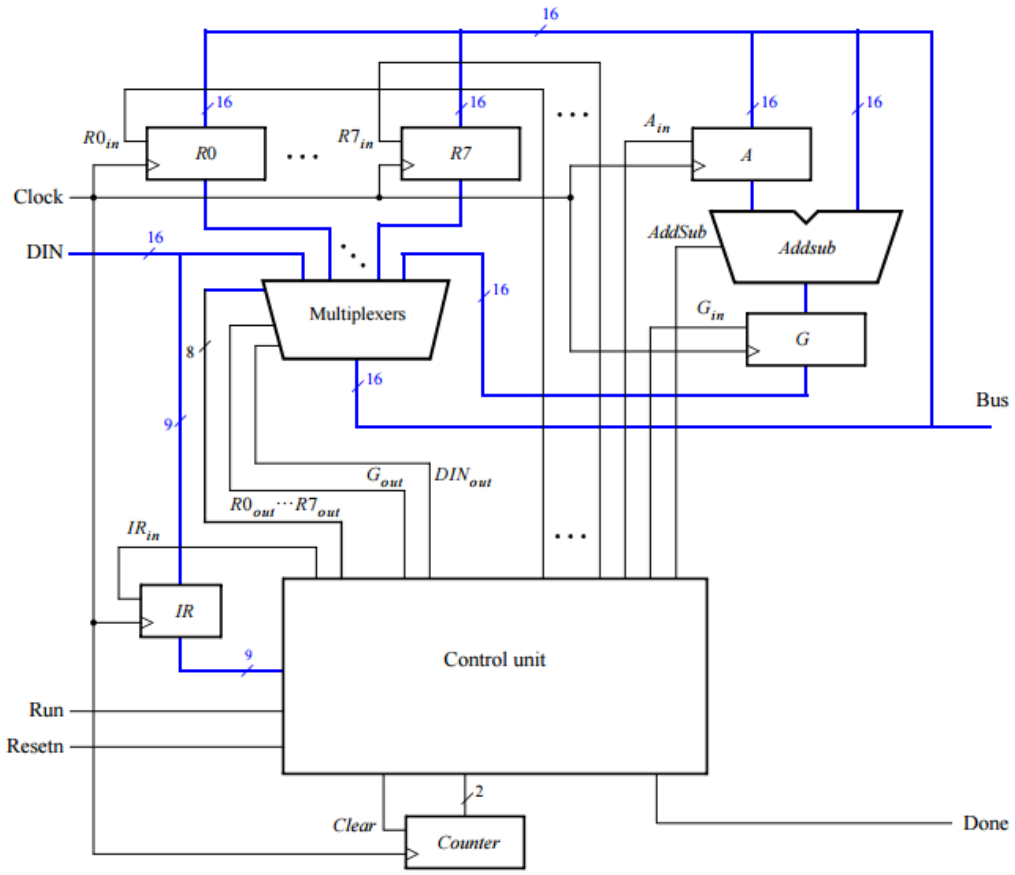


Figure 1. A digital system.

Imagem 1. Diagrama Processador

Para atendar as especificações exigidas, estabelecemos um OPCODE de 3 bits, pois temos no total 8 operações. Os próximos 6 bits são destinados ao registrador de destino e registrador fontes, respectivamente. Os demais bits foram colocados como “don’t care”. Esta situação está sendo mostrada na Imagem 2.

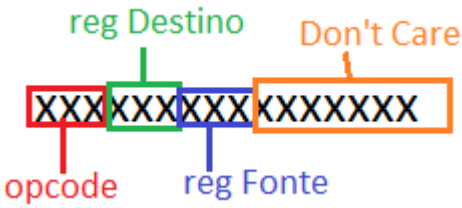


Imagem 2. Divisão da Instrução

As instruções implementadas no processador foram as seguintes:

OPCODE	Instrução
000	MV
001	MVI
010	ADD
011	SUB
100	OR
101	SLT
110	Shift Left
111	Shift Right

Tabela 1. Instruções

Para um processador realizar operações como ADD, SUB, SLT, SL e SR, uma ULA é necessária (e não somente um módulo de ADDSUB, como mostrado no diagrama). Implementamos uma então e simulamo-la para todas as situações. Temos que nos certificar que a implementação desta seja perfeita, para não acarretar em erros ao final do projeto. Todas as simulações realizadas estão sendo apresentadas na sessão de Simulações.

