Department of Computer Science
Course number: CSC.T341

コンピュータ論理設計
Computer Logic Design

9. シングルサイクルプロセッサの設計と実装
Design and Implementation of a Single Cycle Processor

吉瀬 謙二　情報工学系
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp　www.arch.cs.titech.ac.jp/lecture/CLD/
講義：月曜日 10:45-12:25, 木曜日 10:45-12:25

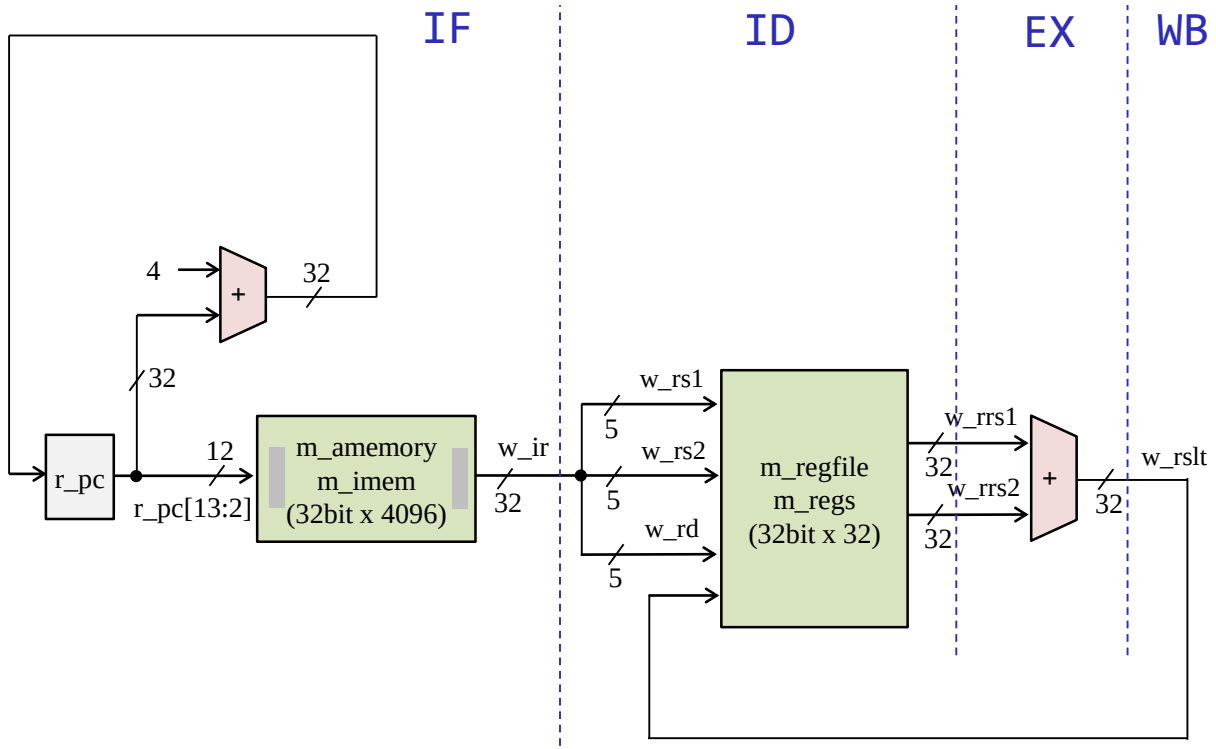
Ver. 2023-05-11a

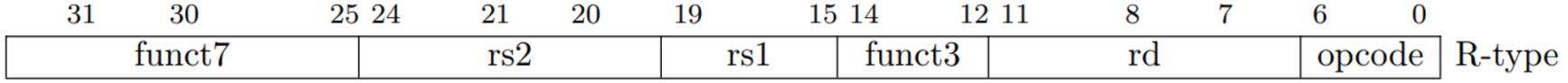
プロセッサが命令を処理するための基本的な5つのステップ

- IF (Instruction Fetch)
  メモリから命令をフェッチする.
- ID (Instruction Decode)
  命令をデコード(解読)しながら, レジスタの値を読み出す(Operand Fetch)
- EX (Execution)
  命令操作の実行またはアドレスの生成を行う.
- MEM (Memory Access)
  必要であれば, メモリ(データ・メモリ)のオペランドにアクセスする.
- WB (Write Back)
  必要であれば, 結果をレジスタに書き込む.

m_proc02 addを処理するシングルサイクルのプロセッサ

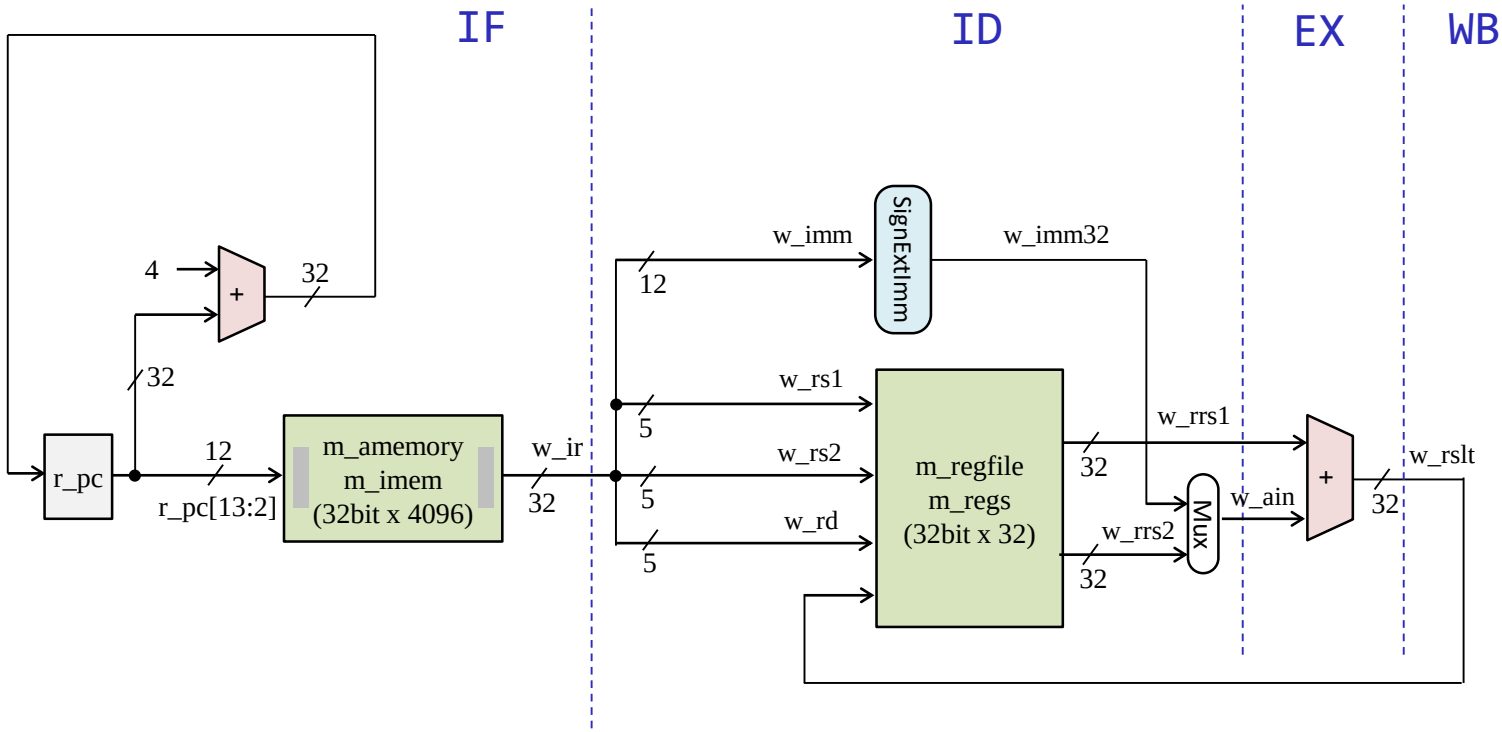
- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), ライトバック(WB) の処理をおこなう加算命令 (add) のみに対応したプロセッサのブロック図

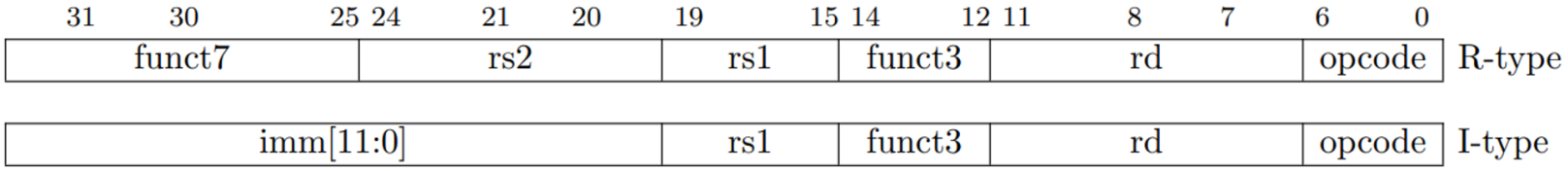




m_proc03 add と addi を処理するプロセッサ

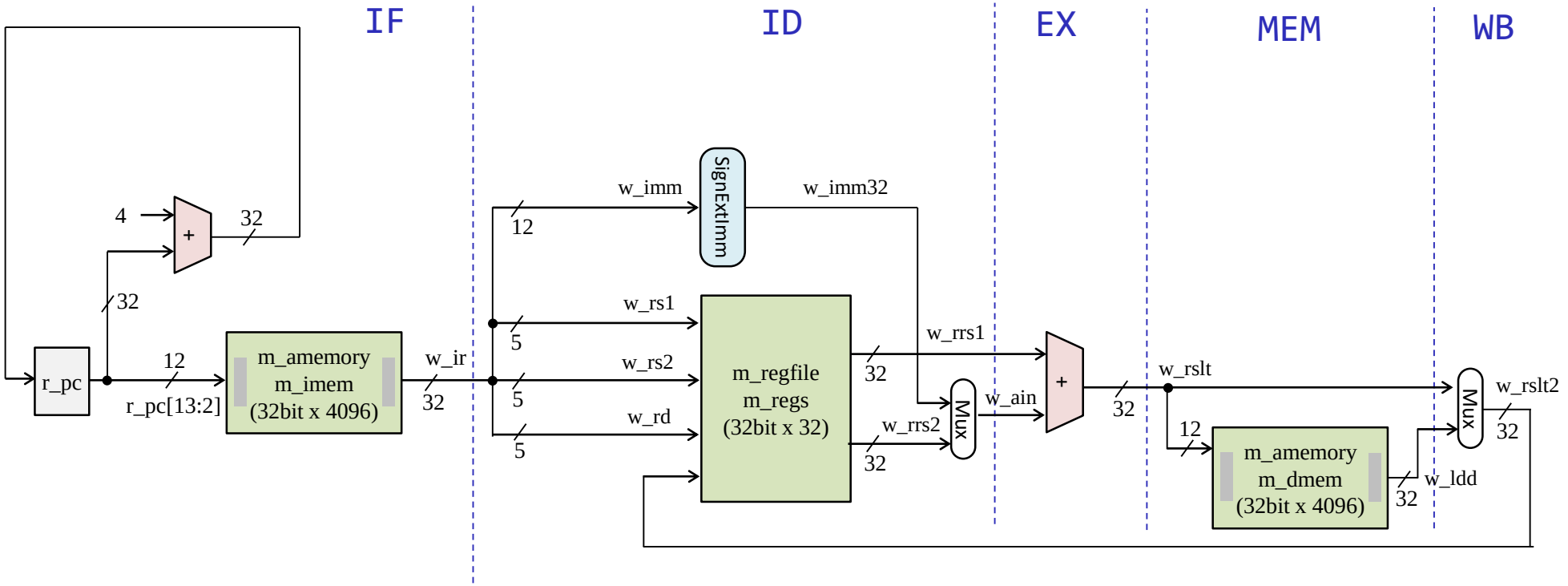
- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), ライトバック(WB) の処理をおこなう加算命令(add, addi)に対応したプロセッサのブロック図

