Solution 1: local copies of each matrix

```
void multiply(a_b_t a[I][K], a_b_t b[J][K], r_t r[I][J]) {
    #pragma HLS INTERFACE axis register both port=r
    #pragma HLS INTERFACE axis register both port=b
    #pragma HLS INTERFACE axis register both port=a

    a_b_t atmp[I][K], btmp [K][J];
    r_t rtmp[I][J];

Row_a_copy: for(int i=0; i<I; i++)
    Col_a_copy: for(int k=0; k<K; k++)
        atmp[i][k] = a[i][k];

Row_b_copy: for(int k=0; k<K; k++)
    Col_b_copy: for(int j=0; j<J; j++)
        btmp[k][j] = b[k][j];

Row: for(int i=0; i<I; i++)
    Col: for(int j=0; j<J; j++) {
        rtmp[i][j] = 0;
        Product: for(int k=0; k<K; k++)
            rtmp[i][j] += atmp[i][k] * btmp[k][j];
    }

Row_res_copy: for(int i=0; i<I; i++)
    Col_res_copy: for(int j=0; j<J; j++)
        r[i][j] = rtmp[i][j];

}
```

multiply:

Latency		Interval		
min	max	min	max	Type
553476	553476	553476	553476	none

loops:

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_a_copy	4224	4224	66	-	-	64	no
+ Col_a_copy	64	64	1	-	-	64	no
- Row_b_copy	4224	4224	66	-	-	64	no
+ Col_b_copy	64	64	1	-	-	64	no
- Row	532608	532608	8322	-	-	64	no
+ Col	8320	8320	130	-	-	64	no
++ Product	128	128	2	-	-	64	no
- Row_res_copy	12416	12416	194	-	-	64	no
+ Col_res_copy	192	192	3	-	-	64	no

Solution 2: pipeline all innermost loops

```
void multiply(a_b_t a[I][K], a_b_t b[J][K], r_t r[I][J]) {
#pragma HLS INTERFACE axis register both port=r
#pragma HLS INTERFACE axis register both port=b
#pragma HLS INTERFACE axis register both port=a

    a_b_t atmp[I][K], btmp [K][J];
    r_t rtmp[I][J];

Row_a_copy: for(int i=0; i<I; i++)
    Col_a_copy: for(int k=0; k<K; k++)
        #pragma HLS PIPELINE
        atmp[i][k] = a[i][k];

Row_b_copy: for(int k=0; k<K; k++)
    Col_b_copy: for(int j=0; j<J; j++)
        #pragma HLS PIPELINE
        btmp[k][j] = b[k][j];

Row: for(int i=0; i<I; i++)
    Col: for(int j=0; j<J; j++) {
        rtmp[i][j] = 0;
        Product: for(int k=0; k<K; k++)
            #pragma HLS PIPELINE
            rtmp[i][j] += atmp[i][k] * btmp[k][j];
    }

Row_res_copy: for(int i=0; i<I; i++)
    Col_res_copy: for(int j=0; j<J; j++)
        #pragma HLS PIPELINE
        r[i][j] = rtmp[i][j];

}
```

multiply:

Latency		Interval		
min	max	min	max	Type
548872	548872	548872	548872	none

loops:

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_a_copy_Col_a_copy	4096	4096	1	1	1	4096	yes
- Row_b_copy_Col_b_copy	4096	4096	1	1	1	4096	yes
- Row_Col	536576	536576	131	-	-	4096	no
+ Product	128	128	2	2	1	64	yes
- Row_res_copy_Col_res_copy	4097	4097	3	1	1	4096	yes

- loops automatically flattened (saves 2 cycles)
- can’t pipeline Product due to dependency

Solution 3: temporary register

```
Row: for(int i=0; i<I; i++)
    Col: for(int j=0; j<J; j++) {
        r_t acc = 0;
        Product: for(int k=0; k<K; k++)
            #pragma HLS PIPELINE
             acc += atmp[i][k] * btmp[k][j];
        rtmp[i][j] = acc;
    }
```

multiply:

Latency		Interval		
min	max	min	max	Type
274441	274441	274441	274441	none

loops:

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_a_copy_Col_a_copy	4096	4096	1	1	1	4096	yes
- Row_b_copy_Col_b_copy	4096	4096	1	1	1	4096	yes
- Row_Col_Product	262144	262144	2	1	1	262144	yes
- Row_res_copy_Col_res_copy	4097	4097	3	1	1	4096	yes