Finalizando a parte de documentação e planejamento, faltava apenas realizar a conexão entre o processador e uma memória, que no caso, é do tipo ROM. A implementação desta parte foi baseada também no arquivo auxiliar em inglês disponibilizado no Moodle, e a Imagem 3 pode ser encontrada lá também.

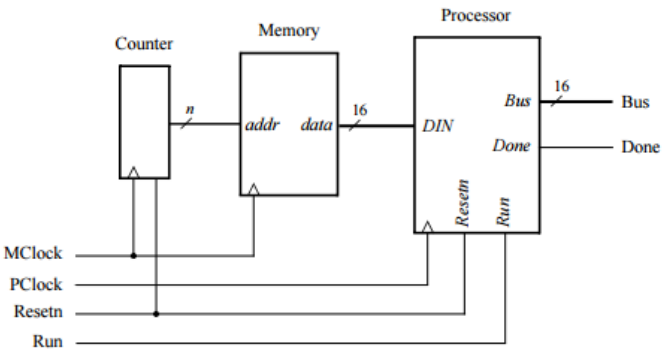
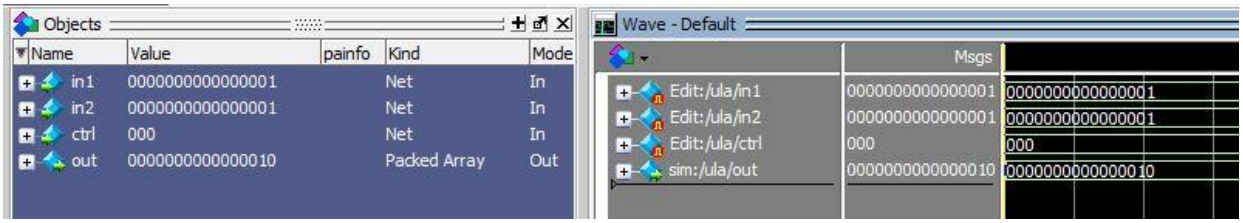


Imagem 3. Interconexão Processador-Memória ROM

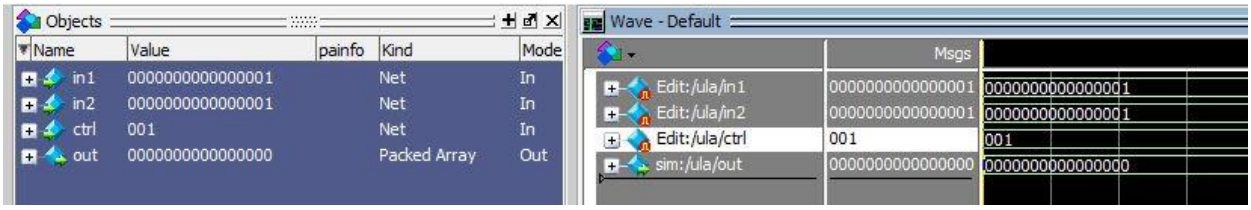
Simulações

Todos os módulos desse projeto foram simulados, a fim de verificar se o comportamento dos mesmos estava dentro do esperado. Cada simulação aqui apresentada será discutida individualmente.



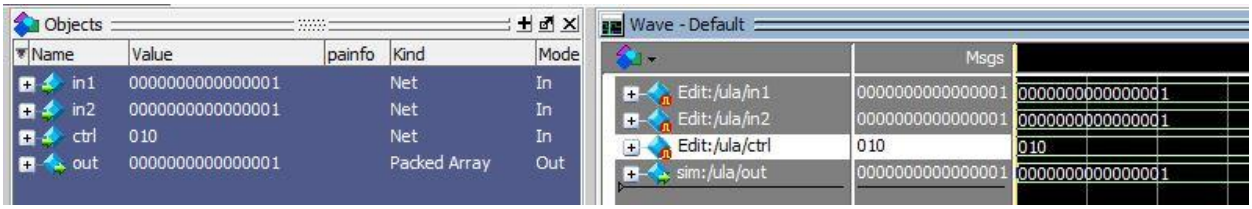
Simulação 1. ULA ADD

Nesta simulação, entramos com 1 e 1, e a saída retornou 2, como esperado.