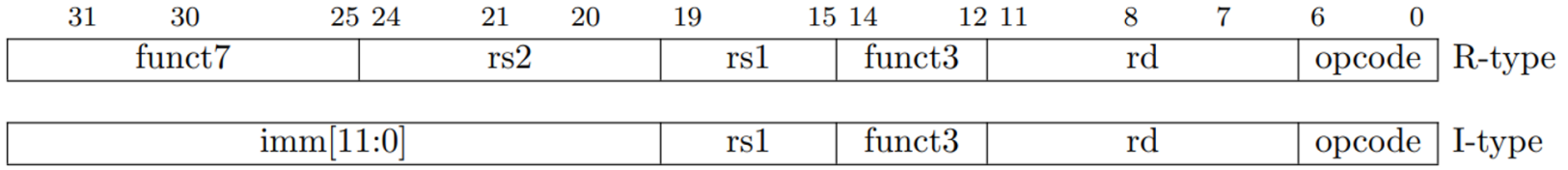




m_proc04 add, addi, lw を処理するプロセッサ

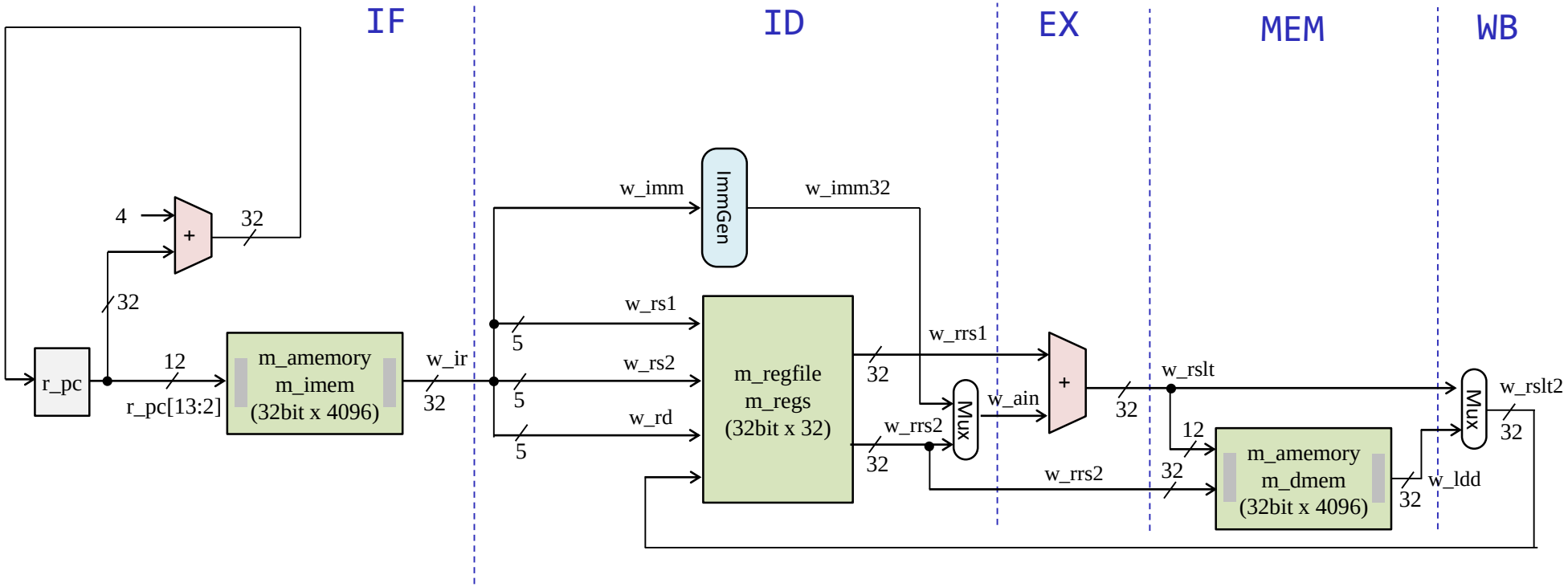
- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), メモリアクセス(MEM), ライトバック(WB) の処理をおこなうadd, addi, lw命令に対応したプロセッサ

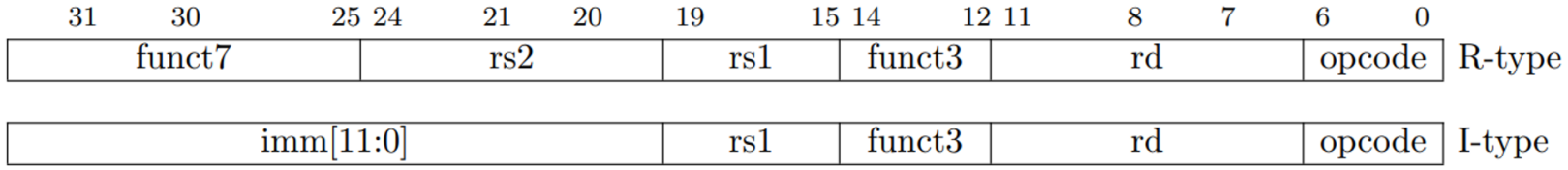




m_proc05 add, addi, lw, sw を処理するプロセッサ

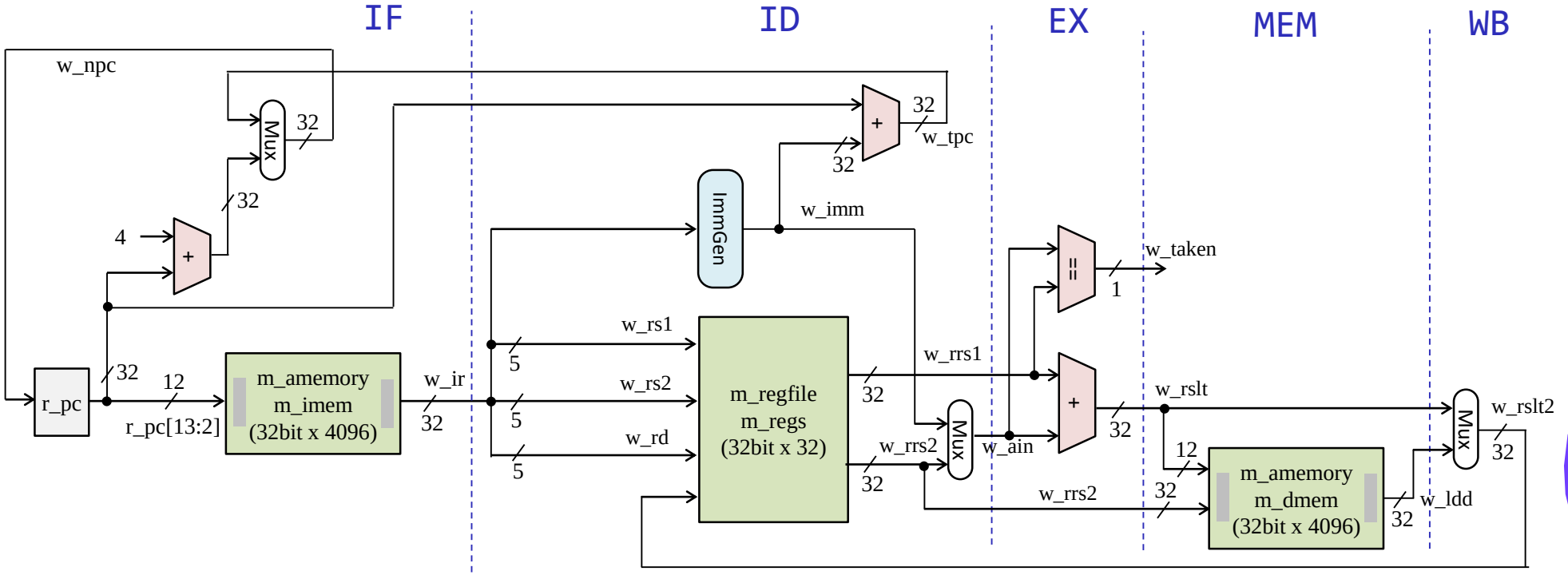
- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), メモリアクセス(MEM), ライトバック(WB) の処理をおこなうadd, addi, lw, sw命令に対応したプロセッサ

