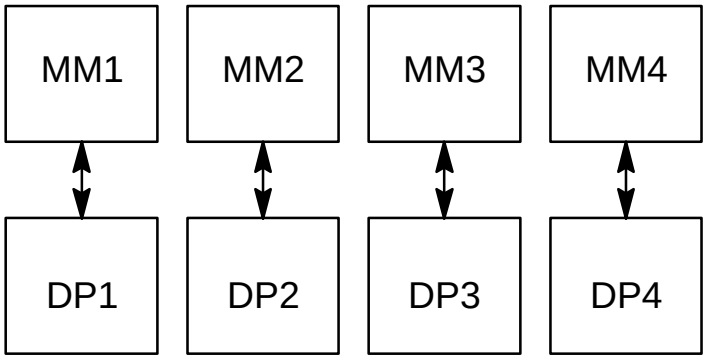
Solutions for Chapter 9

Exercise 9.1

a.

The vectors x can be processed independently. The method of choice, therefore, is to distribute the vectors. The matrix W can be kept locally in each of the memory modules or be distributed across the memory modules, resulting in a decrease of the memory size required. The disadvantage of the latter approach is a slight increase in the complexity of the communication network connecting the memories and the data paths, because the network must provide a broadcast of components of W to all data paths. During processing only one matrix element at a time must be broadcast to the data paths. A single data bus is sufficient to provide this functionality. Under these assumptions, the following two implementation variants can be derived:

Variant I.)



Data accesses:

local: x(0), x(1), ..., x(N–1), x(0), x(1), ..., ...

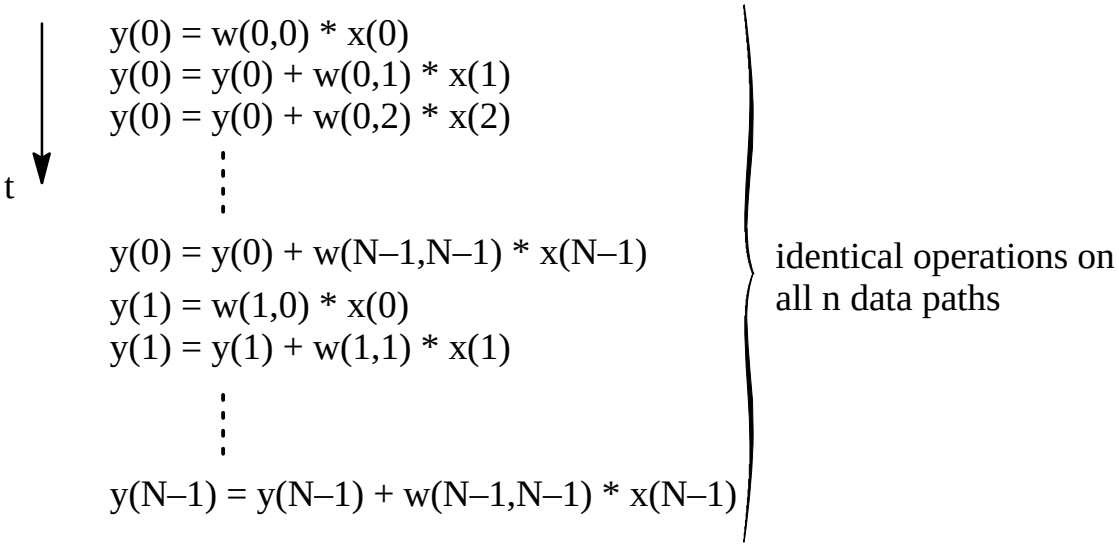
n times

broadcast:

w(0,0), w(0,1), w(0,2), ..., w(0,N–1),w(1,0)w(1,1), ... w(N–1,N–1)

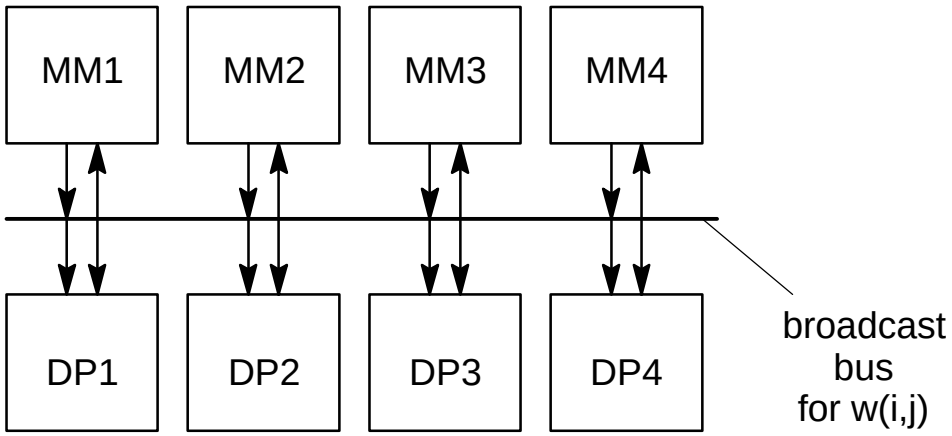
t

Sequence of operations:



Note: The elements of W are complex valued! All operations as mentioned ab-above must be executed for the real and imaginary parts of w(i,j), respectively.