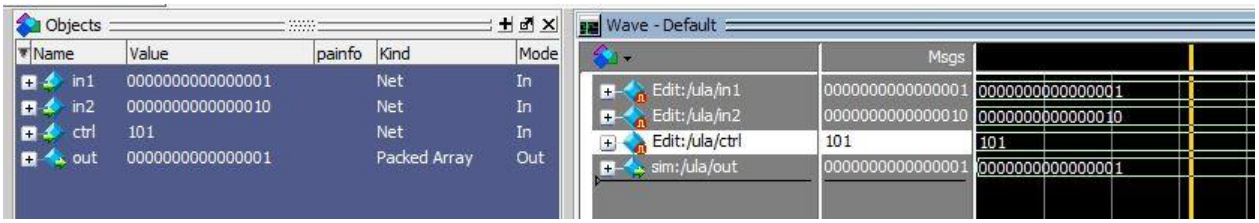
Simulação 2. ULA SUB

Entramos com 1 e 1, e a saída retornou 0, como previsto.



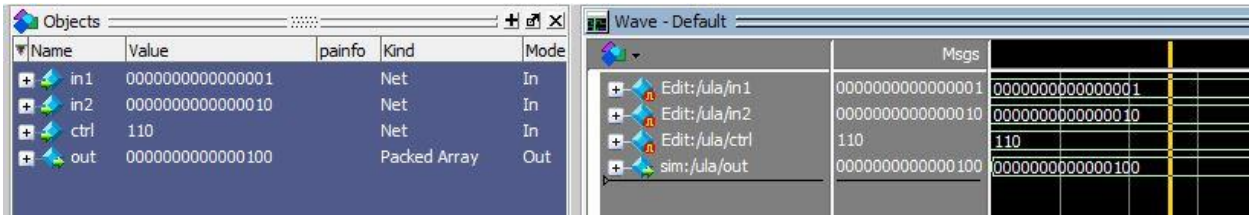
Simulação 3. ULA OR

Novamente, entramos com 1 e 1, e a saída retornou 1, como esperado, pois 1 OR 1 resulta em 1.



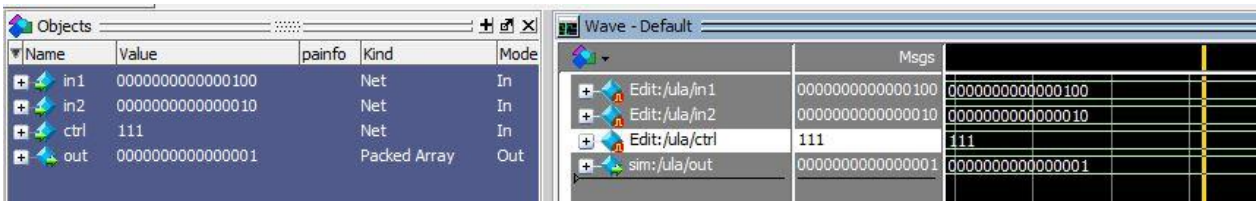
Simulação 4. ULA SLT

Entrando com 1 e 2, e obtivemos 1 na saída, pois 1 < 2, como esperado.



