Architectures for Computer Vision
from Algorithm to Chip with Verilog

Hong Jeong

©2014 John Wiley & Sons Singapore Pte Ltd. Published 2014 by John Wiley & Sons Singapore Pte Ltd.

March 13, 2015
Part 4 Verilog Design
Chapter 12 DP for Stereo Matching
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: The \((x, d)\) space: left/right and center reference systems.
Regions in the Search Space

\[ R^r = \{(x^r, d) | x^r \in [0, N - 1], d \leq D - 1 - \frac{D - 1}{N - 1} x^r\}, \]

\[ R^l = \{(x^l, d) | x^l \in [0, N - 1], d \leq \frac{D - 1}{N - 1} x^l\}, \]

\[ R^c = \{(x^c, d) | x^c \in [0, 2N - 2], d \leq \frac{D - 1}{N - 1}, d \leq \frac{D - 1}{N - 1}(2N - 2 - x^c), \]
\[ d \leq D - 1, x^c + d = \text{even}\}. \quad (1) \]
Figure: A graph for the search spaces. The size is either $D \times N$ or $2D \times (N - 1)$. 

(a) Neighborhood size, $N_e = D$
Figure: Two cases of connections: (a) $x + d = odd$ (b) $x + d = even$. 

(a) Right or left reference

(b) Center reference
Figure: Computing line and strip: left image ($I^l$), right image ($I^r$), and disparity map $D$. 

(a) Line

(b) Strip
Disparity Computation

\[ d(\cdot, y) \leftarrow T(I_l(\cdot, y), I_r(\cdot, y)). \]  (2)
Disparity Computation

\[ d(\cdot, y) \leftarrow T(\{I^l(\cdot, y') | y' \in [y - (L - 1)/2, y + (L - 1)/2]\}, \{I^r(\cdot, y') | y' \in [y - (L - 1)/2, y + (L - 1)/2]\} ). \quad (3) \]
Disparity Computation

\[ d(\cdot, y) \leftarrow T\left(\{I'(\cdot, y')|y' \in [y - (L - 1)/2, y + (L - 1)/2]\}, \{I'(\cdot, y')|y' \in [y - (L - 1)/2, y + (L - 1)/2]\}, d(\cdot, y - 1)\right). \]  

(4)
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: An epiplane plane and three coordinates: right, left, and center reference images (the disparity is shown as an inverse of the depth).
Right Disparity

\[ I^r(x^l, y) - I^l(x^l + d, y), \quad d \in [0, D - 1], \quad (5) \]
Left Disparity

\[ I^l(x^r, y) - I^r(x^r - d, y), \quad d \in [0, D - 1], \quad (6) \]
Center Disparity

\[ I^l((x^c + d - 1)/2, y) - I^r((x^c - d - 1)/2, y), \forall x + d = odd, \quad (7) \]
Relationships between Left, Right, and Center Systems

\[ x^l = \frac{1}{2}(x^c + d - 1), \]
\[ x^r = \frac{1}{2}(x^c - d - 1), \quad \forall x^c + d = odd. \quad (8) \]
Figure: The search space: \((x^r, d^r), (x^l, d^l),\) and \((x^c, d^c)\): black and white dots represent, respectively, matching and occlusion nodes.
Contents

Search Space
Line Processing
Computational Space

Energy Equations

DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Energy Function for Right System

\[
E(d) = \sum_{x=0}^{N-1} \rho(l^r(x), l^l(x + d)) + \lambda \mu(d(x), d(x - 1)),
\]  

(9)
Energy Function for Left System

\[ E(d) = \sum_{x=0}^{N-1} \rho(l^l(x), l^r(x - d)) + \lambda \mu(d(x), d(x - 1)). \quad (10) \]
Energy Function for Center System

\[
E(d) = \sum_{x=0}^{2N} \rho(I^l((x + d - 1)/2), I^r((x - d - 1)/2)) + \lambda \mu(d(x), d(x - 1)).
\]