Variant II.)



The sequence of data accesses is analogue to those described above, except that all w(i,j) are broadcast to all data paths.

The sequence of operations is identical to the sequence described for variant I.)

b.

The parallelisation strategy is to distribute the calculation of the result values. In order to do so, the rows of the matrix W are distributed across the available memory modules, i.e., each data paths stores four consecutive rows of W locally. The input data x(i) is broadcast sequentially to all data paths. Each data path then calculates

four elements of the result vector. The same architecture as presented in variant II in problem part a can be used. The sequence of operations is essentially the same as shown in part a.

Data accesses for N=16 and 4 data paths:

Broadcast: $\underbrace{x(0), x(1), ..., x(15)}_{16 \text{ times}}$, x(0), x(1), ..., ...

local accesses:

in parallel
- w(0,0),w(0,1),w(0,2),...,w(0,15),w(1,0),w(1,1),...,w(3,15)
- w(4,0),w(4,1),w(4,2),...,w(4,15),w(5,0),w(5,1),...,w(7,15)
- w(8,0),w(8,1),w(8,2),...,w(8,15),w(9,0),w(9,1),...,w(11,15)
- w(12,0),w(12,1),w(12,2),...,w(12,15),w(13,0),w(13,1),...,w(15,15)

$\longrightarrow$ t

c.

For n vectors to be processed in parallel using the FFT algorithm, the data distribution for input data and coefficients can be chosen as presented in the solution for part a of this problem. All vectors can be processed independently of each other. Depending on the FFT algorithm selected, the sequence of operations to be executed changes. Please refer to chapter 7 for possible FFT implementations and the associated sequence of operations.

d.

If only one vector is to be processed at a time, a number of different implementations, depending on the FFT algorithm selected, are possible. Assuming that a decimation in frequency algorithm is selected (refer to chapter 7) and that each of the n=4 PEs stores 4 input values in its associated memory bank in the beginning, the operations of each stage of the signal flow graph are distributed across the PEs.

Assuming that the associated memory banks are used to store intermediate results, it is necessary that PEs have access to other memory banks, too. For example, if PE i stores input values 4*i to 4*i+3, $0 \leq i \leq 3$, then PE 0 has to access memory 2 during the operations of stage 0, memory 1 during the operations of stage 1, and its own memory 0 in stages 2 and 3. A communication network that supports these accesses in parallel is required for maximum performance. A candidate for such a network is, e.g., a 2–stage omega network (refer to chapter 9 for an illustration).

Exercise 9.2

a.

Task distribution:

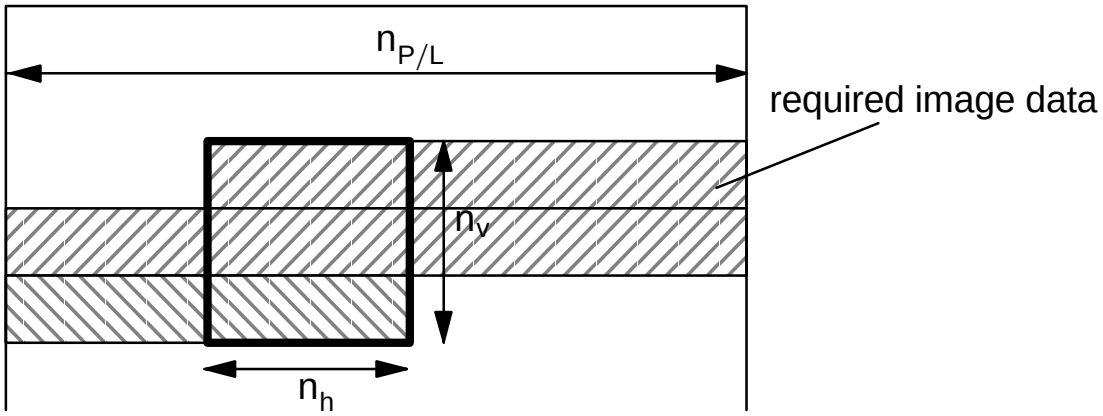
The filter coefficients are distributed across the set of data paths. Each data path then multiplies its coefficient with the corresponding input value. The summation of these intermediate results is performed row-wise and then column-wise (for $n_v \geq n_h$) or column-wise and then row-wise (für $n_h > n_v$), respectively.