Solution 4: unroll M=8

```
void multiply(a_b_t a[I][K], a_b_t b[J][K], r_t r[I][J]) {
    #pragma HLS INTERFACE axis register both port=r
    #pragma HLS INTERFACE axis register both port=b
    #pragma HLS INTERFACE axis register both port=a

    a_b_t atmp[I][K], btmp [K][J];
    #pragma HLS ARRAY_RESHAPE variable=atmp cyclic factor=4 dim=2
    #pragma HLS ARRAY_RESHAPE variable=btmp cyclic factor=4 dim=1
    r_t rtmp[I][J];

    // copy A, copy B

Row: for(int i=0; i<I; i++)
    Col: for(int j=0; j<J; j++) {
        r_t acc = 0;
        Product: for(int k=0; k<K; k++)
            #pragma HLS PIPELINE
            #pragma HLS UNROLL factor=8
            acc += atmp[i][k] * btmp[k][j];
        rtmp[i][j] = acc;
    }

    // copy R

}
```

multiply:

Latency		Interval		
min	max	min	max	Type
57352	57352	57352	57352	none

loops:

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_a_copy_Col_a_copy	4096	4096	4	4	1	1024	yes
- Row_b_copy_Col_b_copy	4096	4096	1	1	1	4096	yes
- Row_Col	45056	45056	11	-	-	4096	no
+ Product	8	8	2	1	1	8	yes
- Row_res_copy_Col_res_copy	4097	4097	3	1	1	4096	yes

Solution 5: Tiling

```
void multiply(a_b_t a[I][K], a_b_t b[J][K], r_t r[I][J]) {
    #pragma HLS INTERFACE axis register both port=r
    #pragma HLS INTERFACE axis register both port=b
    #pragma HLS INTERFACE axis register both port=a

    a_b_t atmp[I][K], btmp [K][J];
    #pragma HLS ARRAY_RESHAPE variable=atmp cyclic factor=4 dim=2
    #pragma HLS ARRAY_RESHAPE variable=btmp cyclic factor=4 dim=2
    r_t rtmp[I][J];
    #pragma HLS ARRAY_RESHAPE variable=rtmp cyclic factor=4 dim=2

    // copy A, copy B

TileRow: for(int ii=0; ii<I/TI; ii++)
    TileCol: for(int jj=0; jj<J/TJ; jj++)
        TileProduct: for(int kk=0; kk<K/TK; kk++)
            #pragma HLS PIPELINE
            Row: for(int i=0; i<TI; i++)
                Col: for(int j=0; j<TJ; j++) {
                    r_t acc = (kk == 0) ? 0 : rtmp[ii*TI + i][jj*TJ + j];
                    Product: for(int k=0; k<TK; k++)
                        acc += atmp[ii*TI + i][kk*TK + k] *
                               btmp[kk*TK + k][jj*TJ + j];
                    rtmp[ii*TI + i][jj*TJ + j] = acc;
                }

    // copy R

}
```

multiply:

Latency		Interval		
min	max	min	max	Type
20489	20489	20489	20489	none

loops:

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Row_a_copy_Col_a_copy	4096	4096	4	4	1	1024	yes
- Row_b_copy_Col_b_copy	4096	4096	4	4	1	1024	yes
- TileRow_TileCol_TileProduct	8192	8192	16	16	1	512	yes
- Row_res_copy_Col_res_copy	4097	4097	6	4	1	1024	yes

Solution 6: Dataflow

```
void multiply(a_b_t a[I][K], a_b_t b[J][K], r_t r[I][J]) {
    #pragma HLS INTERFACE axis register both port=r
    #pragma HLS INTERFACE axis register both port=b
    #pragma HLS INTERFACE axis register both port=a

    #pragma HLS DATAFLOW
    ...tiling solution...
}
```

multiply:
interval = 8195

Summary

Solution	Interval	Clock Period ns	BRAM	DSP	FF	LUT
1	553,476	3.770	8	1	264	695
2	548,872	5.007	8	1	274	918
3	274,441	5.007	8	1	285	1052
4	57,352	6.466	8	4	332	1581
5	20,489	7.733	8	320	7,955	13,579
6	8,195	8.586	8	320	8,751	13,672

- results are estimates – not synthesized
- unrolling M=8 performs 8 multiplications in parallel
- tiling performs 8x8x8=512 multiplications in parallel