(11)
Contents

Search Space
Line Processing
Computational Space
Energy Equations

DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
$$\begin{align*}
\begin{bmatrix}
\varphi(x - 1, d') \\
\eta(x - 1, d')
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x, d') \\
\eta(x, d')
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x + 1, d') \\
\eta(x + 1, d')
\end{bmatrix} \\
\begin{bmatrix}
\varphi(x, d) \\
\eta(x, d)
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x, d) \\
\eta(x, d)
\end{bmatrix}
\end{align*}$$

(a) Right reference system

$$\begin{align*}
\begin{bmatrix}
\varphi(x - 1, d') \\
\eta(x - 1, d')
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x, d) \\
\eta(x, d)
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x - 1, d') \\
\eta(x - 1, d')
\end{bmatrix} \\
\begin{bmatrix}
\varphi(x, d) \\
\eta(x, d)
\end{bmatrix} & \rightarrow \\
\begin{bmatrix}
\varphi(x, d) \\
\eta(x, d)
\end{bmatrix}
\end{align*}$$

(b) Left reference system

(c) Center reference system: matching node

(d) Center reference system: occlusion node

**Figure:** Paths from \((x - 1, d')\) to \((x, d)\) in the three reference systems (only two neighbor nodes are shown).
Algorithm (DP algorithm for the right reference system)

Given \((I^l(\cdot, y), I^r(\cdot, y))\), determine \(\{d(N - 1), \ldots, d(0)\}\).

1. Initialization: \(\varphi(0, d) = \rho(0, d), \eta(0, d) = 0\), for \(d \in [0, D - 1]\).
2. Forward pass: for \(x = 0, 1, \ldots, N - 1\) and \(d \in [0, D - 1]\),

\[
\varphi(x, d) = \min_{k \in [0, D - 1]} (\varphi(x - 1, k) + \mu(k, d)) + \lambda \rho(x, d),
\]

\[
\eta(x, d) = \arg\min_{k \in [0, D - 1]} (\varphi(x - 1, k) + \mu(k, d)).
\]

3. Finalization: \(d(N - 1) = \arg\min_{d \in [0, D - 1]} \varphi(N - 1, d)\).
4. Backward pass: for \(x = N - 2, \ldots, 0\),

\[
d(x) = \eta(x + 1, d(x + 1)).
\]
Algorithm (DP algorithm for the left reference system)

Given \((I^l(\cdot, y), I^r(\cdot, y))\), determine \(\{d(0), \ldots, d(N - 1)\}\).

1. **Initialization:** \(\varphi(N - 1, d) = \rho(N - 1, d), \ \eta(N - 1, d) = 0\), for \(d \in [0, D - 1]\).

2. **Forward pass:** for \(x = N - 1, \ldots, 1, 0\) and \(d \in [0, D - 1]\),

\[
\varphi(x, d) = \min_{k \in [0, D - 1]} \left( \varphi(x + 1, k) + \mu(k, d) \right) + \lambda \rho(x, d),
\]

\[
\eta(x, d) = \arg\min_{k \in [0, D - 1]} \left( \varphi(x + 1, k) + \mu(k, d) \right).
\]

3. **Finalization:** \(d(0) = \arg\min_{d \in [0, D - 1]} \varphi(0, d)\).

4. **Backward pass:** for \(x = 1, \ldots, N - 1\),

\[
d(x) = \eta(x - 1, d(x - 1)).
\]
Algorithm (DP algorithm for the center reference system)

Given \((l_l(\cdot, y), l_r(\cdot, y))\), determine \(\{(x, d(x))| i \in [0, 2N], d \in [0, D - 1]\}\).