Achievable parallelism:

Step 1:	Multiplication	$P = n_h \cdot n_v = 25$
Step 2:	Row-wise summation	$P = n_v = 5$
Step 3:	Column-wise summation	$P = 1$

Memory requirements:

Input data is provided row-wise, which requires to store several rows of a frame for processing (see figure).



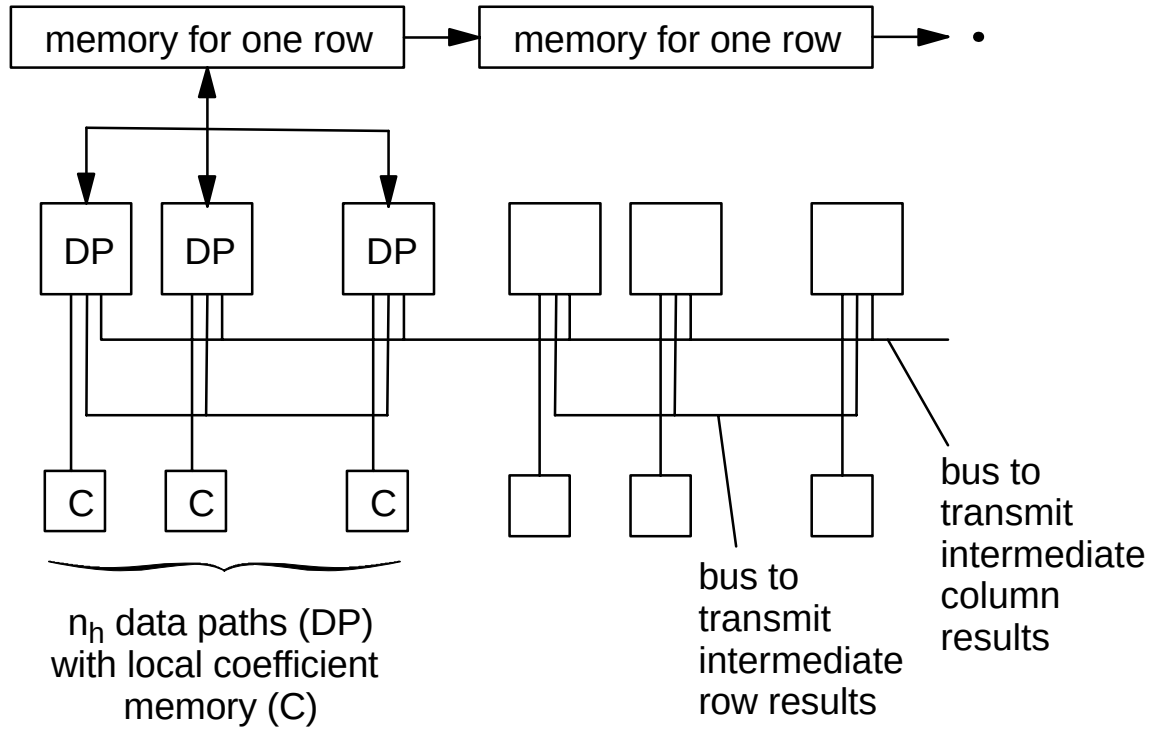
For the general case the required memory S_m can be calculated as:

$$S_m = \underbrace{(n_v - 1) \cdot n_{P/L} + n_h}_{\text{image data}} + \underbrace{n_v \cdot n_h}_{\text{coefficients}} + \underbrace{n_v}_{\text{intermediate results}}$$

Using the values as given in the problem we get:

$$S_m = 4 \cdot 512 + 5 + 5 \cdot 5 + 5 = 2083$$

Memory architecture:



Remark:

It would be possible to reduce the number of processing steps by using a tree-summing approach to calculate the intermediate row and column sums. This, however, would result in a communication network with increased complexity (any two data paths must be able to exchange intermediate results simultaneously).