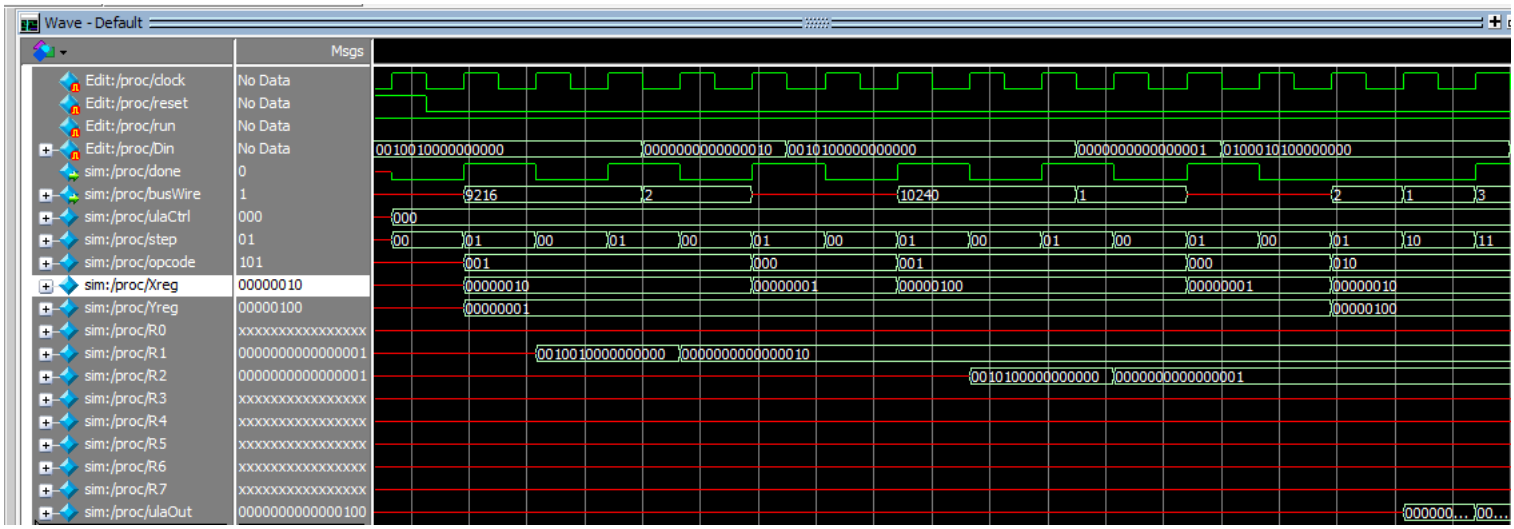
Simulação 5. ULA SHIFT LEFT

Entramos com 1 e 2 novamente, e obtivemos 4, como esperado



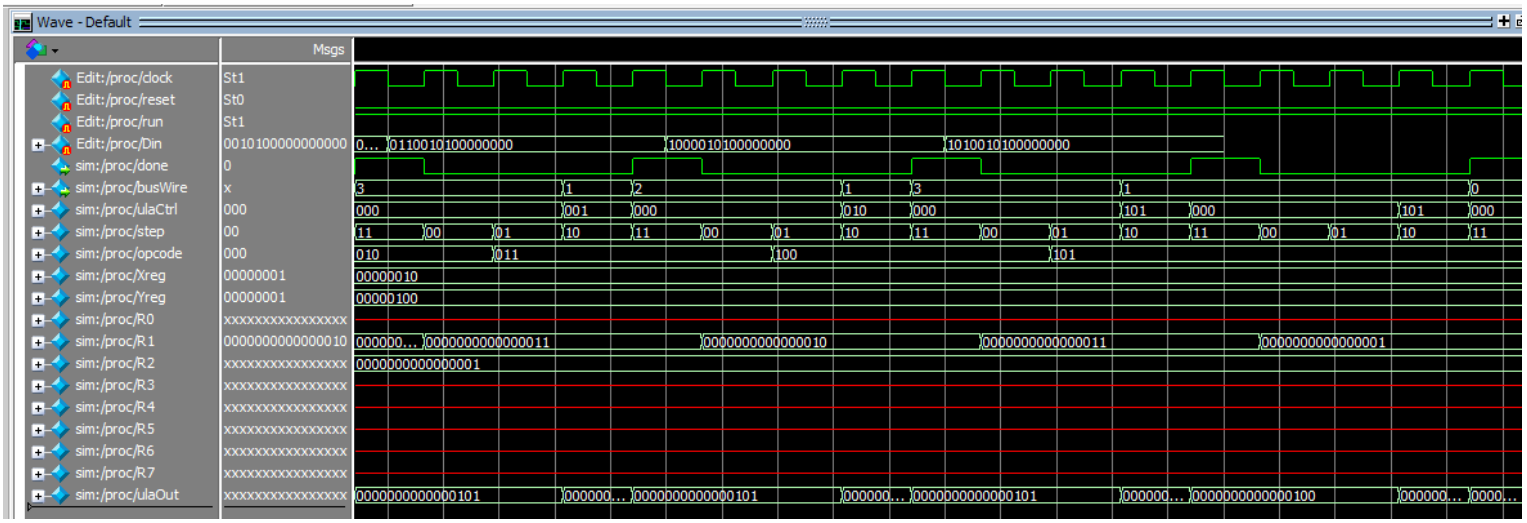
Simulação 6. ULA SHIFT RIGHT

Finalizando com a ULA, entramos com 4 e 2, e obtivemos 1, como estávamos esperando. Portanto, TODAS as operações na ULA estão funcionando corretamente. A seguir, apresentaremos as simulações do processador com instruções pré construídas. As instruções utilizadas aqui, foram documentadas em um arquivo .txt que pode ser encontrado na pasta do projeto, sob o nome de “Instruções.txt”.