1. Initialization: \(\rho(0, 0) = \rho(2N, 0) = 0, \rho(x, d) = \infty\) for \(x - d \leq 0\) or \(x + d \geq 2N\).

2. Forward pass: for \(x = 0, \ldots, 2N, d \in [0, D - 1]\),

   2.1 for matching node \((x + d = \text{odd})\),
   \[
   \varphi(x, d) = \min_{d' \in [0, D - 1]} \{\varphi(x - 1, d') + e_{j-d' + 1} \mu(d', d) + o_{j-k + 1} \alpha\} + \rho(x, d)
   \]
   \[
   \eta(x, d) = \arg\min_{d' \in [0, D - 1]} \{\varphi(x - 1, d') + e_{j-d' + 1} \mu(d', d)\}.
   \]

   2.2 for occlusion node \((x + d = \text{even})\),
   \[
   \varphi(x, d) = \varphi(x - 1, d) + \beta,
   \]
   \[
   \eta(x, d) = d.
   \]

3. Finalization: \(d(2N) = 0\).

4. Backward pass: for \(k \in [0, D - 1]\),

   \[
   d(x) = \begin{cases} 
   0, & x = 2N, \\
   \eta(x + 1, d(x + 1)), & x = 2N - 1, \ldots, 0.
   \end{cases}
   \]
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm

Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: The concept of the Verilog DP machine.
The Overall Operation

\[
D(\cdot, y, t) = F(I^l(\cdot, y), I^r(\cdot, y), D(\cdot, y, t - 1)),
\]

(12)
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture

Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: The state diagram of the DP machine.
Figure: The three modes: left, right, and center (LRC) reference modes.
Listing 1: The Verilog DP machine: framework (1/7)

```
'define WIDTH 113 //image width
'define HEIGHT 94 //image height
'define DATA_BITS 8 //word size
'define ADDR_BITS 15 //pixel counter
'define LINES 3 //strip size
'define LRC //LR or C reference
'define LEFT //LR reference
'define COST_BITS 10 //max cost range
'define INFTY 'COST_BITS'hFFFF //upper bound
'define DISPARITY_BITS 8 //disparity counter
'define DMAX 4 //max disparity
'define ALPHA 10 //penalty

module processor( //DP stereo processor
    input clock, reset,
    output reg ['ADDR_BITS - 1:0] raddr, r_waddr, //address bus
    input ['DATA_BITS - 1:0] i_rdata1, i_rdata2, r_rdata, //data bus
    output reg ['DATA_BITS - 1:0] r_wdata, //data bus
    output reg r_wen //write enable
    );
```
// working array: window of images
reg ['DATA_BITS - 1:0] img1 [0: 3*'WIDTH*'LINES -1]; // 1st image
reg ['DATA_BITS - 1:0] img2 [0: 3*'WIDTH*'LINES -1]; // 2nd image
reg ['DATA_BITS - 1:0] res [0: 3*'WIDTH*'LINES -1]; // disparity map

// variables
reg ['ADDR_BITS - 1:0] k, idx, idx1; // pixel
reg [9:0] i, j, J; // column, row
reg ['DISPARITY_BITS - 1:0] jj, pointer; // pointer
reg ['COST_BITS - 1:0] cost [0:'DMAX - 1], costp[0:'DMAX - 1]; // cost
ifdef LRC // LR reference
reg ['DISPARITY_BITS - 1:0] queue [0:'WIDTH - 1][0: 'DMAX - 1]; // LR
else // center reference
reg ['DISPARITY_BITS - 1:0] queue [0:2*'WIDTH][0: 'DMAX - 1]; // C
endif
reg [2:0] state, statef, stater; // state and substates
reg [9:0] count; // general counter
wire [7:0] xl, xr; // coordinates
wire ['COST_BITS - 1:0] ldistance; // local distance

// DP processing
always @ (posedge clock) begin: PROCESSING // processing block
if (reset) begin //initialize
    state <= 0; //global state
    count <= 0; //counter
    j <= 0; //image line
    k <= 0; //pixel in the strip
end
else begin: MAIN //state machine
    case (state)
    0: begin: BUFFER
        end
    1: begin: READING //fill the bottom
        end
    2: begin: INITIALIZATION //DP initialization
        end
    3: begin: FORWARD //DP forward pass
        end
    4: begin: FINALIZATION //DP finalization
        end
    5: begin: BACKWARD //DP backward pass
        end
    6: begin: WRITING
        end
    default: state <= 0; //fault recovery
endcase
end  //MAIN
end  //PROCESSING