Data transfer rates

Input data:

$$R_{T,IN} = n_{P/L} \cdot n_{L/F} \cdot f_F = 512 \cdot 512 \cdot 25 = 6.55 \cdot 10^6 \, pels/sec$$

Output data: For each input pixel, one output value is calculated. Therefore,

$$\rightarrow R_{T,OUT} = R_{T,IN}$$

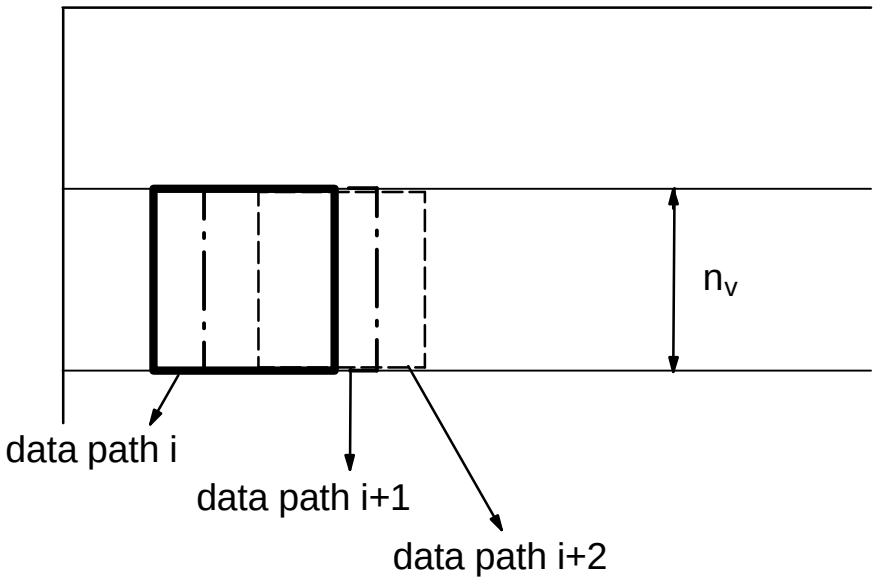
The total data rate required is the sum of both:

$$R_{T,TOTAL} = R_{T,IN} + R_{T,OUT} = 2 \cdot n_{P/L} \cdot n_{L/F} \cdot f_F = 13.11 \cdot 10^6 \, pels/sec$$

b.

Data distribution:

Each data path now calculates one output value on its own. The required input data distribution is shown in the following figure. Note that for any given set of output values that are computed in parallel, some input data is required by several data paths.



Achievable Parallelism:

Maximum parallelism is achieved if the calculation of all output values for a given frame is performed in parallel, with one output value per data path, i.e.,

$$P_{\max} = n_{P/L} \cdot n_{L/F} = 512 \cdot 512 = 262144$$

Remember, however, that input data is provided row-wise in a sequential manner. Assume that each data path needs a time of $T_{C,O}$ to calculate one output value. Considering these two facts, it is of interest to calculate how much parallelism is needed to perform the filtering operation in real time.

For each input value, one output value is calculated. The parallelism needed to satisfy the real-time requirement can be calculated as:

$$P_{REQUIRED} = T_{C,O} \cdot R_{T,IN}$$

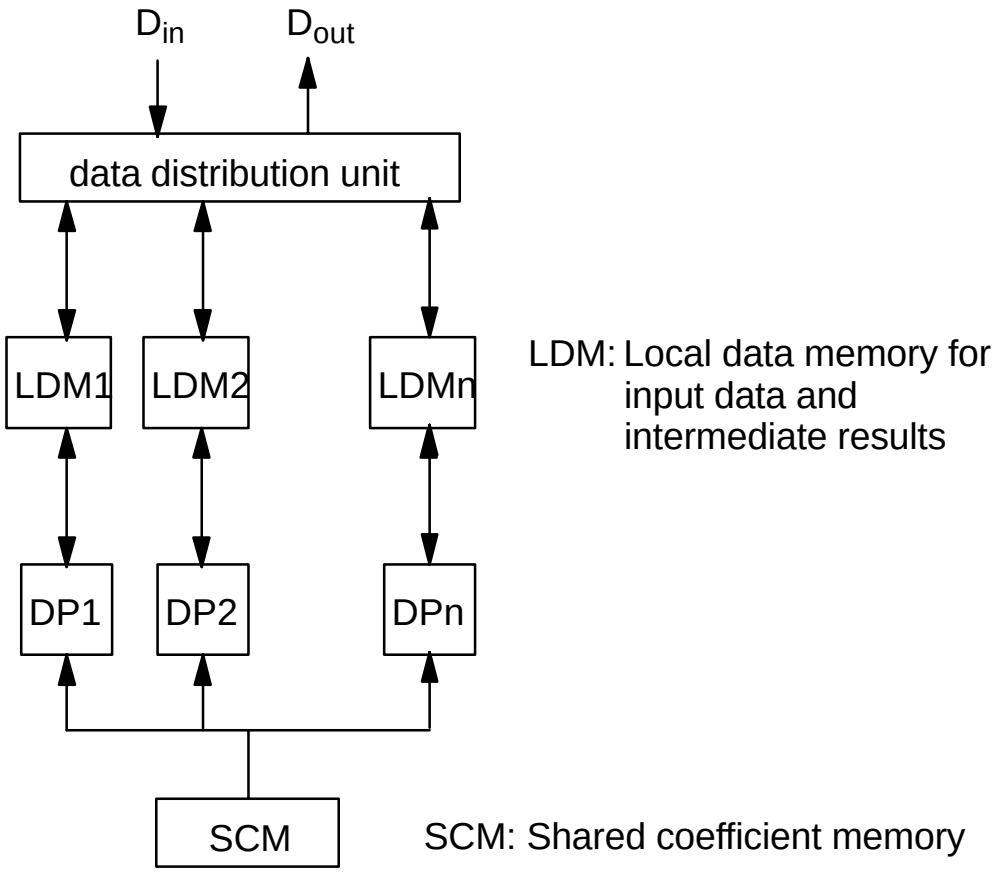
with $R_{T,IN} = n_{P/L} \cdot n_{L/E} \cdot f_F$ (input data rate).