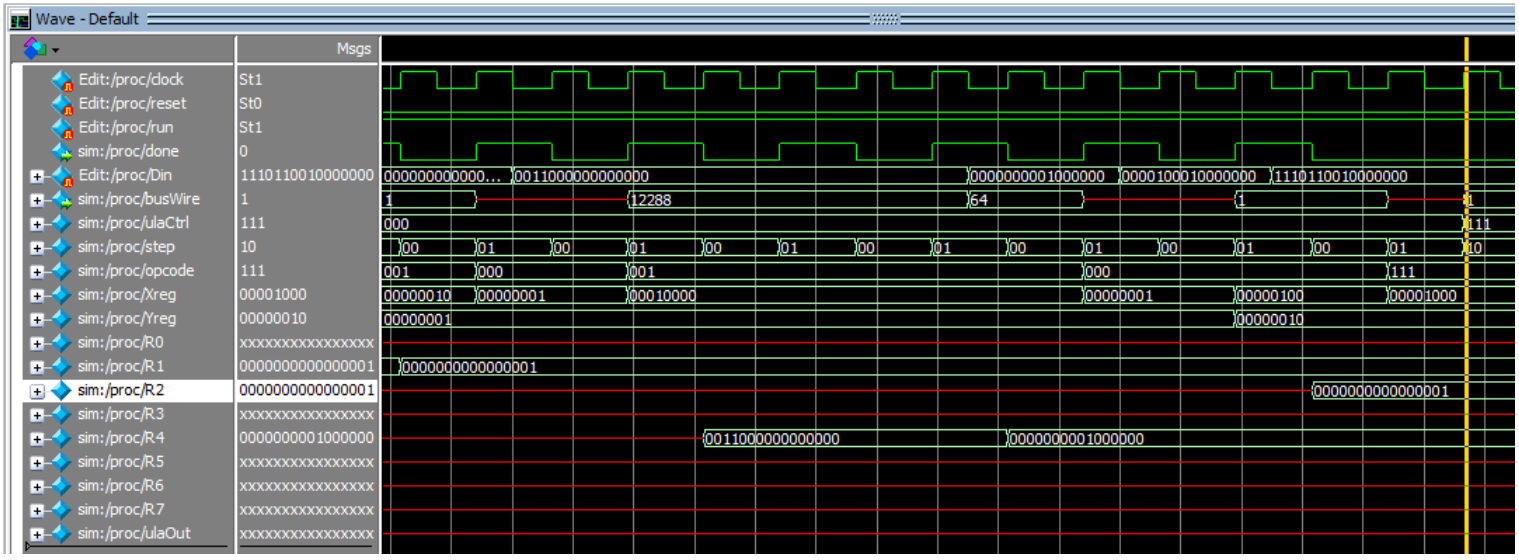
Simulação 7. Processador – Instruções MVI, ADD

A Simulação 7 consiste primeiramente, numa instrução de MVI, armazenando no registrador 1 o valor de 2. Para fins práticos, adicionamos em todas as simulações do processador as ondas referentes aos Registradores (na figura, R1, R2, R3, R4, R5, R6, R7), a fim de acompanhar o conteúdo dos mesmos e verificar se o fluxo está de acordo com o esperado. Observamos então, que o R1 passa a conter o valor de 2. Outra instrução de MVI foi realizada, armazenando o valor de 1 no R2 (observe que R2 passa a conter 2 após o fim da instrução). Uma instrução de ADD foi então processada, fazendo R1 + R2 e armazenando o valor em R1, que no caso, é 3. Verifique que a saída (BusWire) contém exatamente este valor.



