Alterações no OrCS

trace_reader_t::trace_string_to_opcode(...)

trace_reader_t::trace_fetch(...)

Before
bool mem_is_read;
for (uint32_t r=0; r < m->num_reads; ++r) {

for (uint32_t w=0; w < m->num_writes; ++w) {

bool mem_is_read; uint32_t num_reads = (m->num_reads != UINT32_MAX) ? m->num_reads : trace_next_num_accesses(); for (uint32_t r=0; r < num_reads; ++r) {

uint32_t num_writes = (m->num_writes != UINT32_MAX) ? m->num_writes : trace_next_num_accesses(); for (uint32_t w=0; w < num_writes; ++w) {

trace_reader_t::trace_next_num_accesses(...)

Alteração em relação ao **trace_next_memory** if (file_line[0] == '\0' || file_line[0] == "#") { else {
 ERROR_ASSERT_PRINTF(file_line[0] != '\$', "Error searching for num accesses in line
%s\n", file_line); uint32_t num_accesses = (uint32_t)strtoul(file_line + 1, NULL, 10);
assert(num_accesses <= MAX_MEM_OPERATIONS);</pre> return num_accesses;

Instruções Id/st RVV apresentam um número variado de leituras e escritas durante a execução. O atual armazenamento desse número no traço estático não dá suporte a essas variações.

Solução:Instruções com esse número variado apresentarão -1 ao invés do número fixo no traço estático. O traço dinâmico vai conter uma anotação específica indicando o número de acessos dessas instruções. Apenas as instruções com -1 terão essa anotação para manter a retrocompatibilidade.

O formato da anotação é \$<numero>

Based on trace_next_memory

Problema a ser solucionado:

Em seguida os <numero> acessos desse tipo.

Antes do switch(opcode_hash) op_info.variable_num_reads = false;
op_info.variable_num_writes = false; # Para as instruções com número variável de leituras (RVV) case ...:

case ...:

op_info.variable_num_reads = true; break; # Para as instruções com número variável de escritas (RVV)

orcs_tracer_t::memory_trace(...)

op_info = fill_instruction_info(insn, pc, opcode_hash); // Add a marker into the memory trace if (op_info.variable_num_reads) { memory_access_t var_reads; var_reads.is_num_accesses = true; var_reads.num_accesses = op_info.num_reads; this->BBL_mem_accesses.push_back(var_reads);

op_info.variable_num_writes = true;

Adicionar entre

break;

for (uint32_t i=0; i < op_info.num_reads; ++i) { new_op.is_num_accesses = false;

Adicionar entre

for (uint32_t i=0; i < op_info.num_writes; ++i) {

// Add a marker into the memory trace
if (op_info.variable_num_writes) {
 memory_access_t var_writes; var_writes.is_num_accesses = true;
var_writes.num_accesses = op_info.num_writes; this->BBL_mem_accesses.push_back(var_writes);

new_op.is_num_accesses = false;

gzwrite(...

orcs_tracer_t::print_orcs_memory_trace(...)

for (size_t i=0; i < BBL_mem_accesses.size(); ++i) {
 memory_access_t new_op = BBL_mem_accesses[i];</pre> if (new_op.is_num_accesses == false) {
 snprintf(...
} else {
 snprintf(orcs_print_aux, 1024, "\$%u\n", new_op.num_accesses);

Alterações no SPIKE

orcs_tracer_t::fill_instruction_info(...)

opcode_package_t # Adição bool variable_num_reads; bool variable_num_writes;

memory_access_t

Adição bool is_num_accesses; uint32_t num_accesses;