Example:

$$T_{C,O} = 1\mu s\,; R_{T,IN} = 512 \cdot 512 \cdot 25\ 1/s \rightarrow P_{REQUIRED} = 6.56$$

In this case 7 data paths are necessary. It is not useful to use more, because the additional data paths would have no input data to work on.

Architecture:



A useful architecture consists of n data paths. A shared coefficient memory provides the coefficients to all data paths. The input data for each data path are stored in a local data memory. These memories are also used to store intermediate results. A data distribution unit takes care of transfering the input data to the corresponding memories according to the above distribution and retrieving the results.

Memory requirements:

SCM: Coefficients only $\rightarrow S_{n,SCM} = n_h \cdot n_v = 5 \cdot 5 = 25$

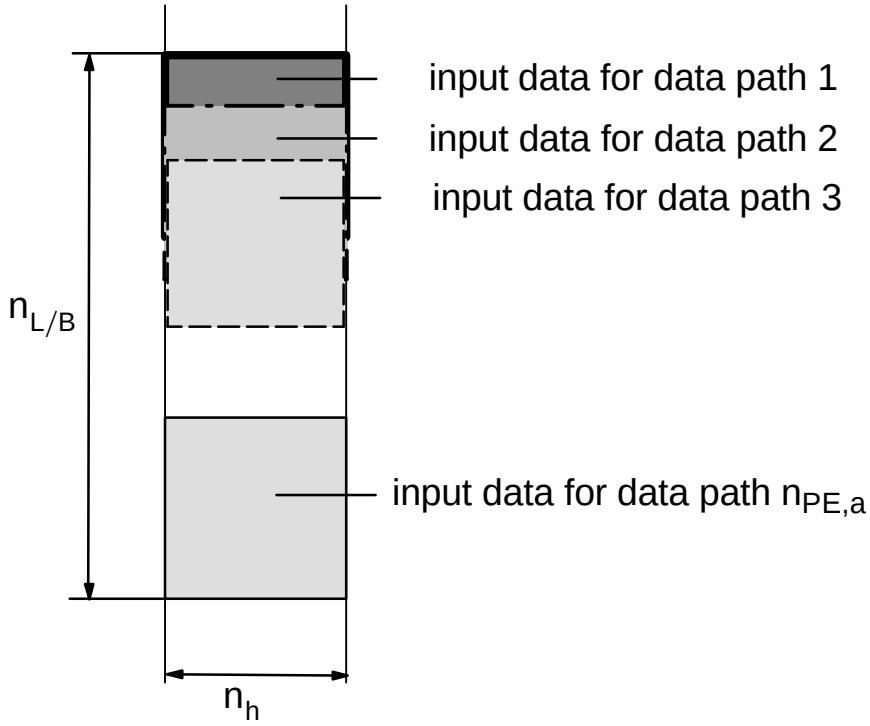
LDM: Input data according to the size of the filter window, in addition one storage location for intermediate results $\rightarrow S_{n,LDM} = n_v \cdot n_h + 1 = 26$

c.

Case 1: $n_{PE,a} = n_{L/B} - n_v + 1$

Shared memory size:

A total of $n_{PE,a}$ output values can be calculated in parallel. The required input data can be used simultaneously in several data paths:



The minimum size of the shared memory of the multiprocessor can be calculated as:

$$S_{M,MP} = n_{L/B} \cdot n_h = 8 \cdot 5 = 40$$

Data transfer rate:

For each vertical scan of the block line memory, $n_{L/B}$ input values are transfered from the block line memory to the processor. For $n_{L/B}$ input values transferred to the processor, $n_{PE,a}$ output values are calculated. To satisfy the real-time requirements, the data transfer rate to the processor must be:

$$R_{D,BL\ to\ MP,a} = R_{D,OUT} \cdot \frac{n_{L/B}}{n_{PE,a}} = n_{P/L} \cdot n_{L/F} \cdot f_F \cdot \frac{n_{L/B}}{n_{L/B} - n_v + 1}$$