//combinational circuits
endmodule
**Contents**

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme

**FIFO Buffer**

Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: The operation of the three buffers: left image ($I^l$), right image ($I^r$), and disparity ($D$).
Storing Disparity

\[ \text{res}(\cdot, (L - 1)/2) = T(\text{img}_1(\cdot, (L - 1)/2), \text{img}_2(\cdot, (L - 1)/2), \text{res}(\cdot, (L - 1)/2)). \]  

(13)
Figure: Three types of coordinates: image plane, buffer, and array.
Mapping Image to Array

\[ i = 3(Na + b) + c. \] (14)
Determining Index

\[ x = 3b + c, \]
\[ y = (y' - (L - 1)/2 + a + M) \% M. \]  (15)
function ['ADDR_BITS - 1:0] id;
    input signed [9:0] row, column, channel;
    begin
        id = 3*('WIDTH*((('LINES-1)>>1)+ row) +column) + channel;
    end
endfunction
Listing 2: The Verilog DP machine: Buffer (2/7)

//DP processing
0: begin: BUFFER //shift buffer
if (k < 3 * 'WIDTH * ('LINES -1)) begin //3 * pixels
  img1[k] <= img1[k + 3*'WIDTH];
  img2[k] <= img2[k + 3*'WIDTH];
  res [k] <= res [k + 3*'WIDTH];
k <= k + 1'b1;
end
else begin //go to the next state
  state <= 1;
  k <= 0; //initialize variable
  idx1 <= 0; //for next state
  idx <= 0; //for next state
end
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer

Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Reading Data from RAM

\[ \text{img}_1(\cdot, L - 1) \leftarrow I^l(\cdot, y), \text{img}_2(\cdot, L - 1) \leftarrow I^r(\cdot, y), \text{res}(\cdot, L - 1) \leftarrow D(\cdot, y), \]
(16)
Storing Result to RAM

\[ D(\cdot, J) \leftarrow \text{res}(\cdot, L - 1), \quad (17) \]
Listing 3: The Verilog DP machine: Reading and writing (3/7)

```verilog
1: begin: READING //fill the bottom
if (j < `HEIGHT) begin //line number
if (k < 3* `WIDTH + 2) begin //pixel
state <= 1; //repeat state
raddr <= 3 * `WIDTH * j + k; //pixel address
img1[3*`WIDTH*(`LINES-1) + idx1] <= i_rdata1; //1st image
img2[3*`WIDTH*(`LINES-1) + idx1] <= i_rdata2; //2nd image
res [3*`WIDTH*(`LINES-1) + idx1] <= r_rdata;
idx1 <= idx;
idx <= k; //delay
k <= k + 1'b1; //delay
end else begin //next block
end else begin
state <= 2; //go to the next state
count <= 0;
J <= (`LINES == 1)? j :
((j - ((`LINES - 1) >>1) + `HEIGHT) % `HEIGHT);
j <= j + 1'b1; //next strip
end
end else begin
j <= 0; //hit the bottom
end //else
```
end

6: begin: WRITING
if (count < ‘WIDTH) begin
if (k < 3) begin //make 3 channels
    r_wdata <= res[id(0,count,0)]; //data
    r_waddr <= 3*‘WIDTH*J + 3*count + k; //address
    r_wen <= 1; //write enable
    k <= k + 1;
end
else begin
    count <= count + 1’b1; //next
    k <= 0;
end
end
else begin
    state <= 0;
    k <= 0;
    count <= 0;
end
end
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Initialization for the Right System

\[ \varphi(0, j) = |I_R^r(0) - I_R^l(j)| + |I_G^r(0) - I_G^l(j)| + |I_B^r(0) - I_B^l(j)|, \quad j \in [0, D - 1]. \quad (18) \]
Initialization for the Left System

\[ \varphi(0, j) = |I^l_R(N - 1) - I^r_R(N - 1 - j)| + |I^l_R(N - 1) - I^r_R(N - 1 - j)| 
+ |I^l_R(N - 1) - I^r_R(N - 1 - j)|, \quad j \in [0, D - 1]. \]  