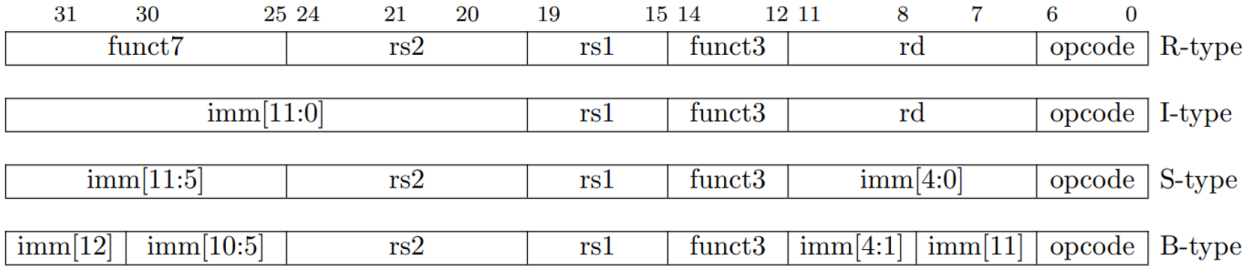




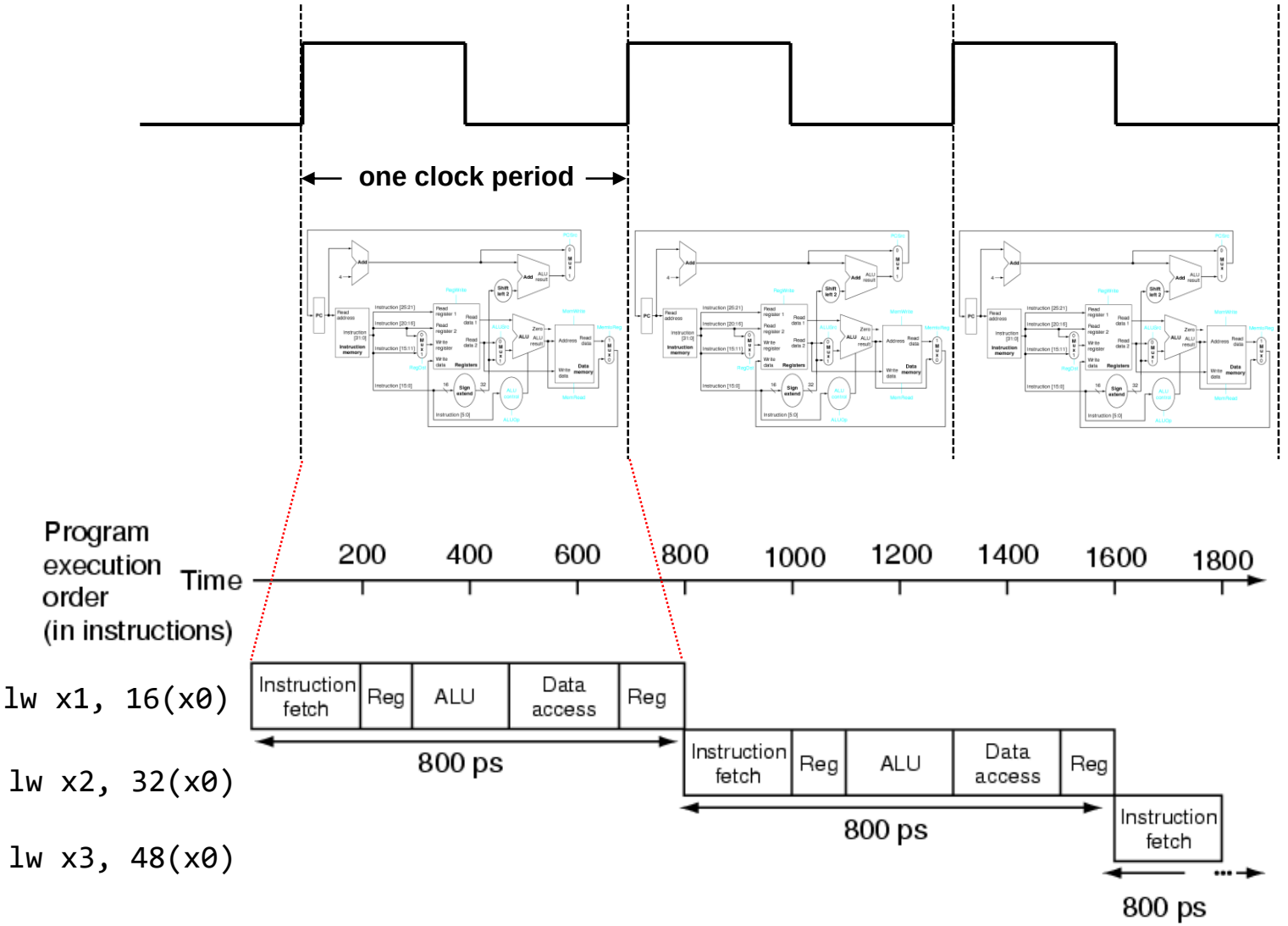
m_proc06 add, addi, lw, sw, beq を処理するプロセッサ

- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), メモリアクセス(MEM), ライトバック(WB) の処理をおこなうadd, addi, lw, sw, beq命令に対応したプロセッサ



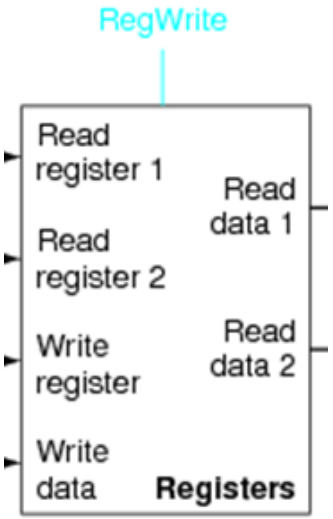


Single Cycle Processor (our baseline processor)



Register file, レジスタファイル m_regfile の実装

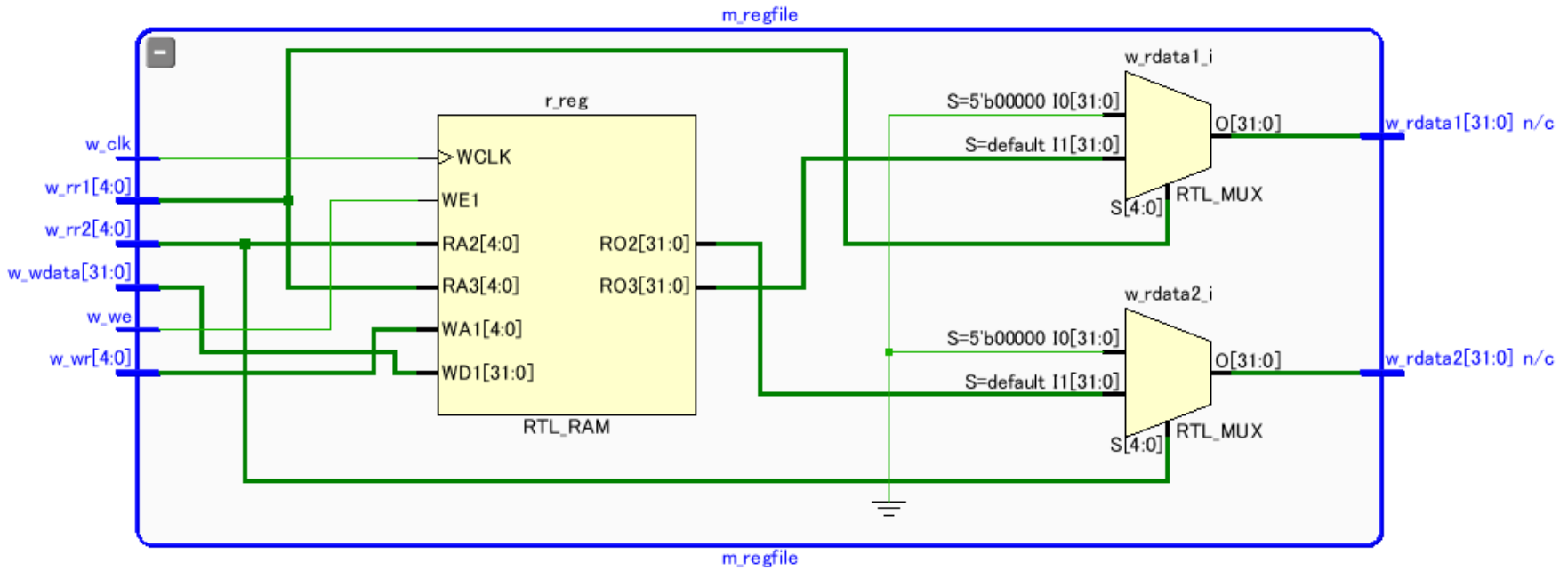
- Verilog HDLでは，ビット幅Bでワード数Wのメモリ m を reg [B-1:0] m [0:W-1] として宣言できる．
- w_rr1 で指定したレジスタの値を読み出し w_rdata1 に出力する．非同期の読み出し．
- w_rr2 で指定したレジスタの値を読み出し w_rdata2 に出力する．非同期の読み出し．
  - ただし，x0 (zero) の読み出しは，値0を出力する．
- posedge w_clk のタイミングで，w_we (write enable) が1の時に，w_wr (write register) で指定されたレジスタに w_wdata (write data) の値を書き込む．
- このモジュールではadd命令の動作確認のために x1 を 1 で，x2 を 2 で初期化している．



code112.v

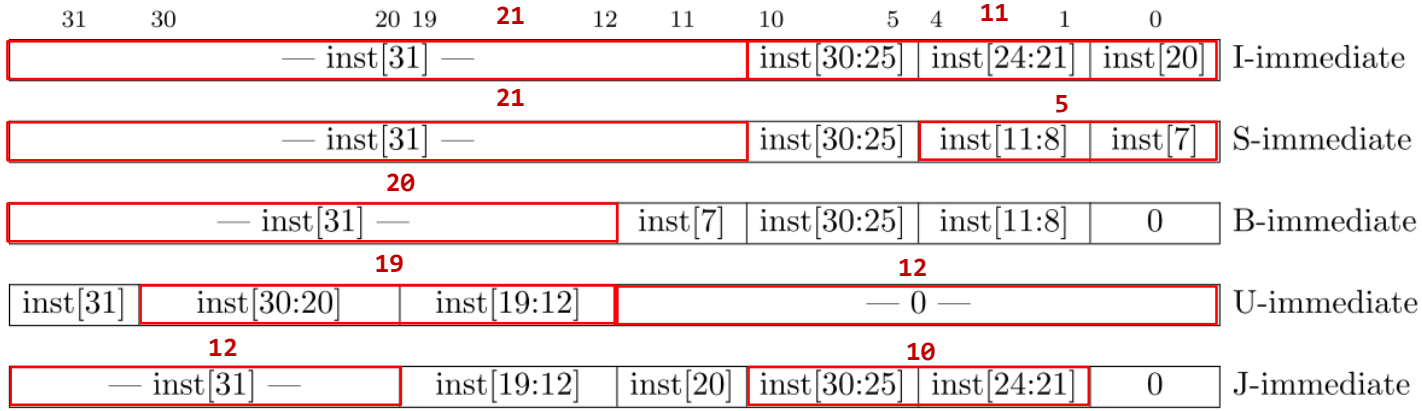
```
module m_regfile (w_clk, w_rr1, w_rr2, w_wr, w_we, w_wdata, w_rdata1, w_rdata2);
  input  wire        w_clk;
  input  wire [4:0]  w_rr1, w_rr2, w_wr;
  input  wire [31:0] w_wdata;
  input  wire        w_we;
  output wire [31:0] w_rdata1, w_rdata2;

  reg [31:0] r[0:31];
  assign w_rdata1 = (w_rr1==0) ? 0 : r[w_rr1];
  assign w_rdata2 = (w_rr2==0) ? 0 : r[w_rr2];
  always @(posedge w_clk) if(w_we) r[w_wr] <= w_wdata;

  initial r[1] = 1;
  initial r[2] = 2;
endmodule
```