Simulação 8. Processador – Instruções SUB, OR, SLT

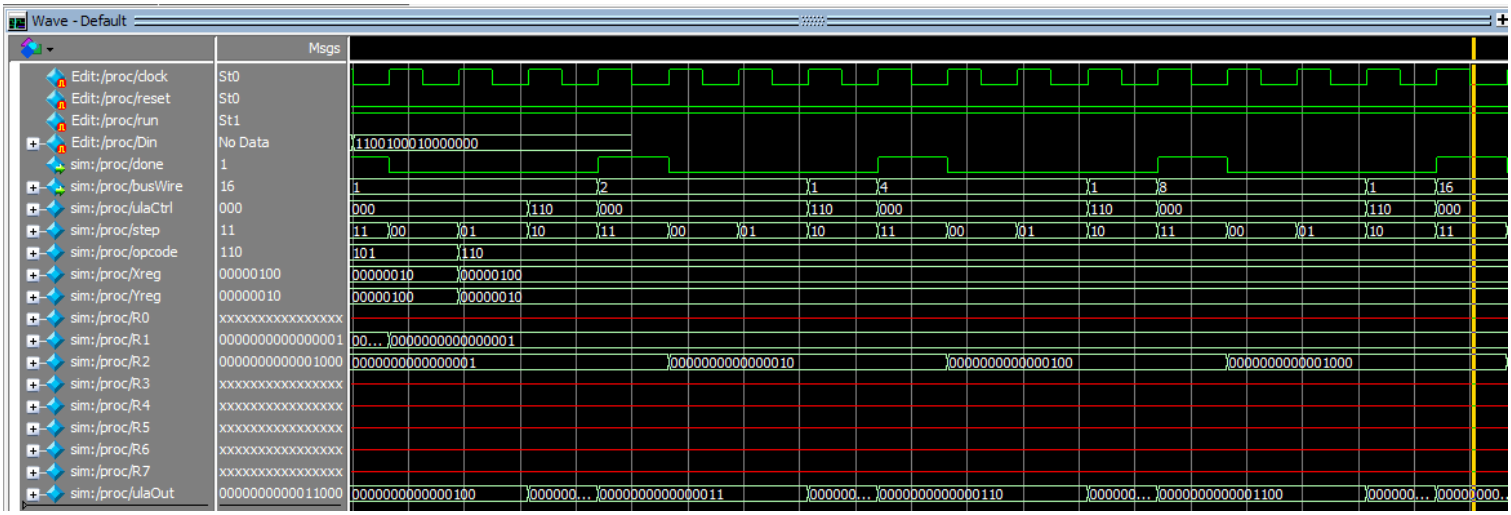
A próxima instrução é um SUB entre os registradores R1 e R2, armazenando o resultado em R1. Podemos observar que, o conteúdo prévio de R1 era 3, devido à instrução de soma realizada no passo anterior. Ao subtrair este valor pelo conteúdo de R2, que é 1 (não foi alterado pelas instruções anteriores), R1 recebe agora novamente o valor de 2. Logo então, uma instrução de OR é executada entre os registradores R1 e R2, comparando os valores 10 binário (equivalente a 2 decimal) e 01 binário (equivalente a 1 decimal). 10 OR 01 = 11 binário (3 decimal), que é o resultado armazenado pelo Bus após o fim da instrução. Uma instrução de SLT é então carregada, comparando os valores de R1 (01) e R2 (01). Logicamente, a instrução retorna 0, fato observado pelo último valor do BusWire dessa simulação.



Simulação 9. Processador – Instruções MV

O DIN (registrador responsável por armazenar todas as instruções) já estava com muitas instruções, dificultando então as próximas simulações, poluindo visualmente e dificultando a análise das ondas. Limpamos então o DIN, e escrevemos novas instruções.

Fizemos inicialmente um MVI com o valor de 64 decimal, armazenando em R1. Posteriormente, realizamos um MV de R4 recebendo o valor de R1. Se olharmos atentamente para onda referente ao R4 após a finalização da instrução de MV, observamos que o mesmo contém 64 como valor armazenado, confirmando que a operação de MV se comportou como esperado.



Simulação 10. Processador – Instruções Shift Left

Limpamos o registrador DIN novamente, como explicado anteriormente. Fizemos dois MVI para registradores R1 e R2, com o valor de 1 decimal. Executamos então a operação de SL R2 << R1, ou seja, o conteúdo do R2 deverá mover em 1 bit de 0 a cada fim de execução da instrução. A execução da instrução de SL é executada N vezes, até o final da onda do Clock ser finalizada. Podemos observar então, que conteúdo de R2 se comporta exatamente como esperado, onde inicialmente era 01. Após uma execução, passa a ser 10. Na terceira execução, 100, e assim por diante.

Conclusão

O desenvolvimento deste trabalho foi muito satisfatório. Todas as etapas foram planejadas e implementadas como previsto sem maiores transtornos. As simulações e as explicações mostram em detalhes que o nosso projeto de processador apresenta um correto funcionamento e uma lógica consistente, apto a ser demonstrado na FPGA em laboratório.