(19)
Initialization for the Center System

\[ \phi(0, j) = \begin{cases} 
0, & j = 0, \\
\infty, & j \in [1, D - 1]. 
\end{cases} \] (20)
Initialization for the Pointer

\[ \eta(0,j) = 0, \quad \forall j \in [0, D - 1]. \] (21)
Listing 4: The Verilog DP machine: Initialization (4/7)

```verilog
2: begin: INITIALIZATION //DP initialization
if (count < 'DMAX) begin //for each disparity
queue[0][count] <= 0; //initialize the queue
count <= count + 1'b1; //next in the queue
'ifdef LRC //LR mode
  'ifdef LEFT //left disparity
    costp[count] <=
        distance(img1[id(0,('WIDTH - 1),0)], img2[id(0,('WIDTH - 1 - count),0)])
        + distance(img1[id(0,('WIDTH - 1),1)], img2[id(0,('WIDTH - 1 - count),1)])
        + distance(img1[id(0,('WIDTH - 1),2)], img2[id(0,('WIDTH - 1 - count),2)]);
  'else //right disparity
    costp[count] <= distance(img1[id(0,count,0)], img2[id(0,0,0)])
        + distance(img1[id(0,count,1)], img2[id(0,0,1)])
        + distance(img1[id(0,count,2)], img2[id(0,0,2)]);
  'endif
  'else //center reference mode
    costp[count] <= (count)? 'INFTY: 0; //assign large number
  'endif
```
end
else begin
state <= 3; //next state
statef <= 0; //next sub-state
i <= 1; //1st pixel
jj <= 0; //zero disparity
end
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Figure: The \((x, d)\) space: left/right and center reference systems.
Figure: The computation of the forward pass: \( \varphi(x, d) \) and \( \eta(x, d) \) (the value in the parenthesis is for the center reference system).
Listing 5: The Verilog DP machine: Forward pass (5/7)

3: begin: FORWARD                          //DP forward pass
   ‘ifdef LRC
      if (i < ‘WIDTH) begin: COLUMN         //for each column
         ‘else
            if (i < 2*‘WIDTH + 1’b1) begin: COLUMN     //for each column
               ‘endif
            if (jj < ‘DMAX) begin: ROW             //for each disparity
               case (statef)
               0: begin: COST_INIT                 //initialization
                  statef <= 1;                      //next sub-state
                  count <= 0;                      //parent index
                  cost[jj] <= ‘INFTY;             //cost reset
                  queue[i][jj] <= ‘DMAX - 1;    //queue reset
               end
               1: ‘ifdef LRC
                  if (jj <‘WIDTH - i) begin: TRAPEZOID_IN //matching
                     if(count < ‘DMAX) begin: COMPARISON //shortest path
                        if (costp[count] < ‘INFTY) begin //avoid overflow
                           queue[i][jj]<= ((costp[count] + weight(count,jj))
                                          < cost[jj])?(count): queue[i][jj];
                        cost[jj] <= ((costp[count] + weight(count,jj))
                           < cost[jj])?(count): queue[i][jj];
                  end
               ‘endif
            ‘endif
      ‘endif
   ‘endif
   ‘endif
< cost[jj])?(costp[count] + weight(count,jj))
: cost[jj];

end

count <= count + 1'b1; //for each parent
end

else begin: COST_UPDATE //add ldistance
  ifdef LEFT //left disparity
  cost[jj] <= (ldistance + cost[jj])>>1;
  'else //right disparity
  cost[jj] <= (ldistance + cost[jj])>>1;
  'endif