Register file, module m_regfile



m_immgen RISC-Vの即値を出力するモジュール

- 32ビットの命令を入力 w_i として，RISC-Vの即値 r_imm を出力するモジュール
- reg を使っているが，これは組合せ回路となる．



code160.v

```
module m_immgen(w_i, r_imm); // module immediate generator
  input  wire [31:0] w_i;    // instruction
  output reg  [31:0] r_imm;  // r_immediate

  always @(*) case (w_i[6:2])
    5'b11000: r_imm <= {{20{w_i[31]}}, w_i[7], w_i[30:25], w_i[11:8], 1'b0};   // B-type
    5'b01000: r_imm <= {{21{w_i[31]}}, w_i[30:25], w_i[11:7]};                 // S-type
    5'b11011: r_imm <= {{12{w_i[31]}}, w_i[19:12], w_i[20], w_i[30:21], 1'b0}; // J-type
    5'b01101: r_imm <= {w_i[31:12], 12'b0};                                    // U-type
    5'b00101: r_imm <= {w_i[31:12], 12'b0};                                    // U-type
    default : r_imm <= {{21{w_i[31]}}, w_i[30:20]};                  // I-type & R-type
  endcase
endmodule
```

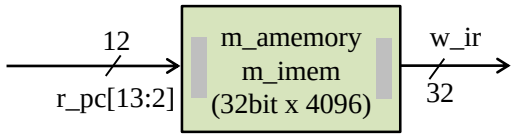
m_amemory 非同期式メモリの記述とシミュレーション

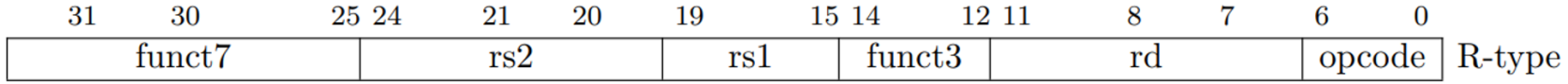
- Verilog HDLでは，ビット幅Bでワード数Wのメモリ m を reg [B-1:0] m [0:W-1] として宣言できる.
- 読み出す動作でクロック信号を利用しないメモリを非同期メモリ (asynchronous memory) と呼ぶ.
- 非同期式メモリの記述例を示す．シミュレーションでの読み出しの遅延を 20nsec とした．w_addr で指定されたアドレスの内容を読み出す．posedge w_clk のタイミングで，w_we (write enable) が1の時に，w_addr で指定されたアドレスに w_din (data in) の値を書き込む.
- このコードをシミュレーションして，波形を確認すること.

```
module m_amemory (w_clk, w_addr, w_we, w_din, w_dout);
  input  wire w_clk, w_we;
  input  wire [11:0] w_addr;
  input  wire [31:0] w_din;
  output wire [31:0] w_dout;

  reg [31:0] cm_ram [0:4095]; // 4K word (4096 x 32bit) memory
  always @(posedge w_clk) if (w_we) cm_ram[w_addr] <= w_din;
  assign #20 w_dout = cm_ram[w_addr];

  initial begin
    cm_ram[0]={7'd0, 5'd0, 5'd0, 3'd0, 5'd0, 7'b0110011}; // add x0, x0, x0
    cm_ram[1]={7'd0, 5'd1, 5'd0, 3'd0, 5'd4, 7'b0110011}; // add x4, x0, x1
    cm_ram[2]={7'd0, 5'd2, 5'd1, 3'd0, 5'd5, 7'b0110011}; // add x5, x1, x2
    cm_ram[3]={7'd0, 5'd5, 5'd4, 3'd0, 5'd6, 7'b0110011}; // add x6, x4, x5
    cm_ram[4]={7'd0, 5'd0, 5'd0, 3'd0, 5'd0, 7'b0110011}; // add x0, x0, x0
  end
endmodule
```

```
module m_top ();
  reg r_clk=0; initial forever #50 r_clk = ~r_clk;
  reg [31:0] r_pc = 0;
  always @(posedge r_clk) r_pc <= #3 r_pc + 4;

  wire [31:0] w_data;
  m_amemory m (r_clk, r_pc[13:2], 1'd0, 32'd0, w_data);

  initial $dumpfile("main.vcd"); /* file name for GTKWave */
  initial $dumpvars(0, m_top);   /* module    for GTKWave */
  initial #1000 $finish;
  always@(*) #80 $write("%3d %d %x¥n", $time, r_pc, w_data);
endmodule
```

code111.v





m_amemory の修正 include “program.txt”

- 命令メモリとデータメモリとして用いる m_amemory の内容を program.txt で初期化.
- x30 への書き込みをレジスタに保存して, プロセッサの出力 w_led とする.
- このプログラムを実行して beq に到達した時点の w_led の値は 55 + 9 = 64 になる.

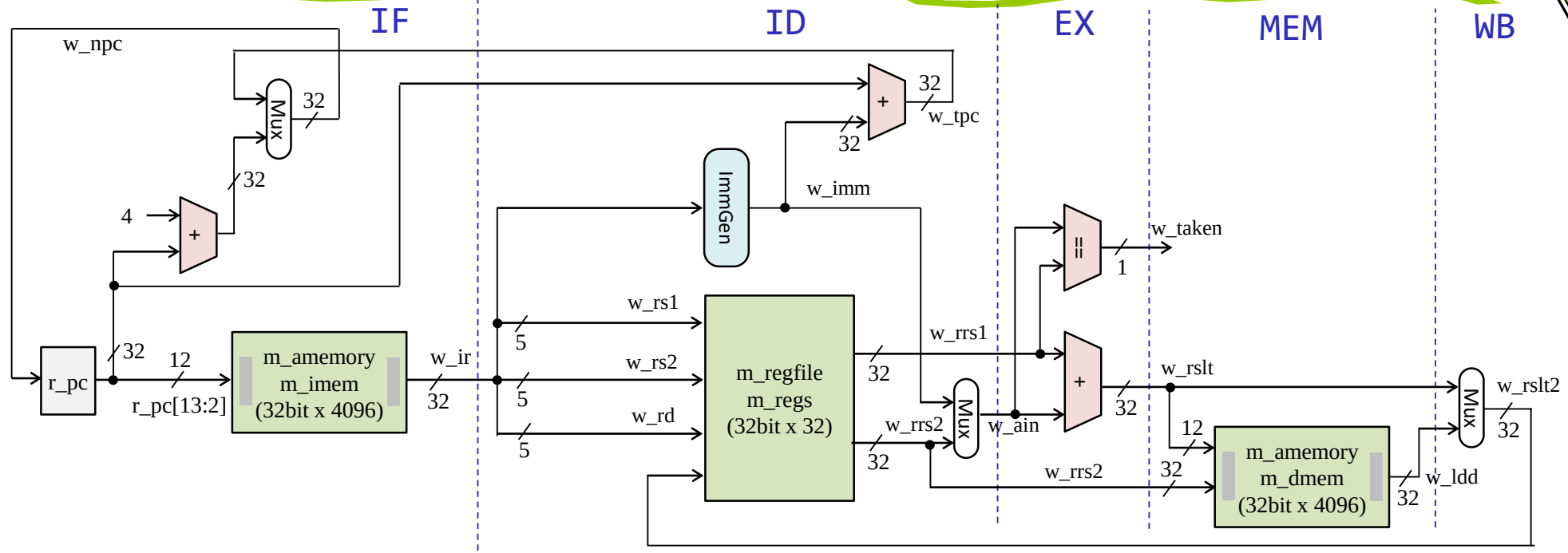
code160.v

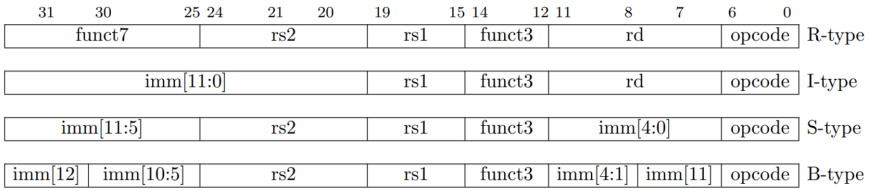
```
module m_amemory (w_clk, w_addr, w_we, w_din, w_dout);
  input  wire w_clk, w_we;
  input  wire [11:0] w_addr;
  input  wire [31:0] w_din;
  output wire [31:0] w_dout;
  reg [31:0]          cm_ram [0:4095]; // 4K word (4096 x 32bit) memory
  always @(posedge w_clk) if (w_we) cm_ram[w_addr] <= w_din;
  assign #20 w_dout = cm_ram[w_addr];
`include "program.txt"
endmodule
```

program.txt

```
initial begin
  cm_ram[0]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b0110011}; //    add  x0, x0, x0  // NOP
  cm_ram[1]={12'd55,     5'd0, 3'b000, 5'd4, 7'b0010011}; //    addi x4, x0, 55  // x4 = 55
  cm_ram[2]={7'd0, 5'd4, 5'd0, 3'b010, 5'd16,7'b0100011}; //    sw   x4, 16(x0)  // m[16] = x4
  cm_ram[3]={12'd16,     5'd0, 3'b010, 5'd7, 7'b0000011}; //    lw   x7, 16(x0)  // x7 = m[16]
  cm_ram[4]={12’d9,      5'd0, 3'b000, 5'd2, 7'b0010011}; //    addi x2, x0, 9   // x2 = 9
  cm_ram[5]={7'd0, 5'd2, 5'd7, 3'b000, 5'd3, 7'b0110011}; //    add  x3, x7, x2  // x3 = x7 + x2
  cm_ram[6]={7'd0, 5'd3, 5'd0, 3'b000, 5'd30,7'b0110011}; //    add  x30,x0, x3  // led = x3
  cm_ram[7]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b1100011}; // L: beq  x0, x0, L
end
```