Using the numbers as given above, we get:

$$R_{D,BL\ to\ MP,a} = R_{D,OUT} \cdot \frac{n_{L/B}}{n_{PE,a}} =$$

$$6.55 \cdot 10^6\ pels/sec \cdot \frac{8}{4} = 13.11 \cdot 10^6\ pels/sec$$

Case 2: $n_{PE,b} = (n_{L/B} - n_v + 1)^2$

Shared memory size:

In this case, a vertical (similar to case 1) and horizontal overlapping of input data is used. The data paths are arranged in a square shape with $n_{PE,v}$ in the vertical and $n_{PE,h}$ in the horizontal direction, with $n_{PE,v} = n_{PE,h}$. The minimum capacity of the shared memory is therefore:

$$S_{M,MP} = n_{L/B}^2$$

Data transfer rate:

To calculate $n_{PE,b}$ output values, $n_{L/B}^2$ input values are required. Remember that new input values are transfered to the processor by means of a meander scan of the block line memory. After a set of new output values has been calculated, some of the input data used for these values can be reused for the next computations. Of the $n_{L/B}^2$ values in the shared memory, $(n_{PE,v} + n_v - 1) \cdot (n_h - 1)$ values can be reused for the next iteration. Therefore, for the next set of computations, only $n_{L/B}^2 - (n_{PE,v} + n_v - 1) \cdot (n_h - 1)$ new values have to be transferred from the block line memory to the processor. Using the relations between these variables as mentioned above, this number simplifies to $n_{L/B}^2 - n_{L/B} \cdot (n_h - 1)$. To satisfy the real-time constraints, the required data rate from the block line memory to the processor is:

$$R_{D,BL\ to\ MP,b} = R_{D,OUT} \cdot \frac{n_{L/B}^2 - (n_{PE,v} + n_v - 1) \cdot (n_h - 1)}{n_{PE,v} \cdot n_{PE,h}}$$

Using the quantities as given above, we get

$$R_{D,BL\ to\ MP,b} = 6.55 \cdot 10^6\, pels/sec \cdot \frac{64 - (7) \cdot (4)}{16} = 14.73 \cdot 10^6\, pels/sec$$

Exercise 9.3

a.

Comparision of SIMD and MIMD systems

For the throughput of an SIMD system, equation (9.2.5) is used. The general form is

$$R_{T,SIMD} = \frac{R_{T,1}}{P_{SS} + k_{SI}P_{SI} + \frac{P_P + k_D P_D}{n_{PE,SIMD}}}.$$

For this problem, we have $n_{SS} = 0$, i.e., $P_{SS} = 0$, $k_{SI} = 1$, and $k_D = 0$. Equation (9.2.5) therefore simplifies to

$$R_{T,SIMD} = \frac{R_{T,1}}{P_{SI} + \frac{P_P}{n_{PE,SIMD}}}.$$

For an MIMD system, equation (9.2.4) applies. The general form is

$$R_{T,MIMD} = \frac{R_{T,1}}{P_{SS} + \frac{k_{SI}P_{SI}}{n_{PAR,SI}} + \frac{P_P + k_D P_D}{n_{PE,MIMD}}} \text{ with } n_{PAR,SI} = \min(n_{PE,MIMD}, k_{SI}).$$

For the given quantities, equation (9.2.4) simplifies to

$$R_{T,MIMD} = \frac{R_{T,1}}{P_{SI} + \frac{P_P}{n_{PE,MIMD}}}.$$

We assume similar silicon area for both architectures. Using equations (9.2.7) and (9.2.8), and considering that $\beta = \alpha - 1$, we get

$$n_{PE,MIMD} = \alpha + (1 - \alpha) \cdot n_{PE,SIMD}.$$

Since $n_{PAR,SIMD} > 1$ and $1 > \alpha > 0$ it follows that

$$n_{PE,MIMD} < n_{PE,SIMD}.$$

Thus, it follows for the throughput that $R_{T,MIMD} < R_{T,SIMD}$.

b.

Alternative MIMD implementation

The throughput for this implementation alternative is determined by the maximum of the time required for the scalar operations and the parallelizable operations. For a total of $n_{PAR,MIMD}$ processor nodes, the parallelizable operations are distributed across $n_{PAR,MIMD} - 1$ processor nodes. The maximum throughput is achieved if the time to execute the parallelizable operations is less than or equal to the time needed for the scalar operations. The cross-over point is the number of processor elements for which both execution times are equal, i.e.,

$$n_{SI} = \frac{n_P}{n_{PE,MIMD} - 1}$$

and therefore

$$n_{PE,MIMD} = \frac{n_P}{n_{SI}} + 1.$$

If $n_{PAR,MIMD}$ is smaller, the throughput rate is determined by the execution time for the parallelizable operations, otherwise the scalar part determines the throughput rate. For this problem, $n_P/n_{SI} = 16$, and the maximum throughput is achieved for $n_{PE,MIMD} = 17$. Additional processor nodes do not increase the throughput for this implementation!