statef <= 0;
jj <= jj + 1'b1; //repeat disparity
end

stater <= 0; //if not upper TRAPEZOID
end

'else
if ((jj <= i - 1) & (jj <= 2*'WIDTH - i - 1)) begin: TRAPEZOID_IN
if ((i + jj) % 2 == 1) begin: MATCHING_NODE
if (count < 'DMAX) begin: COMPARISON
  if (costp[count] < 'INFTY) begin //avoid overflow
    queue[i][jj]<= ((costp[count] + weight(count,jj))
    < cost[jj])?(count): queue[i][jj];
    cost[jj] <= ((costp[count] + weight(count,jj))
    < cost[jj])?(count): cost[jj];
  end
end
end
< cost[jj])?(costp[count] + weight(count,jj))
: cost[jj];

end

count <= count + 1'b1; //for each parent
end

else begin: COST_UPDATE //add ldistance
cost[jj] <= (ldistance + cost[jj])>>1; //normalize
jj <= jj + 1'b1; //repeat disparity
statef <= 0;
end

end

else begin: OCCLUSION_NODE //outside TRAPEZOID
statef <= 0;
jj <= jj + 1'b1;
queue[i][jj] <= jj;
cost[jj] <= costp[jj] + ‘ALPHA; //assign penalty
end

stater <= 0; //if not upper TRAPEZOID
‘endif

else begin: TRAPEZOID_OUT //unobservable region
statef <= 0;
stater <= 0;
cost[jj] <= ‘INFTY; //assign big number

Hong Jeong (©2014 John Wiley & Sons Singapore Pte Ltd.)
jj <= jj + 1'b1; //next disparity
end
   endcase
end //ROW
   else begin //parent update
   case (stater) //initialization
0: begin //next sub-state
stater <= 1;
count <= 0;
end
1: begin: PARENT //reset variable
   if (count < ‘DMAX) begin //for each disparity
      stater <= 1; //repeat state
      costp[count] <= cost[count]; //copy
      count <= count + 1'b1; //next disparity
   end
else begin //reset sub-state
   stater <= 0;
jj <= 0; //reset disparity
   i <= i + 1'b1; //next pixel
end
   endcase
end
end //COLUMN
else begin
state <= 4;
end
end //FORWARD

//go to the next state
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
\[ \eta(N - 1, \cdot) (\eta(2N, \cdot)) \]

\[ x \leftarrow x - 1 \]

\[ x > 0 \]

**Figure:** The backward pass. (The expression in the parenthesis is for the center reference system. Others are for the left and right reference systems.)
4: begin: FINALIZATION //DP finalization
state <= 5; //go to the next state
pointer <= 0; //starting pointer
count <= 0; //reset counter
end

5: begin: BACKWARD //DP backward pass
‘ifdef LRC //LR reference
if (count < ‘WIDTH) begin
pointer <= queue[‘WIDTH - 1 - count][pointer]; //recursion
‘ifdef LEFT
res[id(0, count, 0)] <= queue[‘WIDTH - 1 - count][pointer];
‘else
res[id(0, (‘WIDTH - count), -3)] <= queue[‘WIDTH-1-count][pointer];
‘endif
‘else //center reference
if (count < 2*‘WIDTH + 1'b1) begin
pointer <= queue[‘WIDTH*2 - count][pointer];
‘ifdef LEFT //center left reference
res[id(0, (‘WIDTH - ((count+pointer-1)>>1)), -3])
<= queue[2*‘WIDTH - count][pointer];
‘else //center right reference
pointer <= queue[‘WIDTH*2 - count][pointer];
res[id(0,(‘WIDTH - ((count-pointer-1)>>1)), - 3)]
    <= queue[2*‘WIDTH - count][pointer];
‘endif
‘endif
count <= count + 1’b1;
end else begin
state <= 6; //next state
count <= 0;
end
end
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
Smoothness Term

\[ \mu(k, j) \triangleq |k - j|. \]  \hspace{1cm} (22)
Local Distance Measure

\[ \rho(i, j) \triangleq |I^l_R(i) - I^r_R(j)| + |I^l_G(i) - I^r_G(j)| + |I^l_B(i) - I^r_B(j)|. \]  