m_proc06 add, addi, lw, sw, beq を処理するプロセッサ

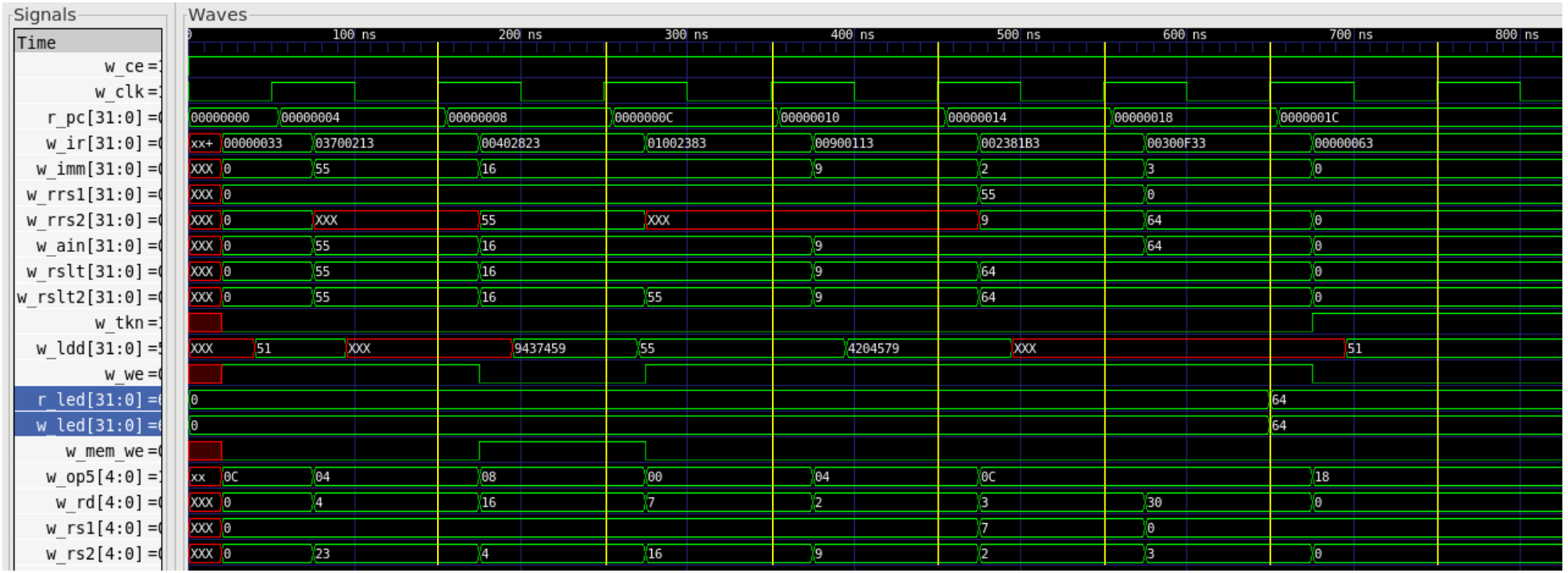


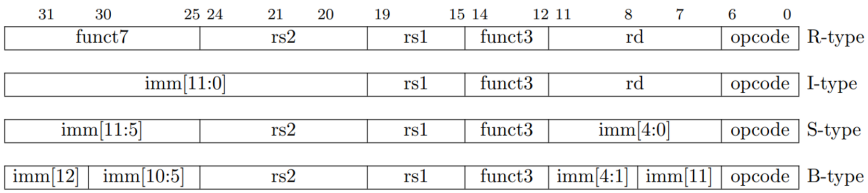


クロック毎の動作を確認する.

```
cm_ram[0]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b0110011}; //    add  x0, x0, x0  // NOP
cm_ram[1]={12'd55,     5'd0, 3'b000, 5'd4, 7'b0010011}; //    addi x4, x0, 55  // x4 = 55
cm_ram[2]={7'd0, 5'd4, 5'd0, 3'b010, 5'd16,7'b0100011}; //    sw   x4, 16(x0)  // m[16] = x4
cm_ram[3]={12'd16,     5'd0, 3'b010, 5'd7, 7'b0000011}; //    lw   x7, 16(x0)  // x7 = m[16]
cm_ram[4]={12'd9,      5'd0, 3'b000, 5'd2, 7'b0010011}; //    addi x2, x0, 9   // x2 = 9
cm_ram[5]={7'd0, 5'd2, 5'd7, 3'b000, 5'd3, 7'b0110011}; //    add  x3, x7, x2  // x3 = x7 + x2
cm_ram[6]={7'd0, 5'd3, 5'd0, 3'b000, 5'd30,7'b0110011}; //    add  x30,x0, x3  // led = x3
cm_ram[7]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b1100011}; // L: beq  x0, x0, L
```

m_proc06 add, addi, lw, sw, beq を処理するプロセッサ






```
cm_ram[0]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b0110011}; //    add  x0, x0, x0  // NOP
cm_ram[1]={12'd55,     5'd0, 3'b000, 5'd4, 7'b0010011}; //    addi x4, x0, 55  // x4 = 55
cm_ram[2]={7'd0, 5'd4, 5'd0, 3'b010, 5'd16,7'b0100011}; //    sw   x4, 16(x0)  // m[16] = x4
cm_ram[3]={12'd16,     5'd0, 3'b010, 5'd7, 7'b0000011}; //    lw   x7, 16(x0)  // x7 = m[16]
cm_ram[4]={12'd9,      5'd0, 3'b000, 5'd2, 7'b0010011}; //    addi x2, x0, 9   // x2 = 9
cm_ram[5]={7'd0, 5'd2, 5'd7, 3'b000, 5'd3, 7'b0110011}; //    add  x3, x7, x2  // x3 = x7 + x2
cm_ram[6]={7'd0, 5'd3, 5'd0, 3'b000, 5'd30,7'b0110011}; //    add  x30,x0, x3  // led = x3
cm_ram[7]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b1100011}; // L: beq  x0, x0, L
```

m_proc06 add, addi, lw, sw, beq を処理するプロセッサ

m_top はシミュレーション用

```
/***** top module for simulation *****/
module m_top ();
   reg r_clk=0; initial forever #50 r_clk = ~r_clk;
   wire [31:0] w_led;

   initial $dumpfile("main.vcd");
   initial $dumpvars(0, m_top);

   m_proc06 p (r_clk, 1'b1, w_led);
   initial $write("time: r_pc     w_ir     w_rrs1  w_ain   r_rslt  r_led¥n");
   always@(posedge r_clk) $write("%4d: %x %x %07d %07d %07d %07d¥n", $time,
                          p.r_pc, p.w_ir, p.w_rrs1, p.w_ain, p.w_rslt, w_led);
   initial #1000 $finish;
endmodule

/***** main module for FPGA implementation *****/
/*
module m_main (w_clk, w_led);
   input  wire w_clk;
   output wire [3:0] w_led;

   wire w_clk2, w_locked;
   clk_wiz_0 clk_w0 (w_clk2, 0, w_locked, w_clk);

   wire [31:0] w_dout;
   m_proc06 p (w_clk2, w_locked, w_dout);

   vio_0 vio_00(w_clk2, w_dout);

   reg [3:0] r_led = 0;
   always @(posedge w_clk2)
     r_led <= {^w_dout[31:24], ^w_dout[23:16], ^w_dout[15:8], ^w_dout[7:0]};
   assign w_led = r_led;
endmodule
*/
```

code160.v

m_proc06 add, addi, lw, sw, beq を処理するプロセッサ

m_main はFPGA用

```
/***** top module for simulation *****/
// module m_top ();
//   reg r_clk=0; initial forever #50 r_clk = ~r_clk;
//   wire [31:0] w_led;
//
//   initial $dumpfile("main.vcd");
//   initial $dumpvars(0, m_top);
//
//   m_proc06 p (r_clk, 1'b1, w_led);
//   initial $write("time: r_pc     w_ir     w_rrs1  w_ain   r_rslt  r_led¥n");
//   always@(posedge r_clk) $write("%4d: %x %x %07d %07d %07d %07d¥n", $time,
//                          p.r_pc, p.w_ir, p.w_rrs1, p.w_ain, p.w_rslt, w_led);
//   initial #1000 $finish;
// endmodule

/***** main module for FPGA implementation *****/

module m_main (w_clk, w_led);
   input  wire w_clk;
   output wire [3:0] w_led;

   wire w_clk2, w_locked;
   clk_wiz_0 clk_w0 (w_clk2, 0, w_locked, w_clk);

   wire [31:0] w_dout;
   m_proc06 p (w_clk2, w_locked, w_dout);

   vio_0 vio_00(w_clk2, w_dout);

   reg [3:0] r_led = 0;
   always @(posedge w_clk2)
     r_led <= {^w_dout[31:24], ^w_dout[23:16], ^w_dout[15:8], ^w_dout[7:0]};
   assign w_led = r_led;
endmodule
```

code160.v

m_proc06 add, addi, lw, sw, beq を処理するプロセッサ

青色の部分を記述する.

code160.v

```
module m_proc06 (w_clk, w_ce, w_led);
  input  wire w_clk, w_ce;
  output wire [31:0] w_led;

  reg [31:0] r_pc = 0;
  wire [31:0] w_ir;
  wire [4:0]  w_op5 = w_ir[6:2];
  wire [4:0]  w_rs1 = w_ir[19:15];
  wire [4:0]  w_rs2 = w_ir[24:20];
  wire [4:0]  w_rd  = w_ir[11:7];
  wire w_we = w_ce & (w_op5==5'b01100 || w_op5==5'b00100 || w_op5==5'b00000);

  wire [31:0] w_imm, w_rrs1, w_rrs2, w_ain, w_rslt, w_ldd, w_rslt2;

  /* Please describe here by yourself */


  reg [31:0] r_led = 0;
  always @(posedge w_clk) if(w_we & w_rd==30) r_led <= w_rslt;
  assign w_led = r_led;
endmodule
```

```
cm_ram[0]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b0110011}; //    add  x0, x0, x0  // NOP
cm_ram[1]={12'd55,     5'd0, 3'b000, 5'd4, 7'b0010011}; //    addi x4, x0, 55  // x4 = 55
cm_ram[2]={7'd0, 5'd4, 5'd0, 3'b010, 5'd16,7'b0100011}; //    sw   x4, 16(x0)  // m[16] = x4
cm_ram[3]={12'd16,     5'd0, 3'b010, 5'd7, 7'b0000011}; //    lw   x7, 16(x0)  // x7 = m[16]
cm_ram[4]={12'd9,      5'd0, 3'b000, 5'd2, 7'b0010011}; //    addi x2, x0, 9   // x2 = 9
cm_ram[5]={7'd0, 5'd2, 5'd7, 3'b000, 5'd3, 7'b0110011}; //    add  x3, x7, x2  // x3 = x7 + x2
cm_ram[6]={7'd0, 5'd3, 5'd0, 3'b000, 5'd30,7'b0110011}; //    add  x30,x0, x3  // led = x3
cm_ram[7]={7'd0, 5'd0, 5'd0, 3'b000, 5'd0, 7'b1100011}; // L: beq  x0, x0, L
```