In order to derive the required formulae, two case must be considered separately.

Case 1: $n_{PE,MIMD} < \frac{n_P}{n_{SI}} + 1 = 17$

A single processor system has a throughput for this application given by

$$R_{T,1} = \frac{1}{T_{CLK} \cdot n_{cyc/op} \cdot (n_{SI} + n_P)}.$$

The throughput in this case is determined by the parallelizable operations, i.e.,

$$R_{T,MIMD,1} = \frac{1}{T_{CLK} \cdot n_{cyc/op} \cdot \frac{n_P}{n_{PE,MIMD}-1}} = \frac{R_{T,1}}{\frac{n_P}{n_{SI}+n_P} \cdot \frac{1}{n_{PE,MIMD}-1}}.$$

Comparing this equation with the corresponding equation for the MIMD system in part a, the new system has a higher throughput rate if

$$\frac{n_P}{n_{PE,MIMD} - 1} < n_{SI} + \frac{n_P}{n_{PE,MIMD}}.$$

Solving for $n_{PE,MIMD}$ yields

$$n_{PE,MIMD} > \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n_P}{n_{SI}}} = 4.53.$$

Case 2: $n_{PE,MIMD} \geq \frac{n_P}{n_{SI}} + 1 = 17$

In that case, the throughput is calculated as

$$R_{T,MIMD,2} = \frac{R_{T,1}}{\frac{n_{SI}}{n_{SI}+n_P}} = 17 \cdot R_{T,1}.$$

Comparing this to $R_{T,MIMD}$ from part a (considering the quantites as given in this part), it can be shown that for all $n_{PAR,MIMD}$ the following relation always holds:

$$R_{T,MIMD,2} > R_{T,MIMD}$$

Therefore, this new implementation yields an increase in throughput compared to the implementation in part a if $n_{PE,MIMD} \geq 5$.

c.

This problem is very similar to part b. Let $n_{PE,MIXED}$ denote the number of data paths implemented, one for the scalar processor and $n_{PE,MIXED} - 1$ for the SIMD processor. The reasoning to determine the throughput rate is exactly the same as presented for the implementation in part b. With

$$P_P = \frac{n_P}{n_P + n_{SI}}$$

and

$$P_{SI} = \frac{n_{SI}}{n_P + n_{SI}}$$

the throughput is determined by

$$R_{T,MIXED}(n_{PE,MIXED}) = \frac{R_{T,1}}{\max(P_{SI}, \frac{P_P}{n_{PE,,MIXED}-1})}.$$

The maximum throughput is achieved if the denominator has a minimum value. Thus, for $P_P = 16/17$ and $P_{SI} = 1/17$, the maximum throughput is obtained for $n_{PE,MIXED} = 17$:

$$R_{T,MIXED,MAX} = 17 \cdot R_{T,1}.$$

The silicon area required for an implementation is calculated as

$$A_{Si,MIXED} = A_{Si,1} + (\alpha + (n_{PE,MIXED} - 1)\beta)\, A_{Si,1}.$$

The efficiency η is given by

$$\eta(n_{PE,MIXED}) = \frac{R_{T,MIXED}(n_{PE,MIXED})}{A_{Si,MIXED}(n_{PE,MIXED})}.$$

For the derivative of η, two cases need to be considered.

Case 1: $n_{PE,MIXED} \geq 17$

With increasing $n_{PE,MIXED}$, $R_{T,MIXED}$ is constant, while the area increases. Thus, η is strictly decreasing and there can be no maximum for the efficiency.

Case 2: $n_{PE,MIXED} < 17$

The first derivative calculates as

$$\frac{\delta\eta}{\delta n_{PE,MIXED}} = \frac{\delta}{\delta n_{PE,MIXED}}\left[\frac{17}{16}\cdot\frac{R_{T,1}}{A_{Si,1}}\cdot\frac{n_{PE,MIXED}-1}{(alpha-\beta+1-\beta n_{PE,MIXED})}\right]$$
$$= \frac{17}{16}\cdot\frac{R_{T,1}}{A_{Si,1}}\cdot\frac{\alpha+1-2\beta}{(\alpha+1-\beta-\beta n_{PE,MIXED})^2}.$$

Since $\alpha = \beta = 0.5$, the derivative is positive for all $2 < n_{PE,MIXED} < 17$. Summarizing, a maximum of the efficiency can therefore only be attained at one of the end points of the interval. Comparing the values for $\eta(2)$ and $\eta(17)$ shows that

$$\eta(2) = \frac{17}{32}\cdot\frac{R_{T,1}}{A_{Si,1}} < \eta(17) = \frac{34}{19}\cdot\frac{R_{T,1}}{A_{Si,1}},$$

i.e., the most efficient solution has one scalar processor and 16 PEs in the SIMD processor.