(23)
Listing 7: The Verilog DP machine: functions (7/7)

//coordinates transformation: (row,column,channel) -> id
function ['ADDR_BITS - 1:0] id;
input signed [9:0] row, column, channel;
begin
    id = 3*('WIDTH*((('LINES-1)>>1)+ row) +column) + channel;
end
endfunction

//absolute distance measure with threshold
function ['DATA_BITS - 1:0] weight; //smoothness constraint
input ['DATA_BITS - 1:0] a, b;
begin
    weight = (((a > b)? (a-b) : (b - a)) < 5)?
        (((a > b)? (a-b) : (b - a)): 'DMAX; //threshold
end
endfunction

//image intensity distance measure
function ['DATA_BITS - 1:0] distance; //intensity distance
input ['DATA_BITS - 1:0] a, b;
begin
distance = (a > b)? (a-b): (b - a); //distance measure
end
endfunction

//corresponding points for the given disparity
‘ifdef LRC
‘ifdef LEFT //left coordinate
assign xl = ‘WIDTH-1-i;
assign xr = ‘WIDTH-1-i-jj; //right coordinate
‘else //right coordinate
assign xl = i + jj; //left coordinate
assign xr = i; //right coordinate
‘endif
‘else
assign xl = (i + jj - 1)>>1; //left coordinate
assign xr = (i - jj - 1)>>1; //right coordinate
‘endif

//local distance: four-neighborhood
assign ldistance =
  distance(img1[id(0,xl,0)],img2[id(0,xr,0)]) //center pixel
+ distance(img1[id(0,xl,1)],img2[id(0,xr,1)])
+ distance(img1[id(0,xl,2)],img2[id(0,xr,2)])
//south neighborhood
+ ((\textsc{LINES} < 2)? 0:
+ ((\text{distance(img1[id(1,xl,0)],img2[id(1,xr,0)])}) //south neighbor
  + distance(img1[id(1,xl,1)],img2[id(1,xr,1)]))
  + distance(img1[id(1,xl,2)],img2[id(1,xr,2)]))
//north neighborhood
+ distance(img1[id(-1,xl,0)],img2[id(-1,xr,0)]) //north neighbor
+ distance(img1[id(-1,xl,1)],img2[id(-1,xr,1)]))
+ distance(img1[id(-1,xl,2)],img2[id(-1,xr,2)]))
//east neighborhood
+ (((xl < \texttt{WIDTH} - 1) && (xr < \texttt{WIDTH} - 1))? //east neighbor
  distance(img1[id(0,(xl+1),0)],img2[id(0,(xr+1),0)]) //boundary
  + distance(img1[id(0,(xl+1),1)],img2[id(0,(xr+1),1)]))
  + distance(img1[id(0,(xl+1),2)],img2[id(0,(xr+1),2)]): 0)
//west neighborhood
+ (((xl > 0) && (xr > 0))? //west neighbor
  distance(img1[id(0,(xl-1),0)],img2[id(0,(xr-1),0)]) //boundary
  + distance(img1[id(0,(xl-1),1)],img2[id(0,(xr-1),1)]))
  + distance(img1[id(0,(xl-1),2)],img2[id(0,(xr-1),2)]): 0)>>2));
Contents

Search Space
Line Processing
Computational Space
Energy Equations
DP Algorithm
Architecture
Overall Scheme
FIFO Buffer
Reading and Writing
Initialization
Forward Pass
Backward Pass
Combinational Circuits
Simulation
**Figure:** Disparity maps: point operations.
Figure: Disparity maps: four-neighborhood operations.