ヒント（m_proc06 の実装）

青色の部分を少し記述した.

code160.v

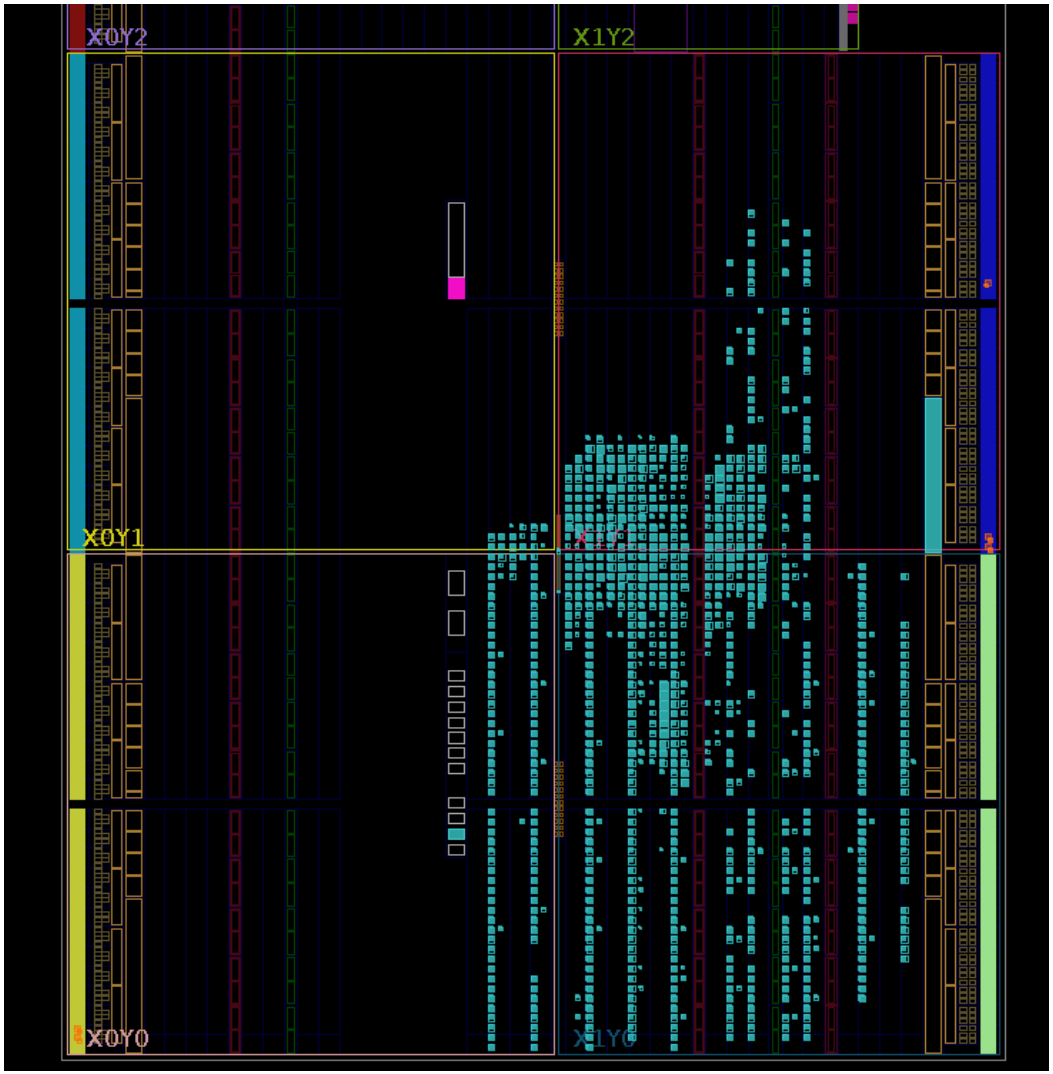
```
module m_proc06 (w_clk, w_ce, w_led);
  input  wire w_clk, w_ce;
  output wire [31:0] w_led;

  reg [31:0] r_pc = 0;
  wire [31:0] w_ir;
  wire [4:0]  w_op5 = w_ir[6:2];
  wire [4:0]  w_rs1 = w_ir[19:15];
  wire [4:0]  w_rs2 = w_ir[24:20];
  wire [4:0]  w_rd  = w_ir[11:7];
  wire w_we = w_ce & (w_op5==5'b01100 || w_op5==5'b00100 || w_op5==5'b00000);

  wire [31:0] w_imm, w_rrs1, w_rrs2, w_ain, w_rslt, w_ldd, w_rslt2;

  /* Please describe here by yourself */
  m_amemory m_imem (w_clk, r_pc[13:2], 1'd0, 32'd0, w_ir);
  m_immgen m_immgen0 (w_ir, w_imm);
  m_regfile m_regs (w_clk, w_rs1, w_rs2, w_rd, w_we, w_rslt2, w_rrs1, w_rrs2);
  assign w_ain = (w_op5==5'b01100) ? w_rrs2 : w_imm;
  assign w_rslt = w_rrs1 + w_ain;

  // about data memory here

  // about r_pc update here



  reg [31:0] r_led = 0;
  always @(posedge w_clk) if(w_we & w_rd==30) r_led <= w_rslt;
  assign w_led = r_led;
endmodule
```

m_proc06 の論理合成，配置配線の結果

Utilization Post-Synthesis | **Post-Implementation**

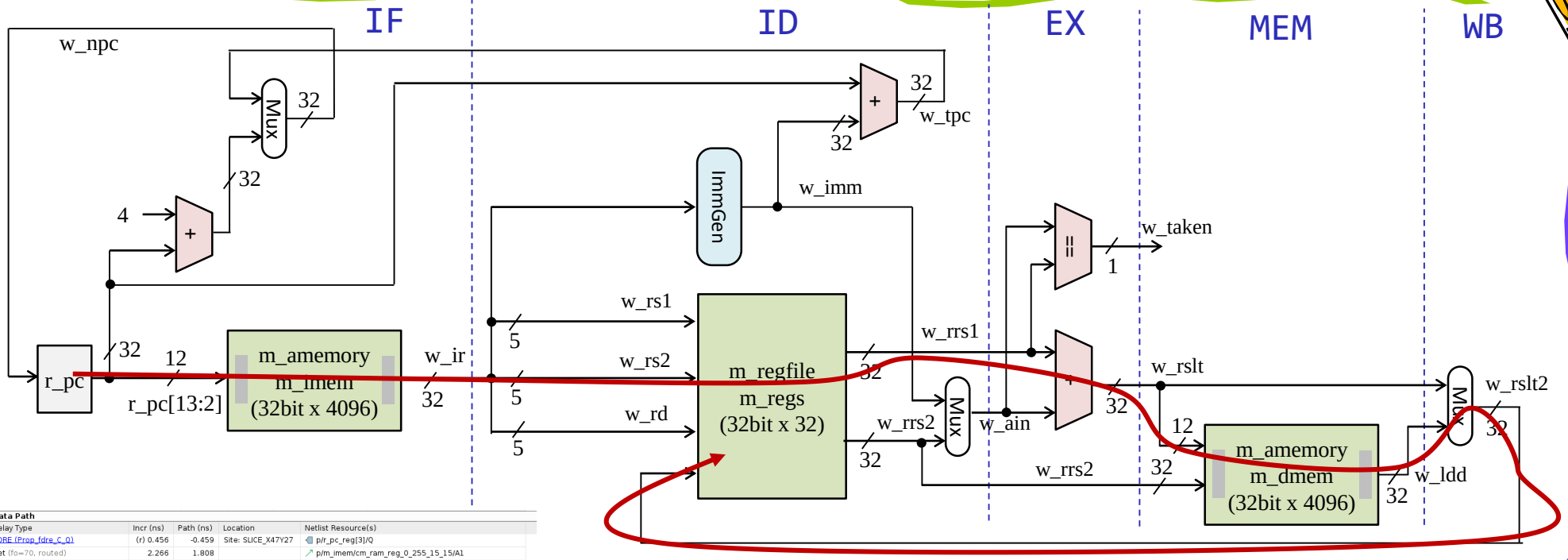
Graph | **Table**

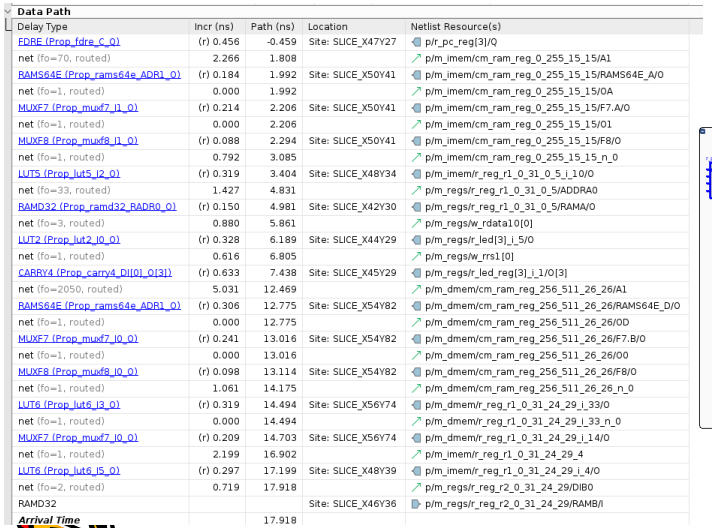
Resource	Utilization	Available	Utilization %
LUT	3095	20800	14.88
LUTRAM	2184	9600	22.75
FF	1199	41600	2.88
IO	5	210	2.38
BUFG	3	32	9.38
MMCM	1	5	20.00



この配置配線の結果は，実装によって変わる．

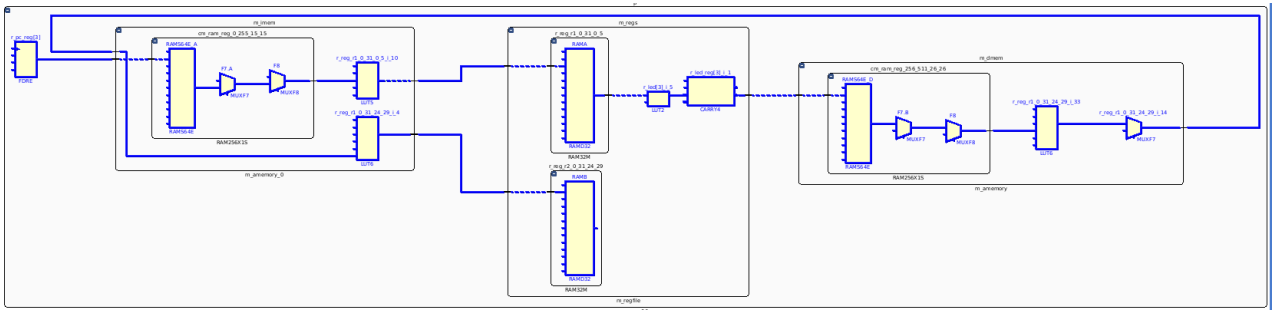
m_proc06 のクリティカルパスは





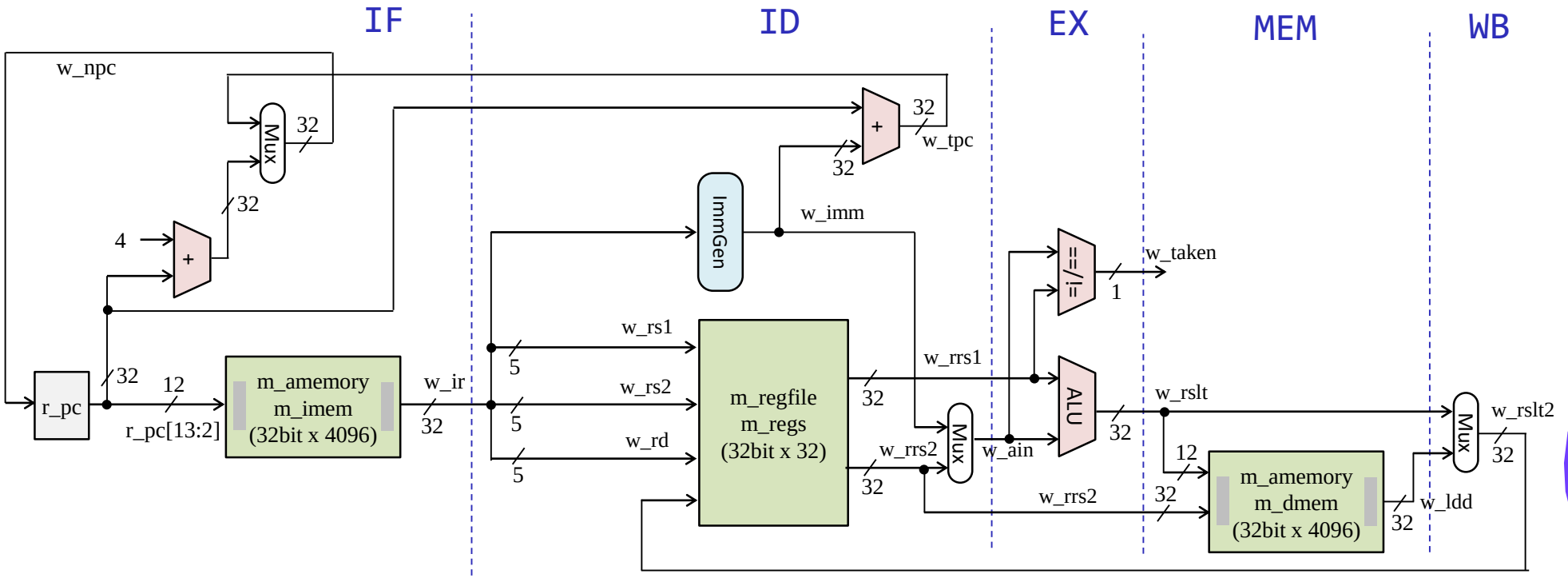
Data Path

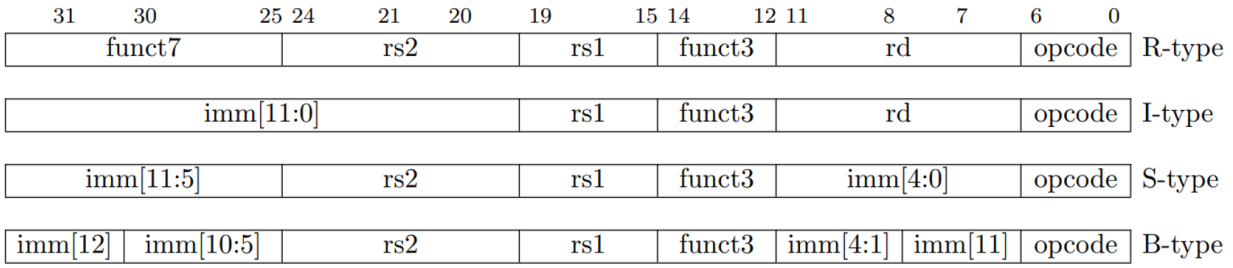
Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.456	-0.459	Site: SLICE_X47Y27	p/r_pc_reg[3]/Q
net (fo=70, routed)	2.266	1.808		p/m_imem/cm_ram_reg_0_255_15_15/A1
RAMS64E (Prop_rams64e_ADR1_O)	(r) 0.184	1.992	Site: SLICE_X50Y41	p/m_imem/cm_ram_reg_0_255_15_15/RAMS64E_A/O
net (fo=1, routed)	0.000	1.992		p/m_imem/cm_ram_reg_0_255_15_15/0A
MUXF7 (Prop_muxf7_I1_O)	(r) 0.214	2.206	Site: SLICE_X50Y41	p/m_imem/cm_ram_reg_0_255_15_15/F7.A/O
net (fo=1, routed)	0.000	2.206		p/m_imem/cm_ram_reg_0_255_15_15/01
MUXF8 (Prop_muxf8_I1_O)	(r) 0.088	2.294	Site: SLICE_X50Y41	p/m_imem/cm_ram_reg_0_255_15_15/F8/O
net (fo=1, routed)	0.792	3.085		p/m_imem/cm_ram_reg_0_255_15_15_n_0
LUT5 (Prop_lut5_I2_O)	(r) 0.319	3.404	Site: SLICE_X48Y34	p/m_imem/r_reg_r1_0_31_0_5_i_10/O
net (fo=33, routed)	1.427	4.831		p/m_regs/r_reg_r1_0_31_0_5/ADDRA0
RAMD32 (Prop_ramd32_RADR0_O)	(r) 0.150	4.981	Site: SLICE_X42Y30	p/m_regs/r_reg_r1_0_31_0_5/RAMA/O
net (fo=3, routed)	0.880	5.861		p/m_regs/w_rdata10[0]
LUT2 (Prop_lut2_I0_O)	(r) 0.328	6.189	Site: SLICE_X44Y29	p/m_regs/r_led[3]_i_5/O
net (fo=1, routed)	0.616	6.805		p/m_regs/w_rrs1[0]
CARRY4 (Prop_carry4_DI[0]_O[3])	(r) 0.633	7.438	Site: SLICE_X45Y29	p/m_regs/r_led_reg[3]_i_1/O[3]
net (fo=2050, routed)	5.031	12.469		p/m_dmem/cm_ram_reg_256_511_26_26/A1
RAMS64E (Prop_rams64e_ADR1_O)	(r) 0.306	12.775	Site: SLICE_X54Y82	p/m_dmem/cm_ram_reg_256_511_26_26/RAMS64E_D/O
net (fo=1, routed)	0.000	12.775		p/m_dmem/cm_ram_reg_256_511_26_26/0D
MUXF7 (Prop_muxf7_I0_O)	(r) 0.241	13.016	Site: SLICE_X54Y82	p/m_dmem/cm_ram_reg_256_511_26_26/F7.B/O
net (fo=1, routed)	0.000	13.016		p/m_dmem/cm_ram_reg_256_511_26_26/00
MUXF8 (Prop_muxf8_I0_O)	(r) 0.098	13.114	Site: SLICE_X54Y82	p/m_dmem/cm_ram_reg_256_511_26_26/F8/O
net (fo=1, routed)	1.061	14.175		p/m_dmem/cm_ram_reg_256_511_26_26_n_0
LUT6 (Prop_lut6_I3_O)	(r) 0.319	14.494	Site: SLICE_X56Y74	p/m_dmem/r_reg_r1_0_31_24_29_i_33/O
net (fo=1, routed)	0.000	14.494		p/m_dmem/r_reg_r1_0_31_24_29_i_33_n_0
MUXF7 (Prop_muxf7_I0_O)	(r) 0.209	14.703	Site: SLICE_X56Y74	p/m_dmem/r_reg_r1_0_31_24_29_i_14/O
net (fo=1, routed)	2.199	16.902		p/m_imem/r_reg_r1_0_31_24_29_4
LUT6 (Prop_lut6_I5_O)	(r) 0.297	17.199	Site: SLICE_X48Y39	p/m_imem/r_reg_r1_0_31_24_29_i_4/O
net (fo=2, routed)	0.719	17.918		p/m_regs/r_reg_r2_0_31_24_29/DIB0
RAMD32			Site: SLICE_X46Y36	p/m_regs/r_reg_r2_0_31_24_29/RAMB/I
Arrival Time		17.918		



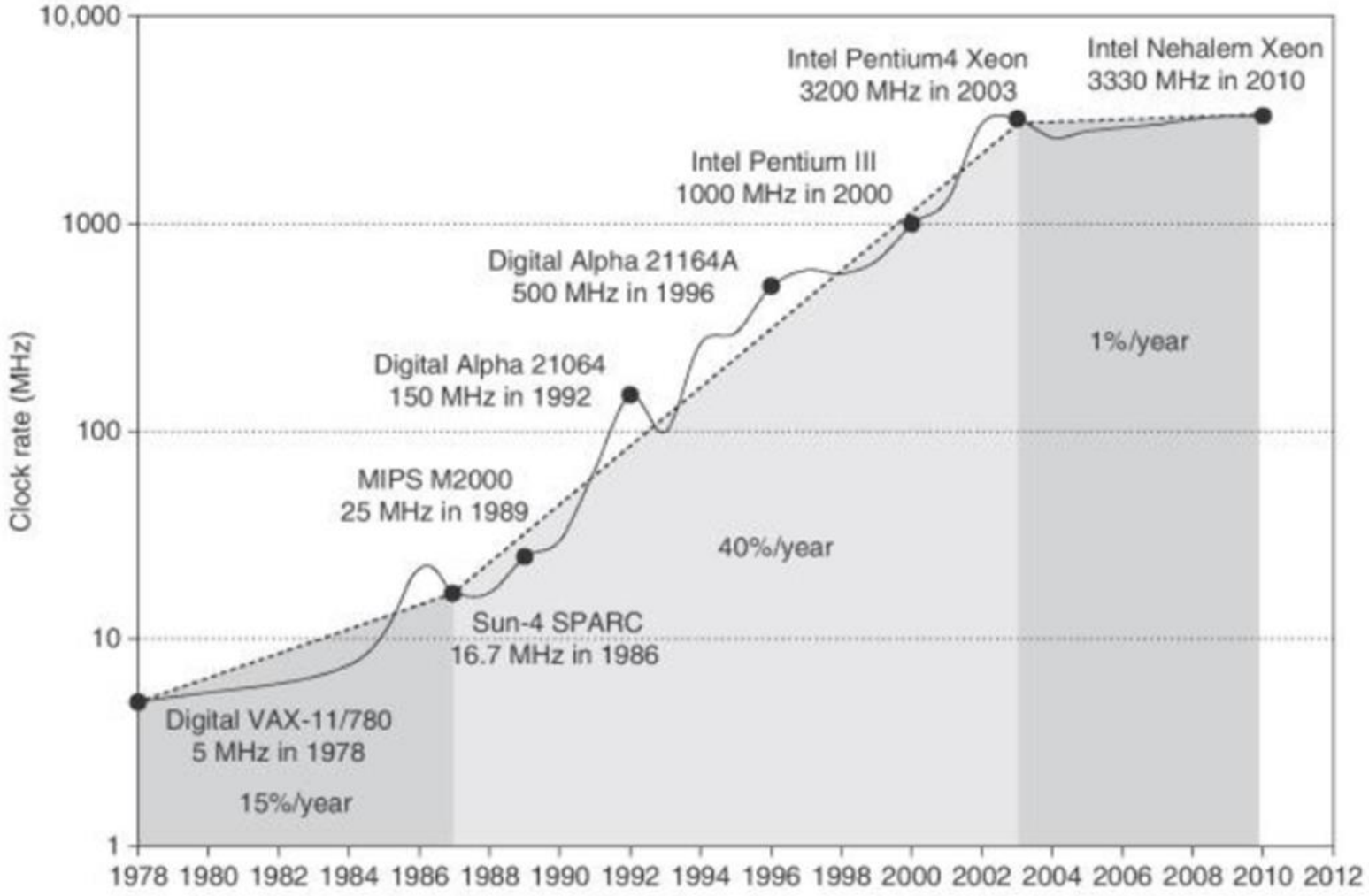
m_proc07 ベースラインのプロセッサ（シングルサイクル）

- 命令フェッチ(IF), デコードとオペランドフェッチ(ID), 実行(EX), メモリアクセス(MEM), ライトバック(WB) の処理をおこなうadd, addi, sll, srl, lw, sw, beq, bne命令に対応したプロセッサ





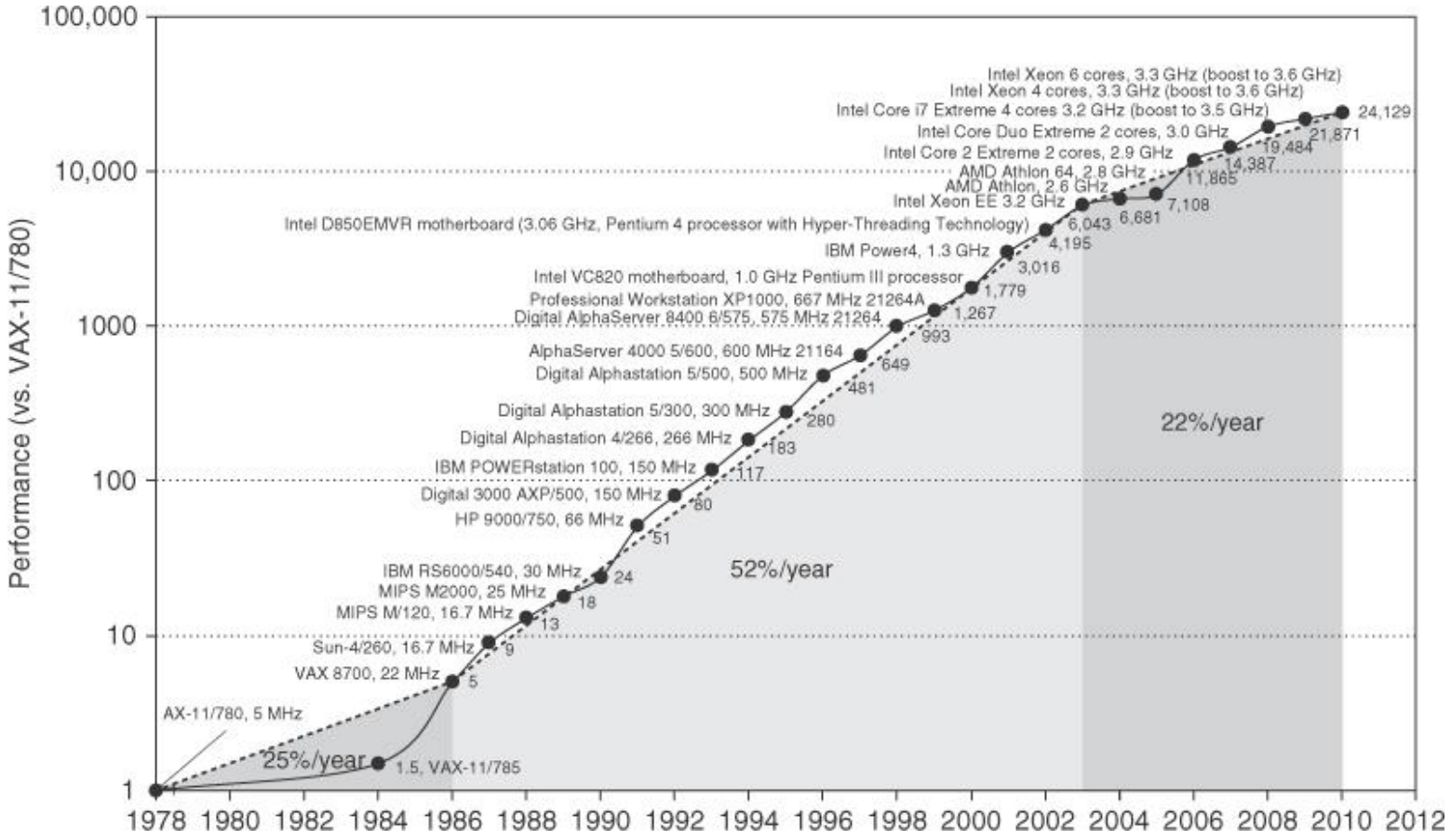
Growth in clock rate of microprocessors



From CAQA 5th edition

Growth in processor performance

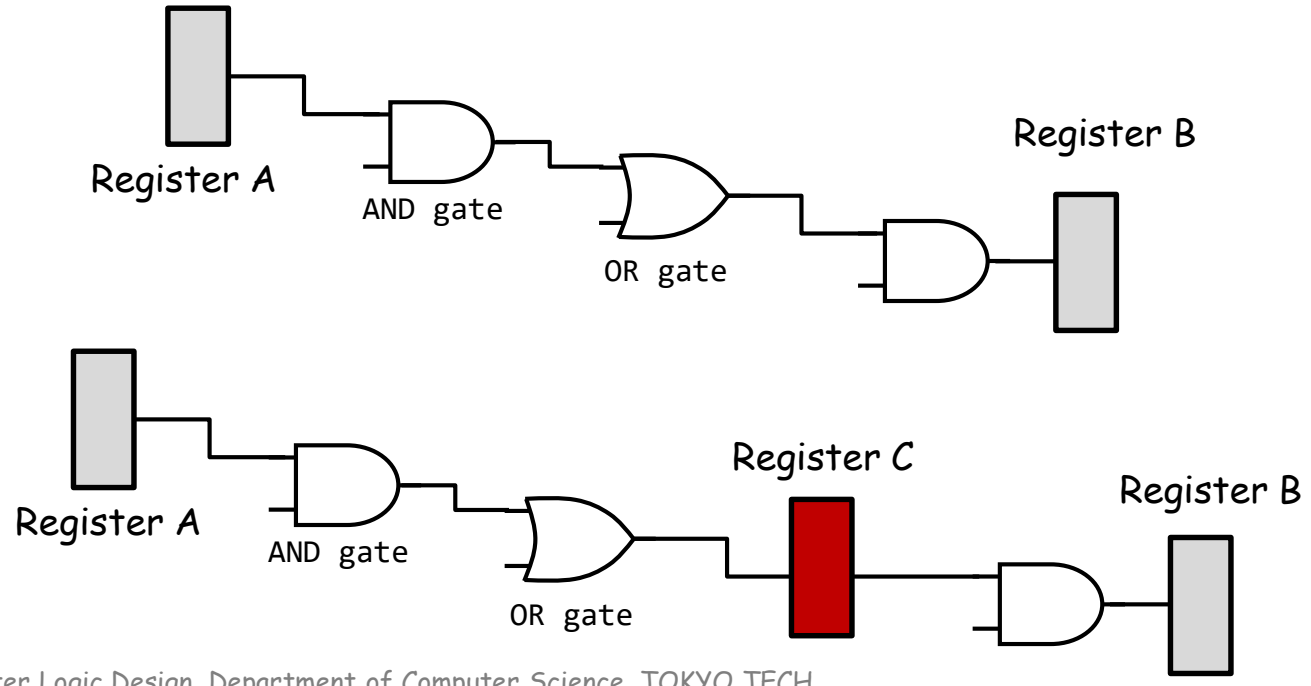
- Performance = f x IPC
  - f: frequency (clock rate)
  - IPC: retired machine Instructions Per Cycle



From CAQA 5th edition

Clock rate is mainly determined by

- Switching speed of gates (transistors)
- **The number of levels of gates**
  - The maximum number of gates cascaded in series in any combinational logics.
  - In this example, the number of levels of gates is 3.
- Wiring delay and fanout



References

- Computer Logic Design support page
  - https://www.arch.cs.titech.ac.jp/lecture/CLD/
- ACRi Room
  - https://gw.acri.c.titech.ac.jp
- ACRi Blog
  - https://www.acri.c.titech.ac.jp/wordpress/
- 情報工学系計算機室
  - http://www.csc.titech.ac.jp/
- Xilinx Vivado Design Suite
  - https://japan.xilinx.com/products/design-tools/vivado.html
- Digilent Arty A7-35T
  - https://reference.digilentinc.com/reference/programmable-logic/arty-a7/start
- Verilog HDL
  - https://ja.wikipedia.org/wiki/Verilog