The Pipeline of the TinyRISC Processor

Eliseu M. Chaves Filho

e-mail: eliseu@lam.ufrj.br

January 25, 1999

1 Introduction

This document describes the pipeline structure of the TinyRISC processor. There are two implementations of the TinyRISC processor: the full version and a prototype version, called Reduced TinyRISC (rTinyRISC). Both implementations have an identical pipeline, with four stages:

1. Fetch stage: this stage fetches instructions from the instruction memory;

2. Decode stage: this stage decodes the instructions and reads the operands stored in registers. It contains the data register file and the branch unit, which executes control transfer instructions;

3. Execute stage: this stage executes the instructions. It also accesses the data memory via the load/store instructions;

4. Write-back stage: this stage requests the storage of data into the register file.

2 The Internal Interfaces

Figure 1 (at the end of this document) shows the signal interfaces between the pipeline stages. These signals are described in the next subsections. Signal names have a one-letter prefix which indicates the signal source. The prefixes are: f, fetch; d, decode; x, execute; w, write-back stage. All outputs signals from a pipeline stage are clocked through the stage's output pipeline register, except for those with a nc suffix. These latter signals bypass the pipeline register, or go to another stage through a latch.

Signal differences between full TinyRISC and rTinyRISC are emphasized in this document, when appropriate.

2.1 The Fetch/Decode Interface

The signals in the fetch/decode interface are listed below (refer to Figure 1).

- \texttt{f\_NS\_PC} (32 bits): the address of the fetched instruction.

- \texttt{f\_NS\_OP} (7 bits): the \texttt{opcode} field of the fetched instruction;
• \texttt{INS\_IMM}: the \textit{imm} field of the fetched instruction;

• \texttt{INS\_SR1} and \texttt{INS\_SR1\_NC} (4 bits): the \textit{sr1} field of the fetched instruction;

• \texttt{INS\_SR2} and \texttt{INS\_SR2\_NC} (4 bits): the \textit{sr2} field of the fetched instruction;

• \texttt{INS\_DR} (4 bits): the \textit{dr} field of the fetched instruction;

• \texttt{INS\_MM} (16 bits): the \textit{imm} field of the fetched instruction;

• \texttt{READY}: it indicates to the decode stage that a new instruction has been fetched;

• \texttt{CHG\_PC}: it indicates to the fetch stage that the program counter should be changed to the target address of a control transfer instruction;

• \texttt{TARGET\_PC} (32 bits): the target address of a control transfer instruction;

• \texttt{STALL}: this signal indicates (to the internal clock generator) that the whole pipeline should be stalled;

The \texttt{STALL} signal (see Figure 1) comes from the execute stage. When it is asserted, the fetch stage finishes the current access and does not initiate a new access while the signal remains asserted.

2.2 The Decode/Execute Interface

The signals in the decode/execute interface are listed below (refer to Figure 1).

• \texttt{INS} (32 bits): the instruction;

• \texttt{INS\_SRC1} (32 bits): the first instruction operand;

• \texttt{INS\_SRC2} (32 bits): the second instruction operand;

• \texttt{INS\_DR} (4 bits): the instruction;

• \texttt{ALU\_OP}: it indicates to the execute stage that the decoded instruction is an arithmetic/logical operation;

• \texttt{SHFT\_OP}: it indicates to the execute stage that the decoded instruction is a shift operation;

• \texttt{CMP\_OP}: it indicates to the execute stage that the decoded instruction is a compare operation;

• \texttt{LDLI\_OP}: it indicates to the execute stage that the decoded instruction is a \texttt{LDLI};

• \texttt{LDUI\_OP}: it indicates to the execute stage that the decoded instruction is a \texttt{LDUI};

• \texttt{MEM\_OP}: it indicates that the decoded instruction is a memory operation;