Exercise 9.4

a.

Algorithms with limited, discrete parallelism

The maximum throughput for a pure SIMD architecture is achieved if the parallelism of the architecture is equal to the maximum of the parallelability of the parallelizable part of the algorithm. In this case, the maximum throughput is achieved for $n_{PE} = 16$.

The maximum efficiency can be derived using

$$\frac{\delta\eta}{\delta n_{PE}} = \frac{\delta\left(\frac{R_{T,SIMD}}{A_{Si,SIMD}}\right)}{\delta n_{PE}} = \frac{\delta\left[\frac{R_{T,1}}{(P_{SI}+\frac{P_P}{n_{PE}})\cdot(\alpha+n_{PE}\cdot\beta)\cdot A_{Si,1}}\right]}{\delta n_{PE}} = 0.$$

Solving the resulting equation for n_{PE} yields

$$n_{PE} = \sqrt{\frac{P_P}{P_{SI}}\cdot\frac{\alpha}{\beta}} = 5.$$

Since $n_{PE} = 5$ is not a valid solution under the given constraints, the resulting efficiency has to be checked for the adjacent values $n_{PE} = 4$ and $n_{PE} = 8$:

$$\frac{\eta(n_{PE}=4)}{\eta(n_{PE}=8)} = \frac{(P_{SI}+\frac{P_P}{8})\cdot(\alpha+8\cdot\beta)}{(P_{SI}+\frac{P_P}{4})\cdot(\alpha+4\cdot\beta)} = \frac{(1+\frac{P_P}{8P_{SI}})\cdot(1+8\cdot\frac{\beta}{\alpha})}{(1+\frac{P_P}{4P_{SI}})\cdot(1+4\cdot\frac{\beta}{\alpha})}.$$

Using the values as given in the problem and $\alpha = \beta$, we get

$$\frac{\eta(n_{PE} = 4)}{\eta(n_{PE} = 8)} = 1.02.$$

Therefore, the maximum efficiency is obtained for $n_{PE} = 4$. For this value, we get an area of

$$A_{Si,SIMD} = (\alpha + 4 \cdot \beta) \cdot A_{Si,1} = 2.5 \cdot A_{Si,1}$$

and a throughput rate of

$$R_{T,SIMD} = \frac{R_{T,1}}{(\frac{1}{26} + \frac{25}{26 \cdot 4})} = 3.58\, R_{T,1}.$$

b.

Again, the maximim throughput can be achieved if the parallelism of the architecture is equal to the maximum parallelability of the algorithm. Therefore, the highest throughput is achieved for $n_{PE} \geq 16$.

For this problem, the throughput rate is determined by the time needed for the parallelizable operations, because we have $n_{PE} < \frac{P_P}{P_{SI}}$, i.e., the execution time for the parallelizable operations will always be longer than the execution time for the sequential operations. The maximum throughput is therefore

$$R_{T,MIXED,1,MAX} = \frac{R_{T,1}}{\frac{25}{26 \cdot 16}} = 16.64\, R_{T,1}.$$

In order to determine the configuration with the highest efficiency, the first derivative of the efficiency is calculated:

$$\frac{\delta\eta}{\delta n_{PE}} = \frac{\delta\left(\frac{R_{T,MIXED,1}}{A_{Si,MIXED,1}}\right)}{\delta n_{PE}} = \frac{\delta\left(\frac{R_{T,1} \cdot (n_{PE} - 1)}{P_P \cdot (2\alpha + n_{PE} \cdot \beta) \cdot A_{Si,1}}\right)}{\delta n_{PE}} = \frac{R_{T,1}}{A_{Si,1}} \cdot \frac{(2\alpha + \beta)}{P_P \cdot (2\alpha + n_{PE}\beta)^2}.$$

Thus,

$$\frac{\delta\eta}{\delta n_{PE}} > 0 \text{ for all } n_{PE},$$

and the maximum efficiency is obtained for the maximum number of processing elements for the parallelizable operations, i.e,

$$n_{PE} = 16 + 1 = 17.$$

The throughput rate in this case is given above, the area is given by

$$A_{Si,SIMD} = (2 \cdot \alpha + (16 + 1) \cdot \beta) \cdot A_{Si,1} = 11.5 \cdot A_{Si,1}.$$