• \texttt{MEM\_OP\_NC}: it indicates that the decoded instruction is a memory operation, before the beginning of the next clock cycle;
- **D_MEMW_OP**: it indicates that the decoded instruction is a memory write operation;
- **D_MEMW_OP_NC**: it indicates that the decoded instruction is a memory write operation, before the beginning of the next clock cycle;
- **D_MADDR_NC**: memory location address for the memory unit, available before the next clock cycle;
- **D_MDATA_NC**: memory data for the memory unit, available before the next clock cycle;
- **D_JAL_OP**: it indicates that the decoded instruction is a JAL;
- **D_JAL_ADDR (32 bits)**: the return address from a JAL instruction (JAL instruction’s address incremented by 2);
- **D_MORPH_OP**: it indicates that the decoded instruction is a MorphoSys operation;
- **D_MORPH_OP_NC**: it indicates that the decoded instruction is a MorphoSys operation, before the beginning of the next clock cycle;
- **D_MORPH_DMA_OP**: it indicates that the decoded instruction is a MorphoSys DMA-control operation;
- **D_RCRISC_OP**: it indicates that the decoded instruction is a RCRISC MorphoSys operation;
- **D_DR_WREQ**: it indicates that the decoded instruction writes into the data register file;
- **D_READY**: it indicates to the execute stage that a new instruction has been decoded;

The following signals constitute a data forwarding path from the execute stage to the the decode stage:

- **X_DATA_NC (32 bits)**: the data from the execute stage;
- **X_DR_NC (4 bits)**: the destination register of the instruction currently in the execute stage;
- **X_DR_WREQ_NC**: indicates whether the instruction currently in the execute stage writes into a data register.

The **F_STALL** indicates that the fetch stage has stalled. When it is asserted, the execute stage finishes any external access (to the data memory or to a MorphoSys component) and does not initiate a new access while the signal remains asserted.

The **X_CACHE_EN** signal is directed to the data memory interface, see Subsection 3.2.

The **X_MORPH_EN** and **X_MORPH_STALL** signals are directed to the MorphoSys decoder, see Subsection 3.3.
2.3 The Execute/Write-back Interface

The signals in the Execute/Write-back internal interface are listed below (see Figure 1).

- **x\_data** (32 bits): the data to be written into the specified register;
- **x\_dr** (4 bits): the identifier of the register to be written;
- **x\_dr\_wreq**: it indicates that the data from the execute stage should be written into the data register file;
- **x\_ready**: it indicates to the write-back stage that the execute stage has processed a new instruction.

2.4 The Execute/Cache Core Interface

The following signals constitute the interface between the execute stage and the data cache core:

- **x\_din** (32 bits): data input from the data cache core;
- **x\_tagin** (25 bits): tag input from the cache core;
- **x\_hit**: it indicates a hit in the data cache;
- **x\_dirtin**: it indicates that the selected cache line is dirty;
- **x\_pdout** (32 bits): data to the cache core, produced by TinyRISC;
- **x\_mdout** (32 bits): data to the cache core, originated from main memory;
- **x\_rd**: it indicates a read access to the cache core;
- **x\_pwr**: it indicates a TinyRISC-data write access to the cache core;
- **x\_mwr**: it indicates a main memory-data write access to the cache core;
- **x\_setval**: it indicates to the cache core that the selected line should be marked as valid;
- **x\_setdirt**: it indicates to the cache core that the selected line should be marked as dirty.

2.5 The Write-back/Decode Interface

The signals in the Write-back/Decode interface are listed below (see Figure 1).

- **w\_data** (32 bits): the data to be written into the specified register.
- **w\_dr** (4 bits): the number of the destination register to be written;
- **w\_dr\_wren**: it indicates that data should be written into the data register file in the current cycle;
2.6 The Clock Signals

The internal clock generator (see Figure 1) produces two clock signals to the pipeline stages. The signals to/from the clock generator are:

- **p stall**: indicates that the fetch state and/or the execute stage has stalled and therefore the whole pipeline should be stalled;
- **CLOCK**: normal clock signal;
- **sCLOCK**: stall clock signal. This is the clock signal to the pipeline registers and the program counter registers. It remains inhibited while p stall is active.

3 The External Interfaces

There are three external interfaces: (1) between the fetch stage and the instruction memory; (2) between the execute stage and the data memory; (3) between the decode stage and the external MorphoSys instruction decoder.

3.1 The Instruction Memory Interface

The signals in the interface between the fetch stage and the instruction cache memory are listed below (refer to Figure 1).

- **IA**: the 32-bit instruction address bus. _In rTinyRISC, the size of the instruction address bus is 10 bits;_
- **IR**: it enables an instruction fetch;
- **IB**: the 32-bit instruction bus;
- **INSACK**: when asserted, it indicates that the instruction will be available in the current cycle.

The IR signal remains continuously asserted during instruction fetch cycles. The instruction cache should keep the INSACK signal asserted during accesses with a hit. In the case of a miss, the instruction cache should deassert the INSACK signal within the same cycle in which the missing address was generated. The IR signal remains asserted during a miss.

The IR signal is deasserted at the end of a cycle in which occurs a stall due to a data cache miss or DMA-control instruction. The instruction cache should deassert the INSACK signal in response to IR being deasserted. A new instruction fetch is not initiated while there is a stall due to a data miss or DMA-control instruction.

A behavioral model of the instruction memory can be found in file tinytest.vhd.
3.2 The Data Memory Interface

The signals in the external interface between the execute stage and the data cache memory are (see Figure 1):

- **DA**: the data 32-bit address bus. *In rTinyRISC, the size of the data address bus is 10 bits;*
- **DR**: it enables a data access;
- **DB**: the 32-bit data bus. *In rTinyRISC, the size of the external data bus is 16 bits. The 16 most significant bits of loaded data are internally zeroed;*
- **DACK**: it indicates that the data will be provided in the current cycle;
- **WR**: when asserted, it indicates a data write operation. When negated, it indicates a data read operation.

The cache_en signal enables or disables the internal data cache. When the data cache is enabled, the above signals are active during a cache line load or cache line write back operation. The data cache is disabled, the signals behave as described next.

The **DR** signal is asserted on the beginning of a data cache access cycle and deasserted at the end of the cycle. It remains continuously asserted during a sequence of contiguous load/store instructions. The **DR** signal remains asserted during a data miss. A new data access is not initiated (**DR** is not asserted) while there is a stall due to an instruction miss or DMA-control instruction.

The data cache should assert the **DACK** signal in accesses with a hit and it should negate **DACK** after **DR** being deasserted. The data cache should keep **DACK** negated while there is a data miss. In the case of a hit followed by a contiguous miss, the data cache should deassert the **DACK** signal within the same cycle in which the missing address was generated.

The **WR** signal is activated on data write accesses. It is asserted on the falling edge of the clock, when the data address is already stable when writing is enabled. The data becomes available on the data bus only when the **WR** signal is asserted. The **WR** signal is deasserted at the end of the write access cycle. It remains asserted during a data write miss.

A behavioral model of the data memory can be found in file tinytest.vhd.

3.3 The MorphoSys Decoder Interface

The signals to/from the decode stage to the external MorphoSys instruction decoder are:

- **MORPHO_INS** (32 bits): the full instruction from TinyRISC;
- **MORPHO_SRC1** (32 bits): the first instruction operand;
- **MORPHO_SRC2** (32 bits): the second instruction operand;
- **MORPHO_EN**: this signal enables the operation of the MorphoSys decoder;
- **MORPHOSTALL**: this signal stalls the TinyRISC processor.
The buses \texttt{morpho\_ins}, \texttt{morpho\_src1} and \texttt{morpho\_src2} do not exist in rTinyRISC.

The Morphosys decoder should assert the \texttt{morpho\_ stall} signal when it receives a DMA-control instruction and the DMA controller is busy. The TinyRISC processor remains stalled while this signal is asserted.
Figure 1: The pipeline